UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANDERSON APOLÔNIO LIRA QUEIROZ

SIMA-IoV – Arquitetura e Mecanismo para a Seleção Inteligente do Método de Autenticação na Internet de Veículos 5G

Recife

2024

ANDERSON APOLÔNIO LIRA QUEIROZ

SIMA-IoV – Arquitetura e Mecanismo para a Seleção Inteligente do Método de Autenticação na Internet de Veículos 5G

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Redes de Computadores

Orientador (a): Kelvin Lopes Dias

Recife

2024

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Queiroz, Anderson Apolonio Lira.
SIMA-IoV ? Arquitetura e Mecanismo para a Seleção Inteligente do Método de Autenticação na Internet de Veículos 5G / Anderson Apolonio Lira Queiroz. - Recife, 2024.
192f.: il.

Tese (Doutorado), Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-graduação em Ciência da Computação, 2024.
Orientação: Kelvin Lopes Dias.

1. Internet de Veículos; 2. Computação de Borda; 3. Tecnologia 5G; 4. Autenticação; 5. Otimização Bayesiana. I. Dias, Kelvin Lopes. II. Título.

UFPE-Biblioteca Central CDD 004.6

Folha de aprovação: Inserir a folha de aprovação enviada pela Secretaria do curso de Pós-Graduação. A folha deve conter a **data de aprovação**, estar **sem assinaturas** e em formato **PDF**.

Com imenso respeito e admiração, dedico esta pesquisa de tese a todas as mães que, em sua luta diária nos hospitais, abandonam suas vidas e objetivos para viverem uma solitária esperança de cura das doenças de seus filhos. A resiliência, o amor incondicional e coragem que demonstram diante destes desafios tão profundos são uma fonte inesgotável de força, dedicação e inspiração imensuráveis.

AGRADECIMENTOS

Primeiramente, agradeço a minha mãe, Marluce Rosa Queiroz, por todo amor e apoio durante minha jornada de vida. Agradeço a Deus pela saúde e controle emocional que me foi concebido para que eu pudesse trabalhar, estudar, cuidar da minha filha e afazeres domésticos, buscando evoluir como, profissional, pai, amigo, companheiro e pesquisador.

Agradeço ao Professor Kelvin Dias, por sua paciência, orientação e todo apoio desde a época dos projetos da rede nacional de pesquisas (RNP) e dos trabalhos na equipe de infraestrutura do Núcleo de Tecnologia da informação da NTI/UFPE (2013). Com a convicção de que nossa parceria continuará firme pelo tempo que for necessário, com a bênção de Deus.

Agradeço especialmente a minha amiga Maria Katarine e o amigo Eduardo Oliveira membros do IANG (Intelligent and Advanced Networking research Group), por nossos muitos dias de trabalho com inúmeros obstáculos, muitas discussões e longos períodos de implementações, testes e integração das soluções para obtenção dos resultados, grato por suas parcerias inestimáveis.

Aos amigos e parentes, David Mota, Danilo Clemente, Luiz Felipe, Emerson Queiroz, Sophia Nair, Marilene Francisca, Sergio Limeira e todos os que de alguma forma colaboraram e apoiaram esta pesquisa.

Gostaria de agradecer a Coordenação de Comunicação de Redes, na pessoa do Sr Eduardo Ramos, da Superintendência de Tecnologia da Informação(STI/UFPE) por todo apoio da infraestrutura computacional e de redes ofertados para as simulações desta tese assim como ao Centro de Informática da Universidade Federal de Pernambuco por proporcionar o espaço de laboratório para o desenvolvimento e implementação desta pesquisa.

Por fim, muito obrigado aos membros da defese de qualificação e da banca de defesa de tese por sua disponibilidade, participação e dedicação na análise desta tese de doutorado, cada observação, críticas e pontos de vista serão considerados como valorosas contribuições à evolução desta pesquisa de tese e do pesquisador.

"A alegria está na luta, na tentativa, na provação envolvida e não na vitória propriamente dita." (Mahatma Gandhi)

RESUMO

O mercado de veículos conectados cresceu substancialmente nos últimos anos, fortalecendo a Internet dos Veículos (IoV). Esse ecossistema viabiliza a conectividade entre veículos, infraestrutura dos sistemas de transporte inteligentes (ITS) e dispositivos de pedestres, além de permitir acesso a serviços de Internet e recursos de nuvens computacionais. Entretanto, a adoção mais ampla da IoV, com todo o seu potencial inovador para as cidades inteligentes, depende de estratégias eficazes de segurança e privacidade. Um dos pilares fundamentais para garantir segurança na IoV é o processo de autenticação. Por um lado, a autenticação deve assegurar que apenas veículos autorizados obtenham conectividade, armazenamento e processamento na IoV. Por outro lado, este processo não deve comprometer a qualidade de serviço (QoS) de aplicações críticas, como streaming de vídeo dos passageiros e atualizações remotas de software automotivas, conhecidas como *over-the-air* (OTA). O desafio desta pesquisa é equilibrar a necessidade de segurança com a manutenção da QoS em um ambiente de alta mobilidade e densidade variável de veículos. O estudo avaliou estratégias de autenticação para veículos em redes móveis 5G-V2X, comparando abordagens centralizadas, baseadas em autoridades de confiança (TA), e soluções distribuídas com suporte de blockchain, utilizando os algoritmos de consenso PoW (Proof of Work) e PoS (Proof of Stake). Foi proposta uma arquitetura em camadas para IoV, incorporando os dois métodos de autenticação em ambientes de computação de borda (edge) e nevoeiro (fog). Os mecanismos foram implementados utilizando os simuladores OMNeT++, Veins, SIMU5G, INET e SUMO, com dados de tráfego da Avenida Agamenon Magalhães, Recife-PE, Brasil, para capturar a variabilidade do ambiente real. O objetivo foi desenvolver um arcabouço inteligente para a tomada de decisão ciente de contexto na escolha do método de autenticação mais adequado para a IoV, utilizando otimização bayesiana para melhorar a segurança e a eficiência operacional dos sistemas de transporte inteligente. A solução proposta, SIMA-IoV, seleciona dinamicamente o método de autenticação (centralizado ou descentralizado) mais adequado com base nos dados coletados. Os resultados das simulações indicam que a computação de borda distribuída entre as estações rádio base 5G garantiu tempos de autenticação na ordem de milissegundos, mantendo a continuidade dos serviços em ambientes de alta mobilidade. O SIMA-IoV demonstrou que a autenticação baseada em blockchain com PoS é mais estável para cenários de alta densidade de veículos, enquanto a TA apresentou melhor desempenho em cenários de baixa densidade. Já o PoW mostrou instabilidade, destacando suas limitações em ambientes IoV de alta demanda.

Os resultados obtidos foram inovadores, mostrando um avanço significativo na otimização do serviço de autenticação em redes móveis veiculares.

Palavras-chave: Internet de Veículos, Autenticação, Computação de Borda, Tecnologia 5G e Otimização Bayesiana.

ABSTRACT

The market for connected vehicles has grown substantially in recent years, strengthening the Internet of Vehicles (IoV). This ecosystem enables connectivity between vehicles, intelligent transportation system (ITS) infrastructure, and pedestrian devices while also providing access to Internet services and cloud computing resources. However, the broader adoption of IoV, with all its innovative potential for intelligent cities, depends on adequate security and privacy strategies. The authentication process is one of the fundamental pillars of IoV security. On the one hand, authentication must ensure that only authorized vehicles gain connectivity, storage, and processing within the IoV. On the other hand, this process must maintain the quality of service (QoS) for critical applications, such as passenger video streaming and remote automotive software updates, known as over-the-air (OTA). The challenge of this research is to balance the need for security with the maintenance of QoS in an environment characterized by high mobility and variable vehicle density. The study evaluated authentication strategies for vehicles in 5G-V2X mobile networks, comparing centralized approaches based on trusted authorities (TA) with distributed solutions supported by blockchain, using the PoW (Proof of Work) and PoS (Proof of Stake) consensus algorithms. A layered architecture for IoV was proposed, incorporating authentication methods in edge and fog computing environments. The mechanisms were implemented using the OMNeT++, Veins, SIMU5G, INET, and SUMO simulators, with traffic data from Avenida Agamenon Magalhães in Recife, PE, Brazil, to capture the variability and complexity of the natural environment. The objective was to develop an intelligent framework for context-aware decision-making in selecting the most appropriate authentication method for IoV, using Bayesian optimization to enhance security and operational efficiency in intelligent transportation systems. The proposed solution, SIMA-IoV, dynamically selects the most suitable authentication method (centralized or decentralized) based on the collected data. Simulation results indicate that distributed edge computing among 5G base stations achieved authentication times on the order of milliseconds, ensuring service continuity in high-mobility environments. SIMA-IoV demonstrated that blockchain-based authentication with PoS is more stable for high-density vehicle scenarios, while TA showed better performance in low-density scenarios. Conversely, PoW exhibited instability, highlighting its limitations in high-demand IoV environments. The results were innovative, demonstrating significant advancements in optimizing the authentication service in vehicular mobile networks.

Keywords: Internet of Vehicles, Authentication, Edge Computing, 5G Technology and Bayesian Optimization.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3G	*Third Generation of Mobile Networks*
4GLTE	*Fourth Generation of Mobile Networks/Long-Term Evolution*
5G	*Fifth Generation of Mobile Networks*
API	*Application Programming Interface*
AR	*Augmented Reality*
AUSF	*Authentication Server Function*
BC	*Block Chain*
BS	*Base Stations*
CA	*Certificate Authority*
DLT	*Distributed Ledger Technology*
eMBB	*Enhanced Mobile Broadband*
IA	*Artificial intelligence*
IoT	Internet das Coisas
IoV	Internet de Veículos
ITS	*Intelligent Transport Systems*
mMTC	Machine Type Mass Communication
NFV	*Network Functions Virtualization*
ns-3	Simulador de Redes v.3
OB	Otimização Bayesiana
OBU	*On-Board Computerized Unit*
OMNeT++	Simulador de Eventos Discretos
OTA	Over-The-Air
PoS	*Proof of Participation*
PoW	*Proof of Work*
QoE	*Quality of Experience*

QoS	*Quality of Service*
RAN	*Open Radio Access Network*
RSU	*Roadside Unit*
SDN	*Software Defined Networks*
SIMA-IoV	Arquitetura e Mecanismo para a Seleção Inteligente do Método de Autenticação na Internet de Veículos 5G
TA	*Trusted Authority*
TPE	*Panzen Tree Estimator*
uRLLC	*Ultra Reliable and Low Latency Communication*
V2I	*Vehicle for Infrastructure*
V2V	*Vehicle to Vehicle*
V2X	*Vehicle to Everything*
VANET	*Veicular Ad Hoc Networks*
VEC	*Vehicular Edge Computing*
VoLTE	*Voice over LTE*
VR	*Virtual Reality*

SUMÁRIO

1 CAPÍTULO INTRODUÇÃO

Neste capítulo são apresentados o contexto e motivação, os objetivos e contribuições da pesquisa, e por último, são apresentadas a organização e estrutura da tese.

1.1 CONTEXTO E MOTIVAÇÃO

A Internet de Veículos (IoV) (YANG et al., 2014) surgiu como uma das aplicações mais relevantes da Internet das Coisas (IoT) (ATZORI; MORABITO, 2010), uma evolução das *Veicular Ad Hoc Networks* (VANET) *Ad-hoc* (HARTENSTEIN; LABERTEAUX, 2008). Suas características abrangem uma gama mais ampla de funções, incluindo a disponibilização de serviços, o compartilhamento de dados e a segurança das aplicações dentro do ecossistema dos *Intelligent Transport Systems* (ITS). De acordo com os dados apresentados pela revista *Business Insider Intelligence*[1], o número de veículos conectados deverá sofrer um aumento de 133% em 8 anos, ultrapassando 77 milhões de unidades até 2026. Diante deste cenário, cresce a responsabilidade e o compromisso tecnológico com requisitos relacionados à eficiência dos serviços utilizados para tal ambiente. Assim como nas VANETs, a IoV permite diferentes modos de comunicação entre os objetos conectados: *Vehicle for Infrastructure* (V2I), *Vehicle to Vehicle* (V2V) e *Vehicle to Everything* (V2X), isto como suporte de processamento, armazenamento e conectividade estruturados para atender todo o ecossistema ITS (GAO et al., 2020).

O ambiente das cidades inteligentes se beneficia da IoV com os seus diversos elementos constituintes, como as estações de comunicação, postos governamentais de fiscalização aduaneira e segurança pública ou privada, pedágios, unidades de salvaguarda e regastes, veículos motorizados ou a propulsão humana, equipamentos de sinalização interativos e os próprios pedestres com seus dispositivos IoT (CASTILLO; IBAñEZ, 2018a). As IoVs são compostas por um conjunto tecnológico de diversos artefatos, tais como: Centro de Dados, Aplicações, Dispositivos de Comunicação, *Roadside Unit* (RSU) e *Base Stations* (BS), Sensores, Câmeras, Semáforos, Placas de Trânsito Interativas, Veículos de Transporte, *On-Board Computerized Unit* (OBU) e múltiplas tecnologias de comunicação em rede sem fio, 802.11p, 802.15, 3G, 4G/LTE e 5G/NR (ZHANG; LI; CUI, 2018a).

Com a implantação atual das redes no padrão *Fifth Generation of Mobile Networks* (5G) ocorrendo gradualmente, a IoV terá um impulso significativo com os benefícios proporciona-

[1] https://www.businessinsider.com/iot-connected-smart-cars

dos pelos novos atributos do padrão de comunicação de quinta geração, *Ultra Reliable and Low Latency Communication* (uRLLC), altas taxas de transmissão com a *Enhanced Mobile Broadband* (eMBB) e Machine Type Mass Communication (mMTC) ampliando e melhorando questões pertinentes à ubiquidade e à pervasividade (PANWAR; SHARMA; SINGH, 2016). Cada vez mais os serviços da Internet estão sendo requisitados por um número maior de veículos conectados. Esses serviços envolvem processamento na nuvem, informações sobre o trânsito, alertas, sistemas de entretenimento a bordo e segurança dos passageiros, pedestres e das vias. A atualização de *software* via Over-The-Air (OTA) é um importante mecanismo para distribuir e instalar atualizações de novas funcionalidade e correções de erros sem a necessidade de conexão física a um computador. Essa tecnologia é amplamente utilizada em dispositivos móveis, a atualização via OTA permite que fabricantes corrijam bugs, adicionem novas funcionalidades e otimizem o desempenho de maneira remota. Isso não só proporciona uma experiência mais segura e atualizada para os usuários, mas também reduz custos e complexidades associadas a manutenção e a evolução tecnológica do ecossistema.

A Internet de Veículos é um ambiente essencialmente dinâmico e complexo, onde os veículos se conectam e participam dinamicamente das redes móveis sem fio. Desta forma necessitam de mecanismos que forneçam garantias de desempenho e qualidade dos serviços e aplicações. A possibilidade de falhas podem ocasionar diversos tipos de sinistros que tanto preocupam as organizações, sociedade civil e a indústria automobilística. Um ponto crucial na IoV são os métodos de autenticação e gerência de chaves por representarem serviços fundamentais para segurança das redes de comunicação no ambiente das IoVs. Por esta razão, tais métodos e serviços quando implementados ou utilizados de maneira inadequada podem impactar diretamente nas garantias destes requisitos (BRUNNER et al., 2020). Por exemplo, os métodos de autenticação e os sistemas criptográficos apresentam requisitos distintos, operando frequentemente de maneira colaborativa para implementação do serviço de proteção e segurança dos dados. Assim, a seleção e aplicação de determinado método de autenticação combinada com determinado algoritmo criptográfico pode resultar em comportamentos distintos, isto, dependendo de cada cenário ou dos requisitos das aplicações e serviços disponibilizados. Ao considerar as alternativas entre as arquiteturais de serviços, sejam eles centralizados ou descentralizados, é imperativo avaliar também as distintas formas de implementação, capacidade de infraestrutura, carga de tráfego, número de dispositivos e a qualidade das tecnologias de comunicação disponíveis (YANG et al., 2014).

Contudo, problemas de segurança entre veículos nos diversos modos de comunicação (V2I,

V2V e V2X), exigem soluções robustas de autenticação que garantam a integridade dos dados e a confiança entre os diferentes participantes. Abordar de maneira abrangente as questões relacionadas à elevada densidade de dispositivos de comunicação nos futuros ambientes das cidades inteligentes representa um desafio significativo. Aprofundar pesquisas que envolvam soluções de decisão inteligentes é essencial para garantir o sucesso e a ampla adoção de serviços e aplicações nesses ecossistemas conectados. Em particular, os métodos de autenticação desempenham um papel crucial com relação à garantia e proteção das informações transmitidas em um cenário cada vez mais complexo e interconectado (ALI; LI, 2019).

Esta tese está inserida no contexto da autenticação no ambiente da Internet dos Veículos, propondo uma solução inteligente que seleciona o método apropriado para não impactar os requisitos críticos do ambiente IoV, como atraso, perdas e eficiência computacional. Ao utilizar técnicas de Inteligência Artificial através da Otimização Bayesiana, a pesquisa desenvolve um mecanismo para seleção e definição do método de autenticação que não apenas protega as comunicações, mas também garanta a manutenção do tempo de autenticação da baixa latência (*delay*) e alta vazão (*throughput*). Essa abordagem busca otimizar a utilização dos recursos disponíveis, proporcionando maior robustez e eficiência às condições variáveis do ambiente IoV.

1.2 OBJETIVOS

O objetivo geral deste trabalho é desenvolver uma Arquitetura e Mecanismo para a Seleção Inteligente do Método de Autenticação na Internet de Veículos 5G (SIMA-IoV). Para chegar ao objetivo geral, alguns objetivos específicos precisaram ser alcançados:

I. Identificar o algoritmo de aprendizado de máquina para seleção do método autenticação;

II. Estruturar uma arquitetura de redes para suporte dos serviços de autenticação no ambiente IoV;

III. Elaborar um ambiente de simulação e integração das ferramentas e soluções para IoV;

1.3 CONTRIBUIÇÕES DA PESQUISA

Com base no que foi exposto, as principais contribuições desta tese incluem:

I. Desenvolvimento dos métodos de autenticação centralizado e descentralizado para o Ambiente IoV;

II. Concepção do mecanismo inteligente baseado em otimização bayesiana para seleção do método apropriado de autenticação para o ambiente IoV (SIMA-IoV).

1.4 ESTRUTURA DA TESE

A estrutura desta tese segue à disposição em capítulos conforme apresentado:

Capítulo 2 - Apresenta os fundamentos teóricos da pesquisa, abordando os conceitos básicos dos elementos necessários para uma compreensão aprofundada da tese;

Capítulo 3 - Explora os estudos relacionados às soluções de autenticação tanto centralizadas quanto descentralizadas no ambiente da Internet Veicular(IoV);

Capítulo 4 - Apresenta e discute o mecanismo inteligente para seleção do método de autenticação baseado no contexto do ambiente IoV em redes 5G;

Capítulo 5 - Demonstra os aspectos do ambiente concebido, as ferramentas de simulação utilizadas e os métodos de autenticação implementados para IoV;

Capítulo 6 - Analisa os resultados das simulações dos métodos de autenticação e do mecanismo inteligente de seleção do método adequado de autenticação, SIMA-IoV;

Capítulo 7 - Sumariza a pesquisa, apresenta a argumentação sobre os resultados obtidos pelo mecanismo SIMA-IoV e exibe as principais limitações da pesquisa e apresenta os trabalhos futuros.

2 CAPÍTULO FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma revisão dos conceitos fundamentais e das tecnologias que apoiam a pesquisa desta tese. Inicialmente, discute-se o papel das redes 5G no fortalecimento da Internet de Veículos (IoV), Em seguida, são abordadas as características da computação de borda, essencial para o processamento de dados próximo ao local de geração, reduzindo latências e aumentando a eficiência dos serviços na IoV. Adicionalmente os esquemas de proteção no contexto da IoV são apresentados, incluindo infraestrutura de chaves públicas (PKI) e os diferentes métodos de autenticação.

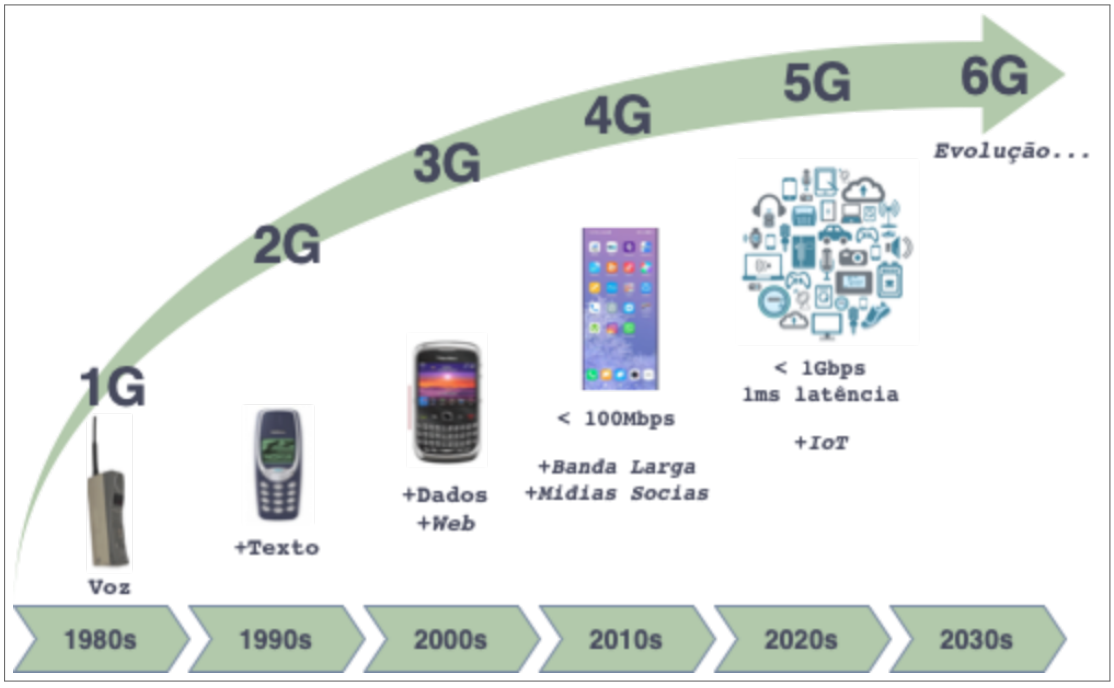
2.1 REDES DE COMUNICAÇÃO 5G

Com a evolução da sociedade conectada e a crescente utilização em massa das redes sem fio, surgem inúmeros desafios dos sistemas de comunicação no intuito de conseguirem entregar tecnologias de múltiplos níveis com recursos flexíveis aos diversos usuários e ambientes existentes. Foi a partir das redes no padrão 3G que a banda larga móvel se popularizou, permitindo que os usuários assistissem filmes, enviassem e-mails e participassem de mídias sociais. Com o 4G foram aprimorados recursos de largura de banda, tempo de resposta e segurança da informação, possibilitando a utilização de novos serviços e aplicações nesta tecnologia de rede. As redes de nova geração "5G", buscam atender as questões que vão além da melhoria do desempenho e segurança, ela visa a melhoria da disponibilidade de conexão, da capacidade de atendimento a diversas categorias de serviços e aplicações e da possibilidade de segmentação da rede conforme objetivos do negócio e/ou parâmetros do ambiente, e usuários (MATTISSON, 2017).

Contudo, por conta das novas demandas dos vídeos em tempo real que exigem melhores requisitos de vazão e menores latências e perdas, a *Third Generation of Mobile Networks* (3G) passou a apresentar limitações em sua utilização (EZHILARASAN; DINAKARAN, 2017). Diante disto, foi dado início a Quarta Geração de Redes Móveis (*Fourth Generation of Mobile Networks/Long-Term Evolution* (4GLTE)), implementando uma melhor qualidade de acesso, transmissão com mais segurança e uma arquitetura com velocidades até 10 vezes maiores que o seu antecessor, o 3G. Propôs o lema "*Anytime and Anywhere*" para os usuários, permitindo acesso a dados multimídia como vídeos e chamadas de voz (*Voice over LTE* (VoLTE)) me-

nos limitados. Contudo, o 4G ainda possuía algumas restrições. Estas surgem do amplo uso das tecnologias móveis e dispositivos celulares, do crescimento da Internet das Coisas (IoT), bem como das evoluções e derivações de outros ecossistemas de comunicação (IoV/UaV), e também do avanço dos modelos de comunicações Máquina-a-Máquina (M2M) e Dispositivo-a-Dispositivo (D2D) existentes. Estima-se que haja milhões de dispositivos por quilômetro quadrado, todos demandando uma variedade de serviços e aplicativos da rede (ATZORI; MORABITO, 2010). No modelo da arquitetura de rede 4G/LTE, todos os serviços de rede operam em uma única infraestrutura (*core*). Dessa forma, para acomodar uma massiva quantidade de dispositivos, seria necessário implantar múltiplas infraestruturas de rede para ampliar sua capacidade toda vez que for preciso (OLIVEIRA; ALENCAR; LOPES, 2018).

Figura 1 – Evolução das redes móveis celulares



Fonte: Autor

Como pode ser visto na Figura 1, a Quinta Geração de Redes Móveis (5G), assim como todas as anteriores, surge para agregar novas funcionalidades a fim de atender novos requisitos e novas demandas de serviços e aplicações conforme suas evoluções e sistemas de integração. Além das questões relacionadas a IoT, veículos autônomos, veículos aéreos não tripulados, *Virtual Reality* (VR) / *Augmented Reality* (AR) (EZHILARASAN; DINAKARAN, 2017)(OLIVEIRA; ALENCAR; LOPES, 2018).

O 5G representa a união e evolução de várias tecnologias de redes sem fio de banda larga. Os objetivos específicos do 5G são: taxa de dados superior a 10 Gbps para as aplicações, latência

de rede abaixo de 1 milissegundo, alta capacidade de expansão do números de dispositivos e ganhos de eficiência energética (MATTISSON, 2017). A Tabela 1 apresenta a evolução básica das tecnologias de rede sem fio móveis.

Tabela 1 – Evolução das Redes Móveis

Geração	Tecnologia de Acesso	Taxa de Dados	Banda de Frequência	Largura de Banda	Comutação	Aplicação
1G	Serviço de telefonia móvel avançado (FDMA)	2.4Kbps	800MHz	30KHz	Circuito	Voz
2G	Sistema global para celular (GSM) (TDMA/CDMA)	10Kbps	800/900/ 1800/1900 MHz	200KHz 1.25MHz	Circuito	Voz e Dados
3G	Sistema Telecomunicação Móvel Universal (UMTS) (WCDMA)	384Kbps	800/900/ 1800/1900/ 2100MHz	1.25MHz 5 MHz	Circuito e Pacote	Voz Dados Chamadas de Vídeo
3.75G	Evolução a Longo Prazo (LTE)(OFDMA/SC-FDMA) Interoperabilidade mundial para acesso a micro-ondas (WIMAX)	100-200 Mbps	1.8 GHz 2.6 GHz 3.5 GHz 5.8 GHz	1.4-20 MHz 3.5MHz 10MHz	Pacote	Voz Dados Jogos Vídeos
4G	Evolução a Longo Prazo (LTE) (OFDMA/SC-FDMA) Interoperabilidade mundial para acesso a micro-ondas (WIMAX)(SOFMDA)	3Gbps D.L 1.5Gbps U.L 100-200 Mbps	1.8 GHz 2.6 GHz 2.3 GHz 2.5 GHz 3.5 GHz	1.4-20 MHz 3.5MHz 5-7MHz 8.7MHz 10 MHz	Pacote	Voz Dados Jogos Vídeos de Alta Definição
5G	**Acesso múltiplo por divisão de faixa (BDMA) Acesso multiportadora não ortogonal ou de banco de filtros (FBMC)**	**10-50 Gbps**	**1.8 GHz 2.6 GHz esperado 30-300 GHz**	**60 GHz**	**Pacote**	**Ultra Definição Realidade Virtual Aplicações Críticas**

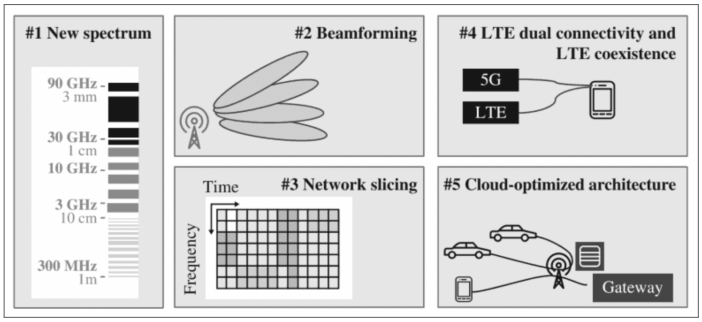
Fonte: (Adaptado de GUPTA; JHA, 2015, p. 1210)

A nova geração 5G representa um imenso passo com relação às capacidades de recursos nas redes móveis. O objetivo principal da nova geração de comunicação móvel é elevar os atuais níveis de comunicação móvel sem fio ao extremo, isto em termos de taxas de transferência de dados, latência, capacidade de dispositivos e disponibilidade da rede (TOSKALA; NAKAMURA, 2020).

A Figura 2, apresenta os principais componentes da tecnologia 5G que serão detalhados:

1. Novo Espectro (New espectrum), o 5G possui um desenho fléxivel da banda com va-

Figura 2 – Componentes Chaves da Tecnologia 5G



Fonte: (TOSKALA; NAKAMURA, 2020)

riação de 400 MHZ até 90 GHZ englobando as faixas licenciadas, não licenciadas e compartilhadas;

2. Formação do Feixe (Beamforming), é uma técnica utilizada em ondas milimétricas que visa diminuir as interferências geradas por um conjunto de antenas e sinais de entrada e saída de determinada localização;

3. Fatiamento da Rede (Network slicing), a possibilidade da segmentação física e lógica da rede 5G permite a rede flexibilidade para suportar diferentes casos de uso, frequências de banda, eficiência energética e espectral;

4. Coexistencia de Tecnologias de Redes Móveis (Coexistence LTE), a viabilidade de conexão dupla com a coexistência das tecnologias 4G e 5G em conjunto facilitam a implementação e utilização inicial oportunizando a migração para um sistema totalmente autônomo 5G/NR posteriormente;

5. Arquitetura otimizada para Nuvem (Cloud-optimized architecture), com os requisitos e compromissos dos diversos ambientes, serviços e aplicações nas redes 5G permitem que sejam realizados ajustes arquiteturais com objetivo de melhor atender as características e parâmetros de determinados processos da transmissão e comunicação;

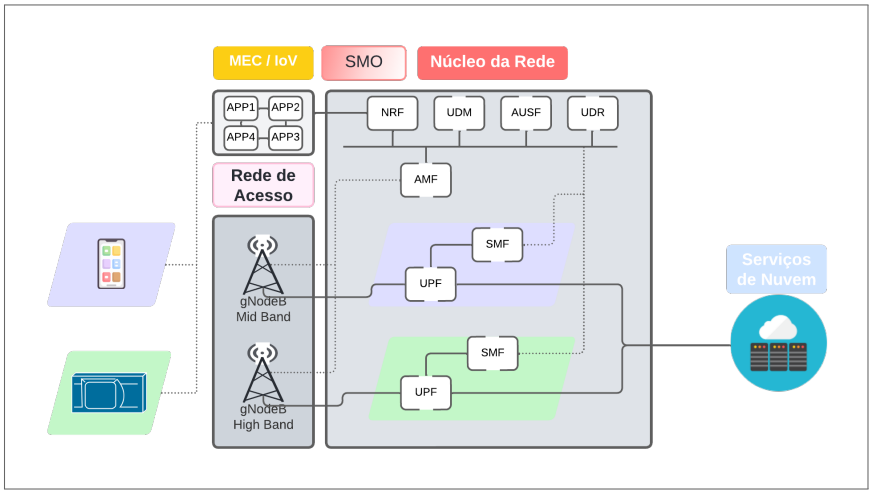
A arquitetura 5G é projetada conforme a Figura 3 os seus principais elementos são a Rede de Acesso (5G RAN), esta camada de rede sem fio é composta por estações rádio base *gNodeBs* para o 5G, oferecendo maior largura de banda, menor latência e maior eficiência espectral em comparação com o 4G/LTE.

O Núcleo de Rede (5G Core), nele consiste em várias funções de rede distribuídas em diferentes locais, como centros de dados, para fornecer serviços e funcionalidades avançadas.

Alguns dos principais componentes do núcleo são, *Access and Mobility Management Function* (AMF) que gerencia o registro e a autenticação de dispositivos na rede. O *Session Management Function* (SMF) que Controla a criação, modificação e exclusão de sessões de comunicação. O *User Plane Function* (UPF) que encaminha o tráfego de dados entre os dispositivos e os serviços de rede. O *Authentication Server Function* (AUSF) que realiza a autenticação dos usuários e dispositivos. E por fim o *Unified Data Management* (UDM) que gerencia a identidade e os perfis dos usuários ou dispositivos.

A Rede de Acesso é responsável por conectar a *Open Radio Access Network* (RAN) ao núcleo da rede. A Rede de Transporte que fornece toda infraestrutura de rede para o tráfego entre os diferentes elementos da rede. A Rede de Borda (5G MEC/Fog/Edge) que cria uma extensão da rede central e fornece recursos de computação e armazenamento mais próximos dos dispositivos finais. Isto auxilia a reduzir a latência e a melhorar o desempenho de algumas aplicações. Orquestração e Virtualização que possui a capacidade de gerenciar e orquestrar recursos de rede de forma dinâmica e automatizada, *Network Functions Virtualization* (NFV), e a *Software Defined Networks* (SDN).

Figura 3 – Arquitetura 5G

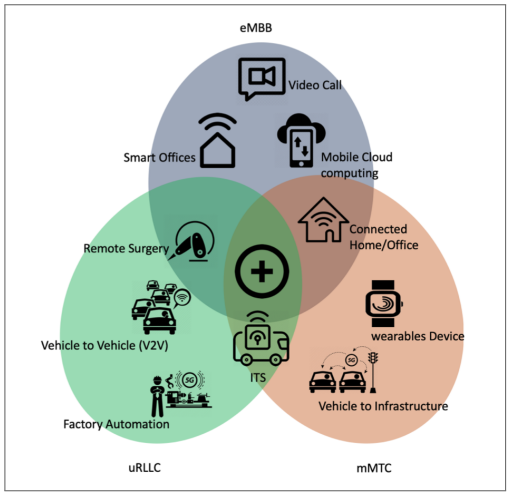


Fonte: Adaptada OAI (2023)

Segundo o 3GPP e a União Internacional de Telecomunicações (ITU), a próxima geração de redes móveis, conhecida como 5G, será baseada na tecnologia de novo sistema de rádio (NR) o que promete melhorar significativamente o desempenho em comparação com o LTE, especialmente em termos de eficiência espectral, taxa de dados, densidade de conexão, latência e consumo de energia de terminais e redes, aplicável a diversos cenários como mMTC, eMBB e uRLLC (BARZEGAR et al., 2020).

O plano do 3GPP prevê que as futuras Redes de Comunicação Sem Fio, transportem pelo menos dez vezes mais tráfego, mantendo latência ultra baixa e alta confiabilidade. A arquitetura do sistema 5G é baseada em serviços, conforme ilustrado na Figura 4. Os requisitos para a rede celular 5G foram finalizados pela ITU e 3GPP, destacando três principais casos de uso, conforme apresentados na Figura 4. A eMBB, objetiva a ampliação das capacidades do MBB convencional, melhorando as taxas de dados, capacidade e cobertura, com um aumento de 10 a 100 vezes em relação ao 4G. As uRLLC são destinadas a aplicações que requerem alta confiabilidade e baixa latência, como internet tática, redes inteligentes, cirurgias remotas e sistemas de transporte inteligente, com latências em torno de 1 ms. O mMTC, são necessários para cenários emergentes de IoT 5G, envolvendo um grande número de dispositivos conectados, como sensores independentes, entre outras tecnologias emergentes, com largura de banda até 1000 vezes maior por unidade de área.

Figura 4 – Requisitos do 5G

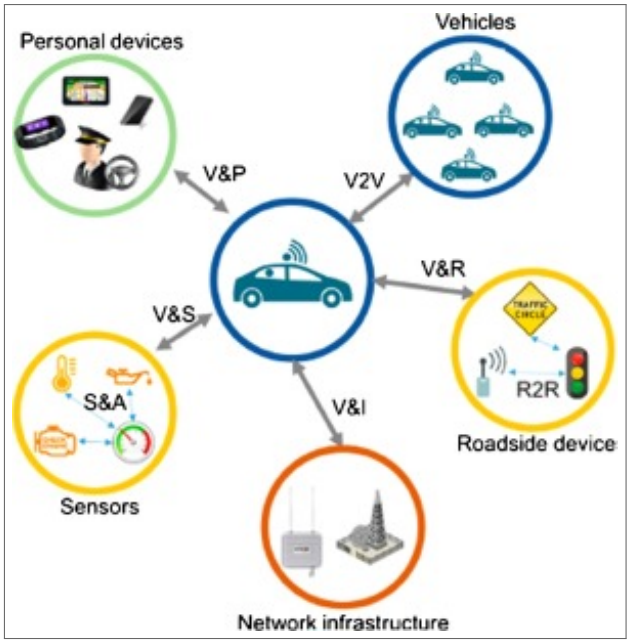


Fonte: (BARZEGAR et al., 2020)

2.2 INTERNET DE VEÍCULOS

Na literatura, a Internet dos Veículos (IoV) é descrita de diversas formas (YANG et al., 2014)(ALAM; SADDIK, 2015)(ALAM; SADDIK, 2018). Com base nessas definições, conceituamos a Internet dos Veículos (IoV) como um ambiente computacional que busca a interconexão de veículos e dispositivos heterogêneos com diversas aplicações por meio de redes de comunicação móveis, abrangendo diferentes ambientes, dispositivos, tecnologias e serviços. Conforme apresentado na Figura 5.

Figura 5 – Formas de Comunicação IoV



Fonte: (CASTILLO; IBAñEZ, 2018a)

Apesar da denominação Internet de Veículos, este sistema de comunicação visa incorporar todos os elementos que compõem um ecossistema de transporte e trânsito metropolitano, rodoviário ou rural. Tais comunicações integram dispositivos portáteis, unidades de beira de estrada (RSUs), dispositivos de trânsito e veículos multimodais nas vias públicas ou privadas, criando assim um ecossistema de comunicação integrado e complexo. A IoV pode ser conceituada como um superconjunto da Rede Ad-hoc Veicular (VANET), pois ela mantém algumas características ampliando, melhorando e integrando novos serviços e aplicações das VANETs (ALAM; SADDIK, 2018).

2.3 COMPUTAÇÃO DE REDES EM CAMADAS

2.3.1 Computação em Nuvem

Gigantes multinacionais da computação, como Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform e IBM Cloud, promovem a adoção do modelo de Cloud Computing (CC), onde as análises, as tomadas de decisão e todos os cálculos ocorrem centralmente em grandes e distantes centro de dados. O modelo CC inclui três serviços principais:

1. Plataforma como Serviço (PaaS): Oferece suporte para desenvolver, executar e gerenciar aplicativos baseados na web sem a necessidade de construir e manter infraestruturas.

Um exemplo é a Apprenda, que oferece PaaS de nuvem privada para .NET e Java.

2. Infraestrutura como Serviço (IaaS): Fornece infraestrutura de computação para operações empresariais, gerenciando recursos de computação, armazenamento e rede. Exemplos incluem AWS, Cisco Metacloud, Microsoft Azure, Google Compute Engine e Joyent.

3. Software como Serviço (SaaS): Fornece aplicativos prontos para uso que são acessíveis pela internet, eliminando a necessidade de instalação e manutenção de software localmente. Exemplos de SaaS incluem Google Workspace, Microsoft Office 365 e Salesforce.

No entanto, esse modelo centralizado apresenta desvantagens, como o aumento do tráfego e do congestionamento na rede, especialmente devido à comunicação do tipo máquina (MTC) na IoT, e a maior latência para dados e aplicações sensíveis a atrasos, dada a distância entre dispositivos finais da IoT e os data centers. Além disso, apenas grandes empresas podem criar uma infraestrutura em nuvem, resultando em um monopólio de mercado (OMONIWA et al., 2019).

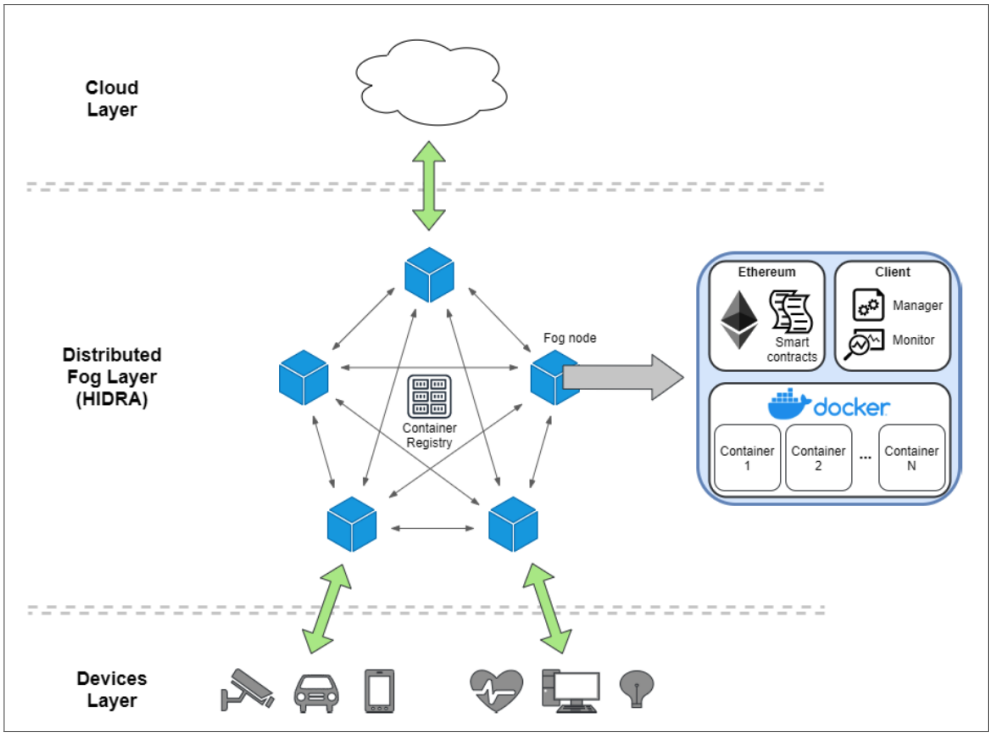
2.3.2 Computação em Nevoeiro

O conceito de *Fog Computing* foi introduzido em 2012 por Pan e McElhannon, da Cisco Inc. O paradigma da computação em neblina é crucial para apoiar a próxima geração da Internet, incluindo aplicações de IoT, como saúde inteligente, cidades, agricultura e indústria, assim como a Internet de Veículos (IoV). A principal ideia da computação em neblina é fornecer suporte computacional próximo à fonte dos dados, com servidores *Fog* no ambiente local, reduzindo o congestionamento da rede. Esta arquitetura visa estender a capacidade de processamento e armazenamento de dados mais próxima dos usuários finais, complementando a computação em nuvem.

O gerenciamento de recursos no ambiente *fog*, inclui operações administrativas como implantação, virtualização, monitoramento, balanceamento de carga, escalonamento automático e provisionamento dinâmico de recursos, garantindo a confiabilidade e a disponibilidade dos serviços. A *Fog* pode implantar nós intermediários em estações base de redes celulares, fornecendo recursos de computação em nuvem dentro da Rede RAN.

As arquiteturas de gerenciamento de recursos em sistemas de neblina podem ser centralizadas ou distribuídas, dependendo de como os recursos são controlados. No controle centralizado,

Figura 6 – Modelo de Computação em Nevoeiro



Fonte: (NúñEZ-GóMEZ; CAMINERO; CARRIóN, 2021)

um único controlador decide sobre o uso dos recursos de ponta, com a maioria das pesquisas focadas em arquiteturas centralizadas, com orquestradores de contêineres (Kubernetes, Docker Swarm, Mesos). Em contraste, em uma arquitetura distribuída, a tomada de decisão é distribuída entre os nós de neblina. Arquiteturas baseadas em *Blockchain* podem ser utilizadas como suporte para implementar controle distribuído em sistemas de computação em neblina (NúñEZ-GóMEZ; CAMINERO; CARRIóN, 2021).

2.3.3 Computação na Borda

A Edge Computing (EC) visa superar desafios ao utilizar as capacidades de armazenamento e processamento de dispositivos IoT conectados à Internet. Isto reduz a carga nos data centers e a latência, permitindo tratamento em tempo real de algumas solicitações localmente. Os dispositivos Edge, devido à sua abundância e distribuição geográfica, também suportam os atributos da mobilidade. A técnica de *offloading* na Edge Computing transfere códigos que consomem muitos recursos para os dispositivos finais, melhorando a eficiência computacional da rede das aplicações e eficiência energética do sistema (DOLUI; DATTA, 2017).

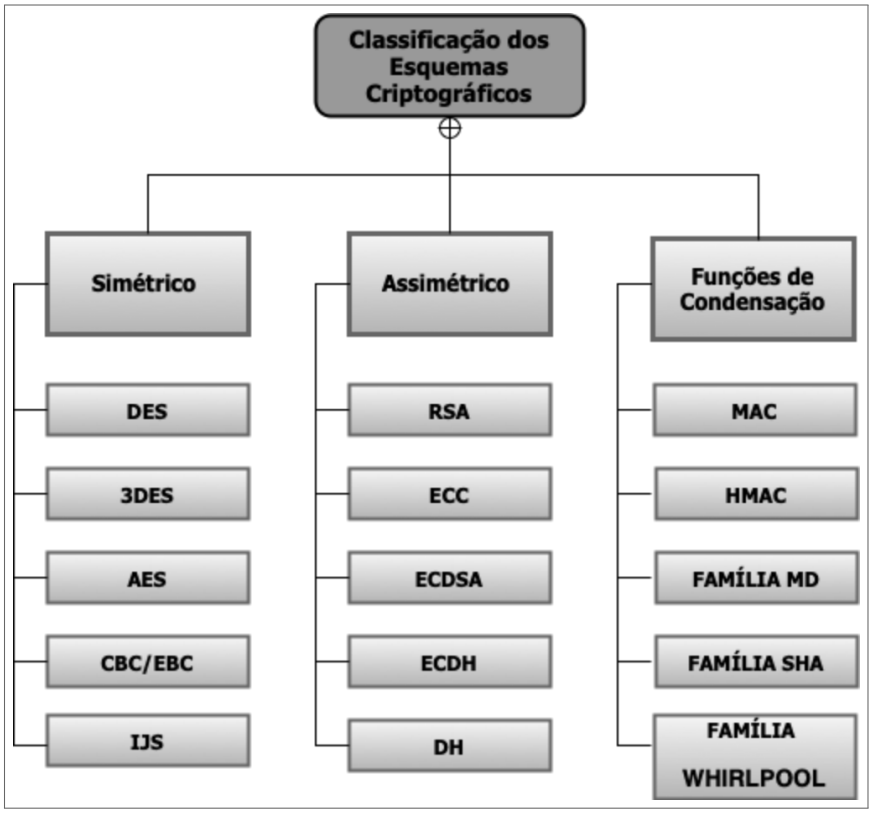
Ao observarmos a Figura 6, os dispositivos presentes na terceira e última camada (*Devices*

Layer) podem colaborar com o processamento e armazenamento de dados, atuando como nós de borda (*edge*) em uma arquitetura de rede em camadas.

2.4 ESQUEMAS DE SUPORTE À SEGURANÇA

Um fator de suma importância para definição dos serviços de autenticação são os esquemas cifragem para proteção. A Figura 7 apresenta os esquemas criptográficos classificados conforme as suas categorias e funções, onde no primeiro grupo "Simétrico"temos como principais elementos a privacidade e confidencialidade, no segundo grupo "Assimétrico"a autenticidade e o não repúdio e no grupo final "Funções de Condensação", responsáveis pela verificação da integridade e conformidade dos dados transmitidos ou armazenados.

Figura 7 – Taxonomia dos Esquemas Criptográficos



Fonte: O autor (2020)

Em um esquema criptográfico simétrico temos que a chave utilizada para realizar a criptografia é sempre igual à chave aplicada para descriptografia, ou seja, a chave é única. Então, quando Alice e Bob [1] necessitam utilizar um esquema simétrico, eles precisam trocar esta

[1] Alice e Bob são arquétipos que representam entidades envolvidas nas transações de criptografia, <https://www.unicamp.br/unicamp/unicamp_hoje/jornalPDF/ju255pag03.pdf>.

chave secreta idêntica antes de iniciar o processo de criptografia dos dados. Este processo do envio da chave única em canais inseguros é um problema a ser tratado nos sistemas de comunicação em redes não confiáveis (BUCHMANN, 2004).

Os esquemas assimétricos de criptografia utilizam chaves distintas para criptografar, descriptografar os dados. Em tais esquemas, uma das chaves pode se tornar pública e a outra chave privada a depender do contexto de cada aplicação. Logo, caso Alice deseje obter mensagens seguras com Bob, ela pode publicar na rede uma de suas chaves, em um fiel depositário denominado *Certificate Authority* (CA), mantendo a outra chave do par de forma secreta e sob seu controle. Desta maneira, Bob poderá utilizar a chave publicada na rede para criptografar ou descriptografar mensagens enviadas, ou recebidas de Alice. Desde modo, apenas a chave secreta ou privada de Alice poderá completar o processo do esquema de proteção assimétricos iniciado por Bob (BUCHMANN, 2004).

Por fim, os esquemas de *hash* ou funções de condensação, também conhecidas como funções de compactação ou unidirecionais. São funções que estruturam cadeias de dados longos em cadeias de dados de comprimento fixo ou truncado. Funções de *hash* são utilizadas para verificação de cadeias de dados diversos, sejam, login, senha ou códigos de sistemas armazenadas em repositórios (BUCHMANN, 2004). Considerando que os esquemas baseados em funções de *hash* são mais eficientes e rápidos quando comparados aos esquemas criptográficos bidirecionais (simétricos ou assimétricos) (LIANHAI et al., 2019).

2.5 MÉTODOS DE AUTENTICAÇÃO

Um método de autenticação com arquitetura centralizada ou distribuída é definido como sendo um terceiro ente altamente confiável, capaz e responsável por registrar os componentes e validar suas operações em um ambiente de rede (AL-SHAREEDA et al., 2021). Por padrão um método de autenticação é conectado aos elementos de comunicação via redes confiáveis e normalmente cabeadas, segmentadas e com níveis de segurança e proteção específicos do serviço. Opera como uma entidade de verificação e autorização cartorial, onde os nós usuários que pretendem ingressar no ambiente deverão ser registrados e permissionados conforme suas características e perfis. Na Internet de Veículos, o método de autenticação é responsável por permitir ou restringir as comunicações entre os veículos, dispositivos e a infraestrutura complexa do ambiente.

2.5.1 Fatores de Impacto dos Métodos de Autenticação

Um fator extremamente impactante é o modo de operação de cada método de autenticação centralizado e descentralizado para cada ambiente de rede utilizado. A maneira como o processo de autenticação opera em diversas arquiteturas e tecnologias de comunicação influenciam diretamente nas aplicações e serviços de cada ambiente que dependem da eficácia e do desempenho do processo de validação para poderem atuar corretamente.

No modo de operação centralizada, a Autoridade de Confiança atua como ponto único dos serviços de autenticação no ambiente da IoV. Os veículos realizam suas solicitações de participação na rede para *Trusted Authority* (TA) e caso sejam autorizados recebem suas chaves assimétricas e simétricas de validação conforme seus perfis. A capacidade e desempenho da autoridade de confiança é relativa ao poder de processamento da máquina servidora e da própria concepção do serviço, assim como de fatores relacionados a largura de banda, número de requisições e usuários cadastrados. Tais parâmetros citados são pré-estabelecidos conforme o conhecimento sobre o ambiente, logo, devemos considerar que as IoVs são ambientes altamente dinâmicos, conseguir prever corretamente suas capacidades e compreender o ambiente para definir os requisitos não é uma tarefa trivial, pois fatores externos sejam típicos ou atípicos podem influenciar em determinado momento todo o processo e reconhecimento dos parâmetros (REIDT, 2007a).

Com relação à solução descentralizada, a Cadeia de Blocos na IoV, atua fornecendo os serviços de autenticação e gerência das chaves de forma distribuída entre os elementos que compõe este ecossistema. Onde os veículos e os dispositivos de comunicação compartilham as responsabilidades e atividades desta gestão. Há também o compartilhamento das decisões e autorizações sobre as requisições e transações na rede, assim o consenso ocorre por meio de um algoritmo de decisão pré-definido, sendo as votações e demais atividades realizadas por determinados participantes selecionados dentre os membros a depender de cada regra consenso. No caso da *Block Chain* (BC) o conhecimento prévio sobre o ambiente para ajustar as capacidades não se faz necessário, pois à medida que aumentam o número de requisitantes, novos participantes são alocados para auxiliarem nas demandas das atividades. O desafio com a tecnologia da *blockchain* está na quantidade de requisições trocadas entre os participantes e no tempo necessário para que essas solicitações sejam avaliadas e divulgadas pelos nós decisores, isto conforme o algoritmo de consenso aplicado (CASTILLO; IBAñEZ, 2018b).

2.5.2 Autoridade de Confiança - *Trusted Authority* (TA)

O Centro Conjunto de Investigação da Comissão Europeia define confiança como "a propriedade de uma relação comercial, de tal forma que a confiança pode ser depositada nos parceiros comerciais e nas transações comerciais desenvolvidas com eles". Nesta perspectiva de confiança, surge a necessidade do estabelecimento de serviços e métodos para viabilizar transações realizadas eletronicamente. As questões relacionadas com a determinação e confiabilidade dos pares, como na confidencialidade de informações, integridade de dados valiosos, prevenção de cópia e uso não autorizados de produções digitais, disponibilidade de informações críticas e confiabilidade de serviços e aplicações são fundamentais para viabilizar os sistemas de comunicação na rede mundial de computadores (GRANDISON; SLOMAN, 2000).

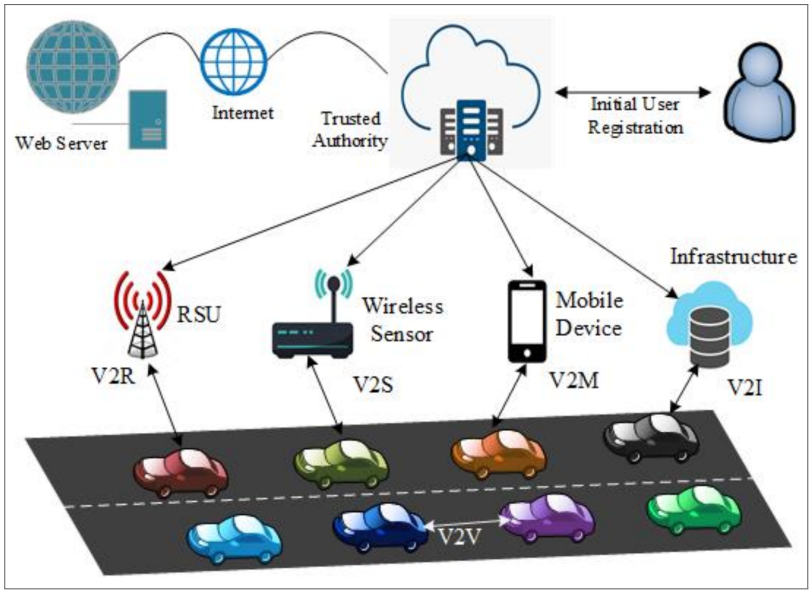
As TA, surgem da necessidade de validação nas interações sensíveis que sejam realizadas entre estranhos nos ambientes das redes de comunicação. A autoridade de confiança visa a substituição dos processos de autenticação por meio de credenciais ou documentos físicos para estabelecer a veracidade entre as transações dos membros. A abordagem de uma autoridade sistematizada é baseada na minimização da intervenção humana no processo de autorização e aprovação das atividades digitais realizadas com base em perfis de utilização de autorização pré-definidos (WINSLETT, 2003).

Os seus serviços da TA formam a base para muitas aplicações e demais serviços ofertados na rede com segurança. A Figura 8 apresenta a atuação da aplicação nas requisições de autenticação de maneira centralizada com a execução no núcleo da rede IoV (NISHU; RAVIKANTI et al., 2020).

A TA pode conter diversos serviços que envolvem desde operações de cadastros de usuários e dispositivos, autenticação, definição de perfis e validação de transações, até as funções de gerência de chaves na rede. As funções relativas à gerência das chaves únicas, públicas ou privadas podem ser terceirizadas. Esta modalidade de contratação de um provedor externo, CA ocorre por motivos legais e às vezes por exigências específicas de ambientes de negócios financeiros, comerciais ou governamentais.

A aplicação de verificação de confiança centralizada representa um desafio significativo tanto do ponto de vista da eficiência quanto dos requisitos de segurança da informação, especialmente para os ambientes (IoT/IoV) distintos das redes tradicionais. (REIDT, 2007b). A utilização de estratégias para aperfeiçoar o desempenho e a segurança dos serviços da TA vem sendo objeto de diferentes estudos, principalmente aqueles que avaliam a execução dos

Figura 8 – Autoridade de Confiança no Ambiente IoV



Fonte: (GUPTA, 2020)

serviços de autenticação e gerência de chaves na borda da rede o que melhoraria em tese problemas de comunicação e sobrecargas na rede (QURESHI GWANGGIL JEON, 2021).

Existem algumas pesquisas no sentido da distribuição de alguns serviços provenientes da TA na rede (REIDT, 2007b; REIDT, 2007a; BALFE STEPHEN WOLTHUSEN, 2009; QURESHI GWANGGIL JEON, 2021), tais estudos, baseiam-se em parâmetros de distribuição de carga, localização, tempo de resposta, números de saltos e até em esquemas federados de colaboração de recursos. Como ocorrem nos bancos de dados hierárquicos (AD/LDAP). As principais diferenças entre as TA's e as BC's estão em suas formas de atuação e validação monocrática e democrática e suas arquiteturas de aplicação centralizadas e descentralizadas.

2.5.3 Cadeias de Bloco - *Blockchain* (BC)

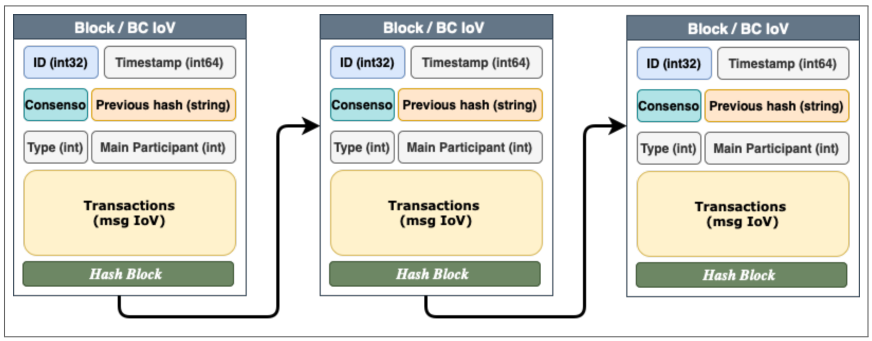
As *Blockchains* (BC) são tecnologias em crescimento e aperfeiçoamento, são originadas da concepção dos livros razão escriturários contábeis ou cartorários (SILBERSCHATZ HENRY F. KORTH, 2011). Logo, uma cadeia de bloco pode ser categorizada como uma *Distributed Ledger Technology* (DLT) ou livro-razão distribuído e gerenciado por um mecanismo participativo de regras com base na decisão por determinado modelo de consenso (SUNYAEV, 2020).

Esta solução foi arquitetada inicialmente por Satoshi Nakamoto[2] para definir um sistema

[2] https://bitcoin.org/bitcoin.pdf

de autenticação e validação de transações realizadas na rede mundial com a criptomoeda *Bitcoin*. As *Blockchains* tem como preceito a inviolabilidade dos registros digitais distribuídos e a primazia das decisões comunitárias, em geral, sem a presença de uma autoridade ou repositório central de confiança. *Blockchain* é uma tecnologia descentralizada, um paradigma de computação distribuída que usa estruturas de blocos encadeados por meio do *hash link* para validar e armazenar e compartilhar dados, algoritmos de consenso e os contratos inteligentes.

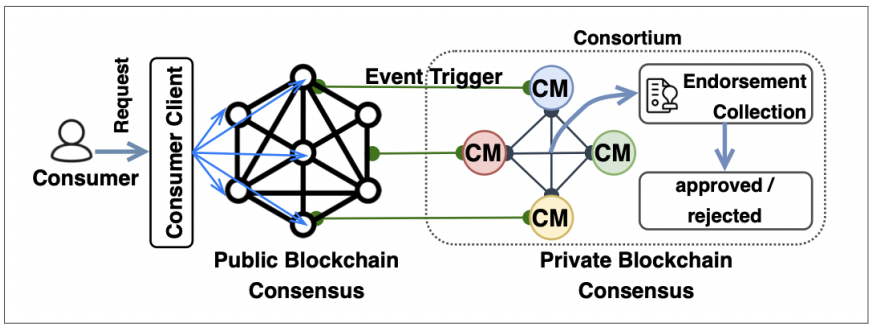
Figura 9 – Estrutura dos Blocos na *Blockchain* Adpatado



Fonte: (ZHAO, 2020)

Os blocos, Figura 9, são compostos por transações submetidas a rede que são autenticadas e validadas de forma compartilhada, ou seja, sem a necessidade de um ente confiável privado ou governamental destinado para este fim (ZHAO, 2020). Atualmente, existem três modos para implementação de um sistema de autenticação distribuída empregando a tecnologia da *Blockchain*, conforme apresenta a Figura 10, pública, privada ou de consórcio (GHOSH TANAY BHARTIA, 2021).

Figura 10 – Modos de Implementação da *Blockchain*



Fonte: (GHOSH TANAY BHARTIA, 2021)

As *Blockchains* consideradas públicas são aquelas onde qualquer nó pode se habilitar a participar atuando nas diversas atividades dos processos de divulgação, mineração, consenso, validação e registro das transações na rede. As redes do tipo consórcio ou federadas são

manipuladas por um conjunto de entidades que possuem objetivos em comum e definem critérios de parceria para utilização compartilhada do sistema distribuído e recursos no referido ambiente de comunicação (GHOSH TANAY BHARTIA, 2021). Neste modo, os nós que desejam participar do ambiente necessitam de algum tipo de autorização e a sua definição do perfil é realizada por um agente atuador que possui autorização especial destinada a algumas ações de gerência na BC. Algoritmos de consenso e contratos inteligentes são mecanismos para manutenção deste modo de aplicação.

Finalmente, as *Blockchains* privadas, estas são um subtipo específico das BCs de consórcio, pois são manipuladas por uma única entidade, empresa ou corporação. As regras, algoritmos de consenso e os contratos inteligentes podem possuir funções e vereditos unilaterais (ZIBIN; SHAOAN; OTHRES, 2020). Em qualquer modo de implementação deste tipo de Cadeias de Blocos (BC) possuem desempenho, eficiência e disponibilidade que vão depender da quantidade de nós participantes, número de transações, método de consenso, gatilhos dos contratos inteligentes, validação e o tamanho do livro razão distribuído conforme a capacidade de recursos do referido ambiente privado.

Um ponto relevante a ser considerado em uma *Blockchain* é sua forma de obtenção da aprovação em grupo para tomadas de decisão de validação dos nós, usuários e transações. Este método democrático de julgamento compartilhado recebe o nome de "técnica de consenso", sendo ele um elemento primordial de distinção das demais tecnologias que realizam tomadas de decisão em rede (JAVED KIRIL ANTEVSKI; BERNARDOS, 2022). A Tabela 2, apresenta um resumo dos principais mecanismos de consenso utilizados nas *Blockchains* bem como os parâmetros de gerência, taxa de transmissão, consumo energético, escalabilidade do sistema e número de transações por segundo.

O consenso **PoW** (*Proof of Stake*) é reconhecido como um método seguro e eficaz para prevenir fraudes e garantir a segurança dos cripto-ativos. No entanto, seu alto consumo energético provoca problemas de escalabilidade e desempenho. O primeiro nó a resolver o problema recebe uma recompensa percentual da criptomoeda, chamada *Satoshi* (NAKAMOTO, 2009).

Já no consenso **PoS**, a validação e criação de blocos são baseadas na seleção dos nós segundo as características específicas do nó. Diferentemente do PoW, onde problemas matemáticos são resolvidos, no PoS os nós validadores são escolhidos com base na sua participação na rede. Quanto maior a participação de um nó, maior a chance de ser selecionado para validar transações e receber recompensas.

Outra importante aplicação da tecnologia BC a ser destacada são as funções dos Con-

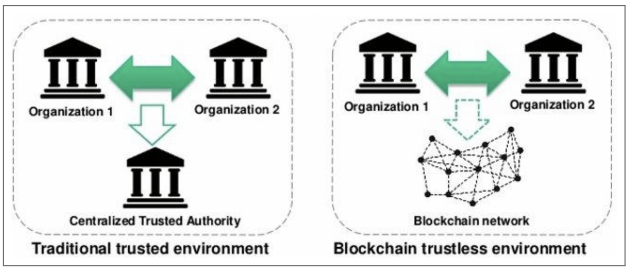
Tabela 2 – Tipos de Consensos na *Blockchain*

CONSENSO	DESCRIÇÃO	GERÊNCIA	TAXA TRANS	CONS ENERG	ESCALAB	TRANS/SEG
PoW	**Proof of Work**	Pública	Baixa	Alto	Forte	<100
PoS	**Proof of Stake**	Parcial	Médio	Médio	Forte	<1000
DPoS	Delegated Proof of Stake	Parcial	Alto	Médio	Forte	<1000
PBFT	Practical Byzantine Fault Tolerance	Privada	Alto	Baixo	Fraca	<2000
RAFT	Consenso Alter Fault Tolerance	Privada	Médio	Médio	Fraca	<10000
PoET	Proof of elapsed time	Privada	Médio	Baixo	Forte	N/A
PoQ	Proof of Quality	Parcial	Alto	Alto	Fraca	<1000
FRChain	Fault Resilient Chain	Pública	Médio	Médio	Forte	N/A
PoA	Proof of Authority	Consórcio	Médio	Baixo	Média	N/A
PoA *	Proof of Activity	Privado	Alto	Baixo	Forte	<1000
PoB	Proof-of-Burn	Público	Médio	Médio	Fraco	N/A
PoC	Proof of Capacity	Público	Baixo	Baixo	Forte	N/A

Fonte: (Adaptado de JAVED KIRIL ANTEVSKI; BERNARDOS, 2022)

tratos Inteligentes, eles representam códigos auto-executáveis determinísticos que funcionam conforme condições transacionais especificas e pré-definidas (ZIBIN; SHAOAN; OTHRES, 2020). Os contratos inteligentes são armazenados na *Blockchain* em um endereço específico designado para estas funções, sua execução automática se dá caso ocorra um evento que realize determinada chamada, conhecido como "gatilho", ao endereço onde se encontra um determinado contrato inteligente. Vale ressaltar que a execução da função de um contrato inteligente é atômica, ou seja, deve ser executado completamente (JAVED KIRIL ANTEVSKI; BERNARDOS, 2022).

Figura 11 – *Trust Authority x Blockchain*



Fonte: (SEUIL, 2018)

A Figura 11, apresenta as duas soluções discutidas neste capítulo. No ambiente tradicional os membros participantes necessitam da TA central, já no ambiente da BC os entes são validados através dos mecanismos de confiança colaborativos na rede.

A tecnologia *blockchain* possui inúmeras aplicações no campo da computação, sendo mais

frequentemente associada às criptomoedas. No entanto, ela pode ser utilizada para resolver uma variedade de problemas reais, incluindo tomadas de decisão democráticas, validação de pares, descentralização de serviços e a garantia da imutabilidade dos dados. Esta pesquisa teve como objetivo analisar a aplicação da tecnologia *Blockchain* exclusivamente como um mecanismo de autenticação e armazenamento das chaves de proteção em ambientes de redes veiculares.

2.6 SISTEMAS SENSÍVEIS AO CONTEXTO

Com o objetivo de otimizar a execução de tarefas computacionais complexas de maneira dinâmica, surgiram os Sistemas Sensíveis ao Contexto. Existem trabalhos na literatura que discutem o que seria a sensibilidade ao contexto de uma aplicação, e quais os fatores a fariam ser considerada sensível a este contexto. Dentre as diversas definições de contexto, a apresentada por (ANIND; GREGORY; DANIEL, 2001), "Contexto é qualquer informação que caracteriza a situação de uma entidade, sendo ela uma pessoa, um lugar ou um objeto considerado relevante para a interação, incluindo a própria aplicação". O contexto abrange geralmente a localização, a identidade, a quantidade e o estado dessas entidades em relação ao ambiente em que estão expostas. Estima-se também, que a definição está diretamente relacionada ao tipo de atividade que o contexto está sendo associado, sendo assim, contexto seria a possibilidade de determinar em um ambiente a relevância entre diversas variáveis envolvidas e assim realizar ações para otimizar as necessidades dos usuários no ambiente analisado (BATISTA; SILVA, 2015).

Com o advento da tecnologia e os atributos das redes no padrão 5G, cada vez mais tem chamado a atenção dos pesquisadores e desenvolvedores para a questão da sensibilidade das aplicações ao ambiente e a possibilidade de otimização do processamento e uso dos recursos mais próximo do usuário, melhorando assim o desempenho e a qualidade das aplicações durante sua experiência de utilização na comunicação. O desenvolvimento de tais sistemas e mecanismos sensíveis podem auxiliar no atendimento dos requisitos determinados pelo 3GPP, como redução de latência e o aumento das taxas de dados, fornecendo aos usuários qualidade de serviço (QoS) e qualidade da experiência (QoE). Sabendo que o desempenho da rede pode ser afetado pelos métodos de autenticação e segurança no ambiente dinâmico das IoVs, é necessário buscar mecanismos capazes de lidar com estes cenários, que quando divididos em arquiteturas de camadas, adicionam complexidade ao funcionamento dos serviços e aplicações.

Conforme a Figura 12, são apresentados os diversos ambientes de autenticação que pos-

Figura 12 – *Ambiente de Decisão de Autenticação em Redes IoV*

Fonte: Autor

suem diferentes comportamentos gerando distintas métricas para o funcionamento de uma rede IoV. Frente às variadas situações apresentadas, existem tecnologias e serviços diversificados de apoio que permitem ajustes essenciais em cada contexto, podendo impactar de forma positiva ou negativa no *Quality of Service* (QoS) e *Quality of Experience* (QoE) nas aplicações e serviços das redes veiculares.

2.7 METODOLOGIA PARA OS SISTEMAS DE INTELIGÊNCIA ARTIFICIAL

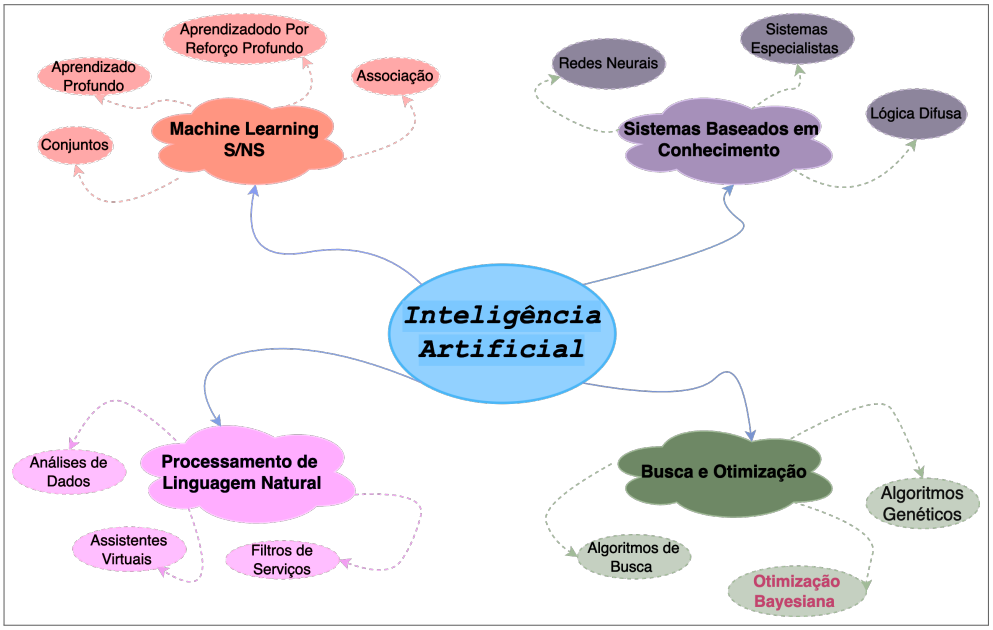
Inteligência Artificial (IA) é o campo da ciência da computação dedicado ao desenvolvimento de sistemas computacionais capazes de realizar tarefas que, tradicionalmente, necessitam da inteligência humana. Isso inclui atividades exaustivas como reconhecimentos de imagens e sons, tomada de decisões e resolução de problemas multifatoriais. A abordagem probabilística, como discutida em (MURPHY, 2022), enfatiza o uso de modelos estatísticos e algoritmos de aprendizado de máquina para lidar com a incerteza e fazer previsões, quantificando as incertezas de maneira flexível e aplicável a uma variedade de cenários. Isso particularmente é utilizado para situações que lidam com as incertezas como forma de fornecer suporte a tomadas decisões com base em dados incompletos ou incertos.

(MURPHY, 2022) discute em sua publicação a vantagem da abordagem de otimização bayesiana, sua capacidade para lidar com as incertezas em eventos únicos, utilizando das notações matemáticas para facilitar a compreensão das funções de contexto no campo do aprimoramento. Termos cruciais, como "função de perda", "função objetivo"e "*solver*", são claramente definidos para garantir uma compreensão precisa. Para maximizar uma função

de pontuação ou recompensa $R(\theta)$, pode-se proceder à minimização de $L(\theta) = -R(\theta)$. O termo "função objetivo"é utilizado de forma genérica para se referir a uma função que se deseja maximizar ou minimizar. Um algoritmo que busca o ótimo de uma função objetivo é frequentemente denominado "*solver*".

Na literatura existem quatro, macros grupos, no que se refere a inteligência artificial (BACK, 1996), conforme apresentado na Figura 13:

Figura 13 – Mapa IA - Otimização Bayesiana



Fonte: Adaptado [3]

2.8 ESTRATÉGIAS DE OTIMIZAÇÃO

As técnicas tradicionais de otimização de hiperparâmetros costumam seguir uma abordagem empírica, baseada em tentativa e erro. Nesta abordagem, diferentes combinações de parâmetros são ajustadas manualmente ou mediante processos repetitivos para identificar quais valores proporcionam o melhor desempenho de um modelo de aprendizado de máquina ou sistema computacional. Segundo Kumar (2022), existem três técnicas tradicionais que seguem essa estratégia:

1. Busca Manual: Nesta técnica, o desenvolvedor testa diferentes combinações de hiperparâmetros para o modelo de aprendizado de máquina que seleciona aquele que apresenta

[3] https://www.datageeks.com.br/machine-learning

o melhor desempenho. Nesta pesquisa o desempenho está relacionado a hiperparâmetros dinâmicos e não correlacionados diretamente com os valores obtidos.

2. Grid Search: O Grid Search é uma versão automatizada da busca manual para otimização de hiperparâmetros. Embora automatize o processo, não é eficiente em termos de custo computacional, pois consome muito tempo para realizar a otimização o que altera o foco desta pesquisa.

3. Random Search: Enquanto o Grid Search testa todas as combinações possíveis de hiperparâmetros, aumentando o tempo de computação, o Random Search treina modelos com combinações de hiperparâmetros escolhidas aleatoriamente. Esta técnica de combinações dos hiperparâmetros para as soluções de autenticação não era o objetivo da pesquisa.

4. **Otimização Bayesiana (não tradicional)**: provê uma estrutura elegante para resolver problemas de maximização ou minimização de uma função objetivo para decisões quando a derivada da função é desconhecida (YE, 2022). Segundo experimentos realizados por (KRAUS, 2019), que compararam diversas técnicas de busca e otimização similares, a técnica bayesiana apresentou melhores acurácias nas análises dos hiperparâmetros para o aprimoramento de sistemas de decisão.

Com relação ao algoritmo Otimização Bayesiana (OB) utilizado nesta tese, que será detalhado na subseção 2.8.1, considerou os recursos e infraestrutura do ambiente IoV e das tecnologias de comunicação em redes 5G. Percebendo que o desempenho de alguns parâmetros podem ser afetados devido aos métodos de autenticação e os métodos de segurança implementados no ambiente veicular. Sendo assim, esta pesquisa visa mensurar os desempenhos e impactos de cada solução na realização da autenticação e verificação da integridade e veracidade das mensagens de dados, a fim de, determinar qual método de autenticação e validação utilizará melhor os recursos disponíveis, garantindo os requisitos necessários ao funcionamento dos ambientes analisados (MOHAMMADI; SHEIKHOLESLAM, 2023).

2.8.1 Otimização Bayesiana

O algoritmo OB é uma técnica da área da *Artificial intelligence* (IA), motivada pela capacidade de seleção de multiparâmetros e configurações de seus valores. São amplamente

aplicados para busca de soluções em problemas de otimização e simulação (YE, 2022). Nesta subseção, exploraremos os conceitos fundamentais da OB, seus componentes essenciais e suas aplicações.

A Otimização Bayesiana é uma técnica poderosa e eficiente, projetada para otimizar funções objetivo que são caras ou demoradas para serem realizadas através da avaliação bruta. Ao contrário dos métodos tradicionais de otimização, que podem exigir inúmeras avaliações diretas da função objetivo, a Otimização Bayesiana utiliza um modelo probabilístico para guiar a busca pelo ponto ou valor ótimo, minimizando o número de avaliações necessárias para dado problema (BROWNLEE, 2019).

A otimização de funções consiste em determinar o valor mínimo ou máximo de uma função objetivo. As amostras são selecionadas de um domínio específico e avaliadas com base nessa função objetivo. Os termos utilizados neste processo são:

Amostra: Um exemplo do domínio do problema é representado como um vetor.

Espaço de Busca: O domínio a partir do qual as amostras podem ser selecionadas.

Função Objetivo: Função que recebe uma amostra e retorna o custo, geralmente baseada no erro de predição do modelo construído com hiperparâmetros específicos no conjunto de validação durante a otimização de hiperparâmetros.

Custo: Valor numérico de uma amostra calculado pela função objetivo.

É importante destacar que o teorema de Bayes é uma abordagem utilizada para calcular a probabilidade condicional de um evento, conforme demonstrado na Equação 2.1:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2.1}$$

Na Equação 2.1:

- P(A|B): Representa a probabilidade de o evento A ocorrer dado que o evento B.
- P(B|A): Representa a probabilidade de o evento B ocorrer dado que o evento A ocorreu.
- P(A): Representa a probabilidade de o evento A ocorrer.
- P(B): Representa a probabilidade de o evento B ocorrer.

Segundo (BROWNLEE, 2019), é possível simplificar esse cálculo removendo o valor de normalização P(B), descrevendo a probabilidade condicional como uma quantidade proporcional. Isso é viável, pois o objetivo desta tese, não foi o de calcular uma probabilidade condicional

específica, mas otimizar uma determinada quantidade. Assim, chegamos à relação mostrada na Equação 2.2.

$$P(A|B) = P(B|A) \times P(A) \tag{2.2}$$

Quando trabalhamos com otimização bayesiana, a probabilidade condicional que calculamos é chamada de **posterior**, a probabilidade condicional reversa é conhecida como **verossimilhança**, e a probabilidade marginal é referida como **anterior**. Assim, a Equação 2.2 passa a ser expressa pela Equação 2.3.

$$posterior = verossimilanca \times anterior \tag{2.3}$$

Dessa maneira, podemos considerar amostras específicas $(x_1, x_2, \cdots, x_n)$ e avaliá-las através da função objetivo $f(x_i)$ que retorna um custo para a amostra x_i. As amostras e seus respectivos custos são coletados de forma sequencial, constituindo os nossos dados D.

Com isso, temos $Dx_i, f(x_i), \cdots, x_n, f(x_n)$, usados para definir a nossa probabilidade de causas a efeitos. A função de verossimilhança é definida como a probabilidade de observar os dados dada a função $P(D|f)$, resultando na relação expressa na Equação 2.4. A função de verossimilhança muda conforme coletamos mais dados.

$$P(f|D) = P(D|f) \times P(f) \tag{2.4}$$

A **função posterior** representa tudo o que sabemos sobre a função objetivo, sendo uma aproximação da **função objetivo** e, por isso, pode ser usada para estimar o custo de amostras candidatas diferentes que nós queremos avaliar. Dessa forma, a probabilidade posterior é uma **função objetivo substituta**.

A função substituta nos fornece uma estimativa da função objetivo, que pode ser usada para direcionar a amostragem futura. A amostragem envolve o uso cuidadoso da probabilidade posterior em uma função conhecida como **função de aquisição**.

O Fundamento desta técnica baseia-se no Teorema de Bayes (SAMPSON, 1976) para atualizar o conhecimento sobre a função objetivo à medida que novas observações são feitas. O processo envolve geralmente os seguintes componentes principais:

- Modelo: probabilístico, como um Processo Gaussiano (GP) ou uma Floresta Aleatória, é usado para aproximar a função objetivo com base nos dados observados. Este modelo

é capaz de fornecer uma estimativa da função objetivo e uma medida de incerteza associada a essa estimativa.

- Função: é utilizada para determinar o próximo ponto a ser avaliado pela função objetivo. Ela equilibra a exploração de áreas não exploradas do espaço de busca com a exploração de áreas conhecidas para melhorar a precisão da otimização. Exemplos de funções de aquisição incluem Expected Improvement (EI), Probability of Improvement (PI) e Upper Confidence Bound (UCB).
- Atualização: após cada avaliação da função objetivo, o modelo é atualizado com as novas observações, refinando a estimativa da função objetivo e sua incerteza.

O processo de Otimização Bayesiana pode ser resumido em uma série de etapas iterativas:

1. - Inicialmente, um pequeno número de pontos é amostrado aleatoriamente e a função objetivo é avaliada nesses pontos.
2. Em cada iteração a função de aquisição é maximizada para selecionar o próximo ponto a ser avaliado. A função é atualizado com as novas observações.
3. O processo continua até que um critério de parada seja atendido, como um número máximo de avaliações ou a convergência da função de aquisição.

Figura 14 – Fluxo da Otimização Bayesiana

Fonte: O autor (2023)

A Otimização Bayesiana é amplamente aplicada em diversos domínios onde a avaliação da função objetivo é custosa. Na aprendizagem de máquina, a otimização bayesiana é frequentemente usada para ajustar os hiperparâmetros de modelos complexos, como redes neurais

profundas, para maximizar a precisão de predição ou minimizar a perda. Nas ciências experimentais é utilizada para otimizar os parâmetros de um experimento ou simulação de forma a obter os melhores resultados com o menor número de experimentos. E nos sistemas de controle, pode ser aplicada para otimizar os parâmetros de controladores para alcançar desempenho ideal em sistemas dinâmicos (LIU; WANG; ZHANG, 2021).

A técnica da OB, possui suas vantagens como redução do número de avaliações necessárias da função objetivo, economizando recursos e tempo. Leva em conta a incerteza nas estimativas, proporcionando uma abordagem mais robusta para a otimização destes cenários. Com relação às desvantagens, pode ser computacionalmente intensiva para problemas de alta dimensionalidade e muitos hiperparametros multivalorados. Requer a seleção da função de aquisição, o que pode ser não aplicável a determinados objetivos (RAZALI et al., 2023).

A OB, é mais eficiente em termos de recursos, pois utiliza um modelo probabilístico para guiar a busca, focando nas regiões mais promissoras de dado espaço. Incorpora incertezas sobre o desempenho das diferentes opções e atualiza continuamente suas estimativas à medida que novos experimentos são coletados, refinando suas predições e permitindo decisões mais assertivas. Esta técnica balanceia de forma eficaz a experimentação, atuando para evitar o efeito "ficar preso", em locais ótimos, pois sempre busca encontrar soluções melhores e dinamicamente mais consistentes (MURPHY, 2022).

Sua capacidade de modelar incertezas e balancear exploração a torna especialmente útil em contextos como ajuste de hiperparâmetros, experimentos e otimização de sistemas de controle e decisão. Com avanços contínuos da computação, a Otimização Bayesiana está se tornando cada vez mais acessível e aplicável a uma ampla gama de problemas complexos em diversos campos da ciência.

2.9 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este Capítulo 2, apresentou os principais pressupostos teórico-conceituais para o desenvolvimento desta tese. Considerando que os temas estão em constante evolução e apresentam especificidades inerentes ao ambiente das IoVs, os diversos temas aqui abordados não tiveram pretensão de exaurir todo o conhecimento acerca das tecnologias e soluções inter-relacionadas, buscando focar nas principais características que se relacionam diretamente com o objeto desta tese.

3 CAPÍTULO TRABALHOS RELACIONADOS

Este capítulo apresenta a taxonomia das soluções arquiteturais e os estudos da literatura que abordam métodos de segurança baseados em autoridades de confiança (TA) e blockchain (BC). A revisão da literatura foi realizada entre 2019 e 2023 e utilizou uma abordagem exploratória, combinando buscas manuais em portais de periódicos científicos com ferramentas automatizadas de revisão de literatura. O planejamento da busca incluiu a definição de questões de pesquisa relativas as soluções de autenticação e segurança na Internet dos Veículos (IoV), com foco em métodos centralizados (TA) e distribuídos (BC), assim como a comparação entre ambos.

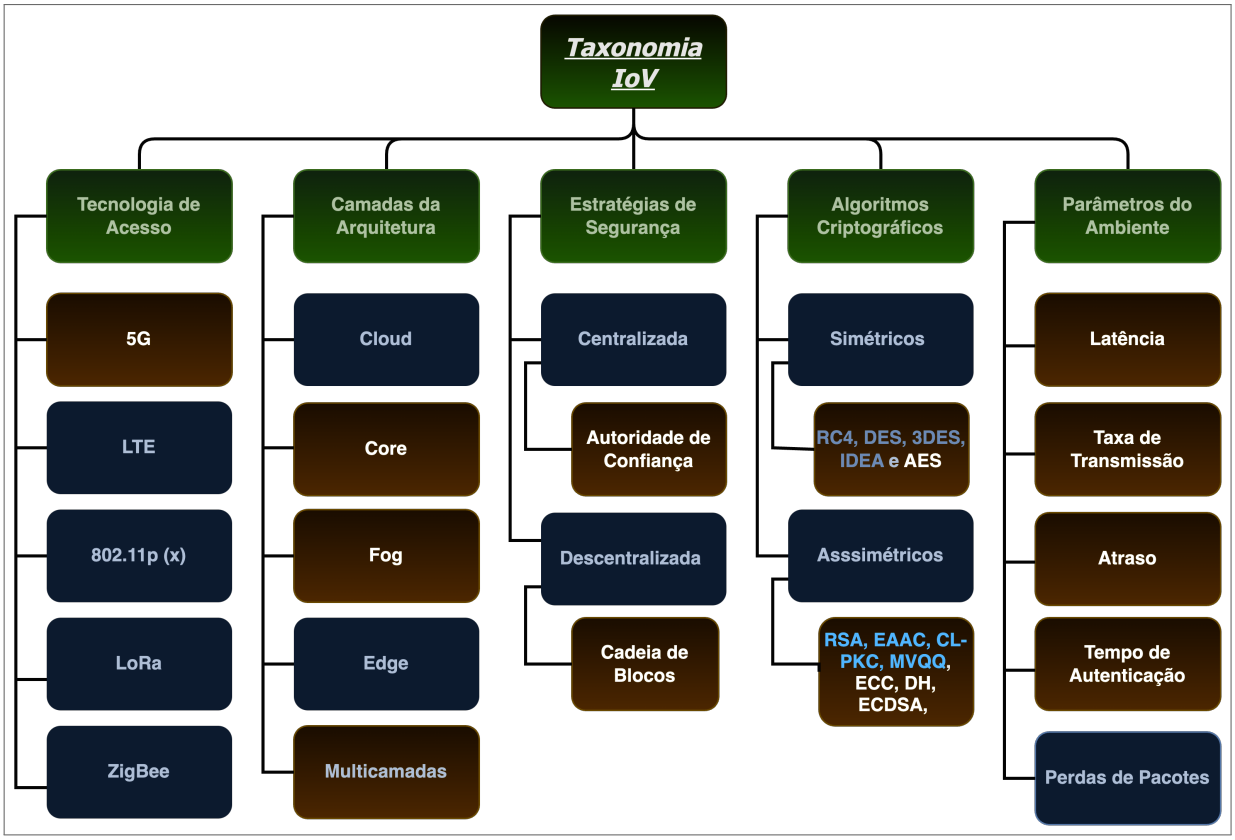
A busca exploratória utilizou strings de pesquisa construídas com termos como "TA-based security in IoV", "Blockchain authentication IoV", "centralized vs. decentralized IoV authentication", e "security methods for vehicular networks". As bases de dados pesquisadas incluíram IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink, e Google Scholar assim como a ferramenta StArt Systematic Review (RS).

A análise dos trabalhos relacionados foi divido em 3 subseções, A seção 3.2, apresenta as pesquisas relacionadas ao método centralizado (TA) no ambiente IoV, já na seção 3.3 exibe os estudos relacionados a aplicação do método distribuído (BC) em redes IoV, e por fim, na seção 3.4 mostra os trabalhos que compararam o funcionamento dos métodos de autenticação centralizados TA e distribuídos BC no ambiente das redes veiculares.

3.1 ARQUITETURAS DE SEGURANÇA PARA IOV

Os elementos constitutivos e relacionados as soluções de tecnológicas e de segurança no ambiente da IoV são apresentados na Figura 15. A taxonomia contém os blocos de construção das arquiteturas pesquisadas que integram os métodos de autenticação na rede *Vehicular Edge Computing* (VEC) para o ambiente IoV. A taxonomia foi estruturada em seis categorias: Tecnologias de Acesso (802.11x, LTE, 5G); Camadas da Arquitetura (*Fog, Edge, Cloud, Muiltlayer*), Estratégias de Segurança (Centralizada e Descentralizada), Algoritmos Criptográficos (Simétricos e Assimétricos) e finalmente os Parâmetros do Ambiente (Latência, Taxas de Transferência, Variação de Atraso, Taxas e Erro e pacotes perdidos).

Figura 15 – Taxonomia dos Elementos e Soluções constitutivas e de Segurança para IoV



Fonte: (QUEIROZ et al., 2020)

3.2 AUTENTICAÇÃO CENTRALIZADA EM IOV

A Tabela 3, apresenta um resumo comparativo entre os trabalhos relacionados com a solução de autenticação baseada no método da Autoridade de Confiança (TA) no ambiente IoV.

Tabela 3 – Resumo das soluções da Autoridade de Confiança em IoV

Artigo	Tecnologia de Acesso	Arquitetura Execução	Estratégia Segurança	Algoritmo Criptográfico
Yang Zhao 2019 (ZHAO et al., 2019)	Definida pela RSU	Nuvem	TA Centralizada	VHPD Proposto
Al-Mutiri Rasha 2018 (RASHA; YUAN, 2018)	802.11p	Nuvem	TA+CA Centralizada	ECDSA Proposto
Prosanta Gope 2019 (GOPE, 2019)	4G/LTE	Nevoeiro	TA Compartilhada	ECIES-AES/CBC-SHA256
Faiza Tahir 2019 (TAHIR; KHALID, 2019)	802.11	Borda	TA Centralizada	HMAC Modificado
Qinlong Huang 2019 (HUANG et al., 2019)	802.11p	Nuvem	TA+CA Centralizada	EAAC/AMDF
Haowen Tan 2020 (TAN, 2020)	5G	Nuvem e Borda	Hibrido TA+BC	ECC/CRT
Lianhai Liu 2019 (LIANHAI et al., 2019)	802.11p	Nuvem	TA Centralizada	DH/AES
Han Zhang 2022 (ZHANG; LAI; CHEN, 2023)	802.11p	Nuvem	TA Centralizada	PKG/ECC
Fayez Alazemi 2023 (ALAZEMI et al., 2023)	4G/LTE	Nuvem	TA Centralizada	NCDC

Fonte: O autor (2023).

O trabalho de (ZHAO et al., 2019) desenvolveu um esquema VHPDT (*Hidden Policy for Privacy preservation Verification and Decryption Tests*) para geração e verificação de chaves

em um ambiente de redes veiculares (VANET), baseadas em TA. A solução pode impactar significativamente nos parâmetros de desempenho da rede, pois o tamanho do texto cifrado com o algoritmo proposto é muito longo. Portanto, um esquema de texto cifrado fixo curto, mas buscando garantir a segurança seria uma solução mais adequada.

O artigo (RASHA; YUAN, 2018) descreveu e apresentou uma implementação de um esquema simples de segurança estruturado em quatro estágios: autenticação, desafio resposta, assinatura e carimbo digital para VANETs. O sistema representa uma modificação do processo de autenticação tradicional já utilizado em outros ambientes de rede com a adição em conjunto de um desafio e um carimbo de validação do processo. A pesquisa não abordou a questão da mobilidade e o impacto dos serviços e aplicações, desconsiderando uma análise de que as Redes Veiculares "Ad Hoc"(VANETs) representam um contexto básico em comparação com o Ambiente da Internet Veicular (IoV).

A criação de um esquema de autenticação na borda para redes veiculares é proposto por (GOPE, 2019). Um esquema em camadas é estabelecido para autenticar dispositivos na borda baseado em TA. O modelo explora a capacidade da camada intermediária para melhorar requisitos da rede como a baixa latência e a perda de dados melhorando o processo de autenticação e validação dos serviços. Neste modelo proposto de (TA) na Borda, persistem os desafios relacionados às atualizações de diversas bases e à sincronização dos dados, resultando ocasionalmente na indisponibilidade de serviços devido ao sobrecarregamento necessário pelo próprio mecanismo de gestão e integração.

Já o estudo de (TAHIR; KHALID, 2019) estabelece um esquema de criptografia híbrido para autenticação em VANET com Autoridade de Confiança. A solução apresentada permite ao sistema definir vários tipos de esquemas criptográficos para o modelo de autenticação mais adequado a rede. A concepção da pesquisa é baseada na melhoria do desempenho restrito a performance dos algorítimos criptográficos em uma rede básica VANET utilizando padrão 802.11p, sendo uma rede comunicação sem fio não plenamente móvel e de curta distância.

A pesquisa de (HUANG et al., 2019) define um esquema que terceiriza os custos dos serviços de segurança tais como: autenticação e criptografia no ambiente IoV com a execução e manutenção dos serviços da TA e CA na nuvem. O esquema busca preservar a privacidade com a redução de custos dos serviços no próprio núcleo do ambiente. O serviço local funciona apenas como uma ponte para os demais servidores externos o que causa uma dependência e latência maior na rede.

Um esquema composto foi elaborado em (TAN, 2020) fazendo uso combinado das chaves

individuais e de grupo com aplicação das soluções da Cadeia de Blocos (BC) e Autoridade de Confiança (TA) para o desenvolvimento de um sistema mais robusto de autenticação e gerência de chaves usando capacidades de recursos das duas soluções na nuvem. Cada tecnologia é utilizada para uma função específica neste sistema híbrido, a TA trata das requisições de autenticação e a BC lida com questões relacionadas aos certificados de validação. Com ambos os métodos localizados na nuvem, os potenciais problemas de desempenho de cada serviço deverão ser devidamente considerados. Demais preocupações sobre o mecanismo de gerência decorrentes da complexidade mencionada estão suscetíveis a tais incidentes relativos à disponibilidade e efetividade dos serviços separados.

O estudo de (LIANHAI et al., 2019) define um sistema para troca de chaves em comboios ou pelotão veicular. Trata-se de um mecanismo de acordo de chaves secretas compartilhadas em grupo na rede. Eles propõem um esquema que utiliza a infraestrutura das RSUs e uma autoridade central confiável (TA) para gerências e controle das autenticações e validações das informações no grupo. A pesquisa (ALAZEMI et al., 2023) propõe a utilização da TA governamental da Arabia Saudita como unidade gestora CA dos certificados dos veículos, onde as análises e validão irão se basear tanto nas chaves secretas quanto na localização e informações dos proprietários no sistema de trânsito do país. Este estudo carece de adequação ao ambiente tradicionalmente público de uma Internet Veicular (IoV), uma vez que se fundamenta em chaves únicas para validação. Isso representa um alto risco do ponto de vista da segurança, pois a divulgação dessas chaves por qualquer motivo pode resultar em graves consequências aos usuários e serviços no ambiente.

3.3 AUTENTICAÇÃO DESCENTRALIZADA EM IOV

Nesta subseção exibimos uma síntese dos principais trabalhos relacionados com a tecnologia Blockchain (BC) suas características e aplicações no ambiente da IoV. A Tabela 4 foi segmentada em quatro abordagens de execução e apresenta um resumo comparativo das pesquisas.

Tabela 4 – Resumo das soluções *Blockchain* em IoV

Artigo	Tipo Blockchain	Camada Blockchain	Consenso	Abordagem Arquitetônica
Jianbin Gao 2019 (GAO et al., 2020)	Consórcio	Borda	PBFT	Camada Borda
Shaoyong Guo 2019 (GUO et al., 2019a)	-	Borda e Nuvem	-	Multicamada Estática
Jiawen Kang 2019 (KANG et al., 2019)	Consórcio	Borda e Neoveiro	PoW	Multicamada Estática
Yueyue Dai 2019 (DAI et al., 2019a)	Privada&Federada	Borda	PoU	Camada Borda
Heena Rathore 2019 (RATHORE et al., 2019)	-	Nevoeiro	TangleCV Policy	Camada Nevoeiro
Youcef Yahiatene 2018 (YAHIATENE et al., 2019)	Pública e Privada	Nevoeiro	DM-CDS	Camada Nevoeiro
XiaoDong Zhang 2018 (ZHANG; LI; CUI, 2018b)	-	Nevoeiro&Borda&Nuvem	-	Multicamada Estática
Chau Qio 2018 (QIU et al., 2018)	Privada&Federada	Borda	PBFT	Camada Borda
Haowen Tan 2019 (TAN; CHUNG, 2020)	Consórcio	Nevoeiro&Borda&Nuvem	-	Multicamada Estática
Vincenzo 2019 (MAIO; URIARTE; BRANDIC, 2019)	-	Nevoeiro&Borda&Nuvem	PoS+PoW	Multicamada Dinâmica
Hong Liu 2018 (LIU; ZHANG; YANG, 2018)	Consórcio	Nevoeiro&Borda	PoEC/PoW + PoF/PoS	Multicamada Estática
Yuchuan Fu 2019 (FU et al., 2020)	Consórcio	Borda	BFT-DPoS	Camada Borda
Y.Wang 2020(WANG et al., 2020)	Consórcio	Nevoeiro&Borda	PoR	Multicamada Estática
Pamela Diana 2022 (OSORIO et al., 2022)	Pública	Borda	-	Multicamada Estática
Shanshan Tu 2023 (TU et al., 2023)	Pública&Federada	Nevoeiro	VBSBC	Camada Nevoeiro
Zheng et al 2023 (ZHENG; WU; LI, 2023)	Privada	Borda/Nuvem	PoS/DPoS	Multicamada Estática

Fonte: O autor (2023)

Com relação ao Tipo de *Blockchain*, Tabela 4, foram categorizadas em Pública, Privada ou Consórcio/Federada. Importante destacar que a maioria dos estudos apresentados optaram pela BC de consórcio a outros tipos. Uma *Blockchain* de consórcio é apenas parcialmente privada, pois neste caso várias empresas compartilham a tecnologia, os métodos e serviços envolvidos e pactuados. As BC's públicas costumam ter baixa escalabilidade, pois são totalmente descentralizadas, cada nó precisa tratar as transações de forma independente. Existem estratégias para superar este problema, como a arquitetura proposta em (RATHORE et al., 2019). As BC's privadas têm seu apelo, pois não possuem a maioria dos problemas de uma BC pública. Uma autoridade central decide quais nós podem entrar, quais nós podem ser mineradores e quais nós devem ter todas as transações. O modo de decisão termina se assemelhando ao de uma TA, pois como uma única empresa gerência os serviços da BC pode definir as regras do consenso e contratos de forma unilateral apenas distribuindo as atividades e tarefas no grupo.

A camada Blockchain, apresentada na Tabela 4, é o espaço onde ocorrem a mineração e o consenso. Esta coluna é significativa para os leitores interessados na tecnologia Blockchain, pois fornece informações sobre as camadas utilizadas nas arquiteturas multicamadas, tanto estática quanto dinâmica. Considerando que nem todo veículo possui a capacidade necessária de armazenamento e processamento, alguns pesquisadores decidiram delegar tarefas da BC para a camada da borda ou diretamente a nuvem. De fato, a maioria dos artigos optou por deixar a BC na borda ou criar uma arquitetura híbrida (*Fog-Edge*) ou (*Fog-Edge-Cloud*). Esta técnica tenta evitar a sobrecarrega computacional nos veículos (YAHIATENE et al., 2019; ZHENG; WU; LI, 2023).

Já a coluna do Algoritmo de Consenso, Tabela 4, especifica qual algoritmo de consenso foi utilizado em cada trabalho pesquisado. É fundamental considerar que o algoritmo de consenso desempenha um papel crítico na eficácia, disponibilidade e desempenho dos métodos que utilizam a tecnologia de blockchain:

- PoW - (*Proof-of-Work*), um dos algoritmos de consenso mais conhecidos, foi implementado primeiramente pela criptomoeda *BitCoin*. Neste algoritmo cada nó compete entre si para resolver um desafio, os nós com maior poder computacional vencem geralmente a competição e conseguem gerar ou validar um novo bloco. O objetivo é dificultar ações maliciosas, pois o custo para atacar uma cadeia, tentando inserir um novo bloco com transações fraudulentas, é muito alto. Isso seria o equivalente a convencer e validar uma falsificação em mais de 50% de toda a rede, que contém e conhece todos os registros públicos das transações passadas. Por outro lado, isso sobrecarrega os nós de mineração, pois para cada consenso há uma necessidade de competição e validação. Em um ambiente IoV, com a tecnologia BC e o método de consenso PoW a melhor escolha é deixar a atividade da mineração fora dos veículos, pois como a dinâmica de participação deles no ambiente é altamente imprevisível "entrada/saída", isto dificultaria as validações e decisões do algoritmo. Os artigos (KANG et al., 2019) e (MAIO; URIARTE; BRANDIC, 2019) utilizaram o PoW combinados com outra estratégia, então a dificuldade do desafio seria menor para certos participantes. Isso aliviaria a carga de consumo de energia, pois o desafio seria resolvido mais rapidamente. O consenso PoR da (WANG et al., 2020) utiliza o modelo de competição do PoW. A proposta (LIU; ZHANG; YANG, 2018) também utilizou uma combinação de algoritmos, PoW e Proof-of-Storage, onde os nós que estão usando mais armazenamento ganham moedas extras a cada rodada de consenso, mas os autores não mudaram a dificuldade do PoW e suas desvantagens permanecem as mesmas.

- PoS - (*Proof-of-Stake*), um algoritmo de consenso muito comum usado pela criptomoeda *Nxtcoin* criada por um desenvolvedor anônimo denominado "BCNext". Este algoritmo veio para tentar resolver a questão inerente ao consumo de energia e desempenho do PoW. Essa abordagem escolhe o minerador com base em determinadas características do nó, como sua condição, capacidade financeira ou até mesmo tempo de participação. O principal problema é que normalmente os nós com as características equivalentes dominam as decisões e validações, e isso pode ser um problema de controle, este algoritmo estava presente em alguns estudos relacionados ao ambiente IoV (KANG et al., 2019;

MAIO; URIARTE; BRANDIC, 2019; ZHENG; WU; LI, 2023).

- PoU - (*Proof-of-utility*), bastante semelhante ao DPoS, neste algoritmo é realizada uma eleição, onde os nós visando selecionar o minerador com maior utilidade. Essa utilidade pode ser calculada usando múltiplas variáveis que mudam seu peso ao longo do tempo, tornando essas variáveis distribuídas de forma não linear. Essas variáveis podem incluir, poder de processamento, qualidade de rede, taxas de participação, antiguidade, capacidade financeira, podendo variar conforme cada implementação e sua aplicação. PoU é escalável e os autores (DAI et al., 2019a) calcularam este utilitário com base no tempo que o nó leva para fornecer informações. O principal problema com essa abordagem é como a utilidade é calculada. Como o número de variáveis e seus pesos estão todos abertos para o desenvolvedor escolher, uma má escolha pode causar um grande problema de erro na seleção e segurança. Portanto, é difícil implementá-lo de forma segura e correta.

- PoR - (*Proof-of-Reputation*), o PoR desenvolvido em (WANG et al., 2020) é muito semelhante ao PoU ou DPoS. Há uma eleição e a reputação do nó é diretamente proporcional ao seu poder de voto. A principal diferença é que, após a eleição, os nós eleitos não irão minerar, eles irão competir em um desafio PoW, e apenas o vencedor será recompensado. Quanto maior sua reputação, menor a dificuldade do desafio. Essa combinação de DPoS e PoW suprime a maioria das deficiências do PoW, já que não precisamos de toda a rede para competir, e aqueles que competem não gastam tanta energia, pois a dificuldade diminui à medida que a reputação aumenta. Logo devemos também considerar um maior risco de centralização das decisões e validações por determinados nós "eleitos".

- PBFT - (*Proof-Bizantine-Fault-Tolerance*) é um algoritmo não anônimo baseado em tratamento de mensagens maliciosas no mecanismo de consenso. O objetivo é que todos nós ajudem a chegar a um consenso sobre o estado do sistema usando a regra da maioria. Um PBFT na IoV, pode atuar com o requisito do número máximo de nós maliciosos para proteger o sistema. Conforme aumenta o número de nós o sistema que implementa este consenso torna-se mais seguro. É possível, no entanto, em uma BC de consórcio que cada organização possa eleger um único nó para participar do consenso. Os artigos (GAO et al., 2020; QIU et al., 2018) apresentam implementações desta abordagem.

- BFT-DPos - (*Delegated-Proof-of-Participation-Byzantine Fault Tolerance*). Este algo-

ritmo reúne dois conceitos; a prova de participação delegada utilizada para escolher os nós que tentarão alcançar o consenso com base em sua participação; e a tolerância a falhas bizantinas baseada no número de delegados eleitos. Esta é uma maneira escalável e rápida de gerar blocos. Como os usuários têm uma pequena chance de influenciar o resultado, em um cenário real, é provável que os usuários repassem seus votos a algum ente que os representará, para que este ente possa votar neles. Isso pode causar centralização de votação, esse problema é comumente visto em DPoS (FU et al., 2020).

- DPoS - (*Delegated-Proof-of-Stake*). Este algoritmo é uma evolução do PoS, pois a prova de participação delegada é utilizada para escolher os nós que tentarão alcançar o consenso com base em sua participação. A lógica para escolha deste subgrupo de representantes pode causar o problema do domínio da votação, enfraquecendo o processo de consenso independente e justo. O artigo (ZHENG; WU; LI, 2023) apresenta a aplicação deste algoritmo com modificações para melhoria da segurança em IoV.

- TangleCV - Para entender a política desse consenso, é necessário explicar a arquitetura do tangle. É baseado em *Blockchain*, mas possui uma abordagem um pouco diferente. Os blocos podem ser encadeados de forma acíclica, cada bloco possui apenas uma transação e cada nova transação requer validação PoW de duas transações anteriores. Após a validação, uma transação ganha um peso calculado com base na quantidade de trabalho necessária para validar aquela transação, e esse peso é sua confiabilidade. A política tangleCV afirma que uma transação que tenha um peso maior que um limite pode se autovalidar. Na arquitetura tangle, essa política tem muitas vantagens, como reduzir o congestionamento de largura de banda e remover atrasos. No artigo não foram realizados testes de segurança (RATHORE et al., 2019). Além disso, é difícil comparar essa política com outros algoritmos de consenso, pois a sua arquitetura é diferente.

- DM-CDS (YAHIATENE et al., 2019) - O algoritmo do conjunto dominante conectado de mineradores distribuídos é baseado em outro algoritmo de consenso dominante, o DSP-CDS (*distributed single-phase*) que é utilizado para decidir se um determinado nó deve entrar em um conjunto de nós dominantes. Ele se ajusta as mudanças de topologia da rede. O DM-CDS usa uma função de pontuação do minerador que considera vários parâmetros, como modelo de confiança, grau de conectividade, qualidade média do enlace, para decidir qual nó pode validar as transações. Este algoritmo é um pouco

semelhante ao PoS onde, em vez de apostar, o principal fator que decide é a pontuação do minerador. E como a pontuação do minerador é variável com o tempo, os mineradores variam mais do que no PoS.

- VBSBC é um protocolo de consenso baseado em prova de trabalho com baixa latência. Ele mantém características do algoritmo de consenso *Proof of Work* (PoW) ofertando mais segurança com base nas limitações e desvantagens das IoVs. O VBSBC verifica e endossa as transações em suas respectivas zonas com intuito de reduzir a sobrecarga e o número de nós participantes, o que otimiza significativamente o tempo de processamento quando os veículos entram ou saem da rede (TU et al., 2023). Apesar disso, este algoritmo ainda demanda enorme capacidade de processamento, já que ao segmentar em zonas, exige comunicação mais intensiva para mineração e validação. Esta seleção de nós por zona pode resultar no problema do domínio do grupo sobre as decisões ou validações em determinada área, ou zona específica.

Para a coluna da Abordagem Arquitetônica da Tabela 4, foram definidos quatro cenários para melhor classificar as diferentes abordagens nos trabalhos pesquisados. Três características foram levadas em consideração para conceber esses cenários:

- Qual camada é responsável pelas tarefas relacionadas a *Blockchain*?;
- Qual é o papel da borda neste cenário?;
- Qual é o método de alocação de recursos neste cenário?.

Em relação à camada responsável pelas diversas tarefas da *Blockchain* os estudos apresentaram quatro possibilidades de execução, quais sejam:

1. A camada de nuvem (*Cloud-layer*) pode ser utilizada para fornecer alguns métodos, serviços ou aplicações públicas, ou confederadas utilizando a Cadeia de Blocos também no ambiente da IoV. Ao distribuir a *Blockchain* na camada final da arquitetura, a robustez do ambiente auxilia a disponibilidade do serviço (FU et al., 2020).
2. Camada Borda (*Edge-layer*): Nesta arquitetura, as principais atividades relacionadas a BC utilizam recursos computacionais e de comunicação da *Edge/MEC* (GAO et al., 2020; DAI et al., 2019a). Esta camada oferta serviços necessários a infraestrutura e aos usuário finais. Ao deslocar a computação para a borda da rede, há em tese, a depender do *link*,

menos atraso entre as trocas de mensagens, mantendo um poder de processamento melhor do que nas OBUs heterogêneas dos veículos. Tanto a camada do nevoeiro quanto a da borda podem gerar problemas de alocação dos recursos, pois o processamento ocorre em camadas, que apesar de distribuídas, normalmente, oferecem menos poder computacional, de rede, armazenamento e infraestrutura quando comparado aos ambientes de nuvem ou no núcleo dos grandes centros de dados distribuídos (QIU et al., 2018).

3. Camada Nevoeiro (*Fog-layer*): Neste cenário, as tarefas relacionadas a *Blockchain* são executadas apenas nos dispositivos dos usuários ou veículos (OBU), que como já mencionado podem possuir limitações impactantes de processamento e comunicação disponíveis as atividades extras (YAHIATENE et al., 2019; RATHORE et al., 2019).

4. Multicamada Estática (*Static Multilayer*): As tarefas relacionadas a BC são separadas por camada, sendo cada uma executada especificamente na sua camada (GUO et al., 2019a; KANG et al., 2019; ZHANG; LI; CUI, 2018b; TAN; CHUNG, 2020; LIU; ZHANG; YANG, 2018; WANG et al., 2020; ZHENG; WU; LI, 2023). Neste cenário, cada uma das camadas tem suas tarefas atribuídas estaticamente. Isto de alguma forma pode resolver o problema de sobrecarregar uma camada, mas ao não analisar o ambiente do usuário está estratégia pode estar criando novos transtornos, como o da tarefa nunca ser concluída devido um congestionamento em determinada camada.

5. Multicamada Dinâmica (Dynamic Multilayer): Este cenário lida com balanceamento de carga de maneira inteligente híbrida(MAIO; URIARTE; BRANDIC, 2019). Como no cenário anterior, as tarefas relacionadas à BC podem ser tratadas e distribuídas em qualquer camada. No entanto, neste caso, os recursos computacionais de cada camada e o ambiente do usuário são levados em consideração para decidir dinamicamente em qual camada cada tarefa deve ser executada. No entanto, se o mecanismo de decisão escolher uma camada inadequada para executar uma tarefa, isso pode causar alguns dos obstáculos já mencionados nos cenários anteriores.

3.4 AUTENTICAÇÃO CENTRALIZADA (TA) X DESCENTRALIZADA (BLOCKCHAIN)

A Tabela 5 destaca os principais aspectos dos trabalhos que de alguma forma relacionaram as soluções e métodos de autenticação da Autoridade de Confiança (TA) e *Blockchain* (BC).

As referidas pesquisas elencadas, possuem diferentes objetivos de estudo no que se refere as arquiteturas, ambientes, esquemas de implementação e tecnologias de comunicação.

Tabela 5 – Trabalhos Relacionados (*Trust Authority x Blockchain*)

Artigo	Ambiente	TAxBC	Segurança	Solução	Análises
(LI et al., 2019)	VANET	√	RSA e MD5	*Blockchain* em VANET, redes 802.11p	Parâmetros de privacidade de identidade e localização com base nos algoritmos(UGG, IPP, LPP)
(YERRAMILLI; SWAMY, 2019)	Cloud	√	RSA e SHA	Autenticação multifator(TAxBC), para serviços eGov	Processos de usabilidade e gestão dos usuários nas arquiteturas centralizadas e distribuídas
(BRUNNER et al., 2020)	Cloud	√	X.509 e TLS	Implementação de um esquema PKI baseada em *Blockchain*	Desempenho das aplicações de infraestrutura de chaves centralizadas e distribuídas na *Internet*
(RIMBA et al., 2020)	Cloud	√	ECDSA e RSA	Diagnostico econômico das arquiteturas Centralizadas e Descentralizada	Custos financeiros para execução de transações confiáveis entre as soluções *BC Ethereum x TA AWS*
(JIANG; FANG; WANG, 2019)	IoV	√	X.509	Aplicação da tecnologia da *Blockchain* na *Internet* de Veículos	Comportamento de armazenamento de dados distribuídos seguro para Big Data
(AYDIN et al., 2020)	IoT	*	ECDH e DLP	Esquema flexível para autenticação de grupo em ambientes IoT	Eficiência energética dos dispositivos e os custos dos mecanismos de segurança centralizada e distribuída
(ZHENG; WU; LI, 2023)	IoV	√	ECC	Modelo Hibrido de Computação de Borda e Nuvem	Um protocolo eficiente que alia uma TA ao compartilhamento de chaves na Blockchain Pública
SIMA-IoV	**IoV**	√	**ECDH ECDSA AES e SHA**	**Mecanismo de Definição da Autenticação baseado no ambiente da IoV**	**Análise de Desempenho das soluções de Segurança TA e BC com relação aos cenários e ambientes da IoV**

Fonte: O autor (2023)

O estudo (LI et al., 2019) comparou a BC e TA com relação aos parâmetros da privacidade e localização, utilizando redes com padrão 802.11p. Além de adotar um cenário 5G-MEC, distinto desta tese, pois compara o tempo de autenticação e sobrecarga na troca de mensagens das duas soluções na IoV.

Já a pesquisa (YERRAMILLI; SWAMY, 2019) buscou realizar uma comparação dos mecanismos de autenticação centralizados (TA) e descentralizados (BC) no contexto de aplicações *eGov* na nuvem. Esta tese busca avaliar e comparar e definiri o melhor método de autenticação no ecossistema da IoV na borda das redes 5G.

O estudo (BRUNNER et al., 2020) apresentou a implementação de um esquema de infraestrutura de chave pública PKI(*Public Key Infrastructure*), baseada em *Blockchain* na nuvem, comparando aspectos da eficácia com o serviço da Autoridade Certificadora no núcleo. Esta tese faz uma análise de desempenho, taxa de vazão e tempos de autenticação e validação das

transações, das soluções TA e BC no ambiente da IoV na borda com redes 5G.

O artigo (RIMBA et al., 2020) realiza uma análise dos custos econômicos da tecnologia *Blockchain* e da Autoridade de Confiança quando executadas em IaaS(*Infrastructure as a Service*). O referido estudo utilizou os ambientes de serviços na nuvem da *AWS* para (TA) e da *Ethereum* para (BC), mas não considerou ambiente/cenário na IoV com comunicação 5G de borda.

Já o estudo (JIANG; FANG; WANG, 2019) faz uma análise da utilização da *Blockchain* na *Internet* de Veículos. O referido estudo realizou avaliações de desempenho em relação ao armazenamento distribuído de *Big Data* em redes 4G/LTE. Esta pesquisa de tese analisa desempenho dos métodos de autenticação e gerência de chaves em trocas de mensagens de tráfego no ambiente da IoV em redes 5G na borda.

A pesquisa de (AYDIN et al., 2020) propõe uma abordagem para autenticação flexível de grupos (GAS - *Group Authentication Schemes*) para IoT em redes 802.11ac. O foco da pesquisa foram os dispositivos com recursos limitados. Uma arquitetura adaptável foi elaborada com foco na eficiência energética e atuação centralizada ou distribuída.

A pesquisa (ZHENG; WU; LI, 2023) apresenta uma separação de tipos de serviços da BC e TA, onde os dados de autenticação permanecem no modo centralizado (TA) e os blocos das transações de informações sobre a rede e seus serviços compartilhados são individualizados para construir uma cadeia integrada. Desta forma, a análise da situação de autenticação do veículo pode ser isolado dos dados referente a privacidade do veículo, aumentando a proteção da segurança e privacidade. A combinação da computação de borda e a blockchain amplia a capacidade dos recursos para alcançar maior eficiência das aplicações de forma segura e de forma combinada, levando em consideração que teremos dois pontos de falhas e a depender do modo de implementação outras adversidades de interação já mencionadas nesta seção.

3.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

O Capítulo 3, retratou os relevantes estudos exploratórios relacionadas ao desenvolvimento desta tese. Levando em conta que os conteúdos destas pesquisas que apresentam tecnologias em progressiva atualização tanto nas redes móveis quanto na Internet Veicular, os diversos trabalhos aqui abordados não tiveram pretensão de esgotar todo o conhecimento com relação à revisão do tema e suas evoluções, buscando explorar os atributos que se associam com a pesquisa.

4 CAPÍTULO MECANISMO SIMA-IOV

Este capítulo demonstra os principais elementos constitutivos da Seleção Inteligente do Método de Autenticação para o Ambiente IoV em redes 5G SIMA-IoV, conforme delineado na subseção 4.1. Em seguida, a subseção 4.2 apresenta a arquitetura que suporta os métodos de autenticação e o mecanismo inteligente. A subseção 4.3 explora a concepção do mecanismo, descrevendo o ambiente de análise, suas métricas, as etapas de busca e seleção e os cenários e ambientes de simulação.

4.1 ELEMENTOS DA TESE

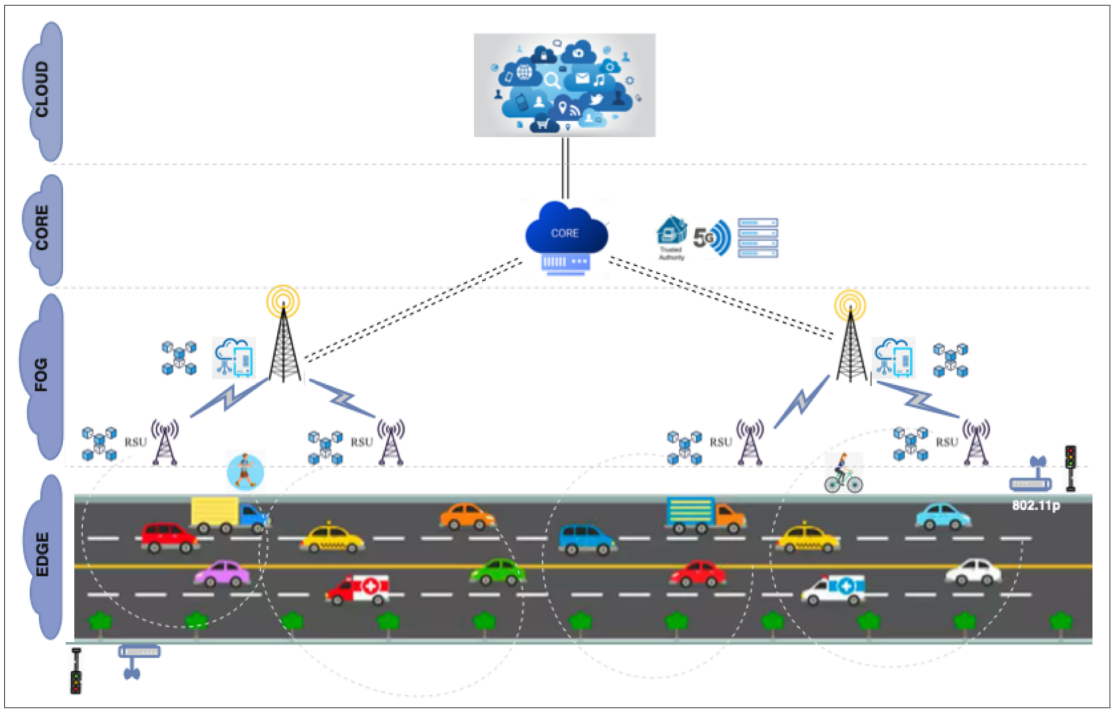
No âmbito das tecnologias de proteção e segurança na IoV, a implementação dos métodos de autenticação visam mitigar os transtornos causados por possíveis falhas, erros, atrasos ou até mesmo as ações maliciosas que possam surgir. Esses fatores podem desencadear desde pequenas divergências de trajetória até incidentes com potencial risco a vida e/ou bens dos bilhões de usuários futuros deste ambiente. Portanto, uma definição apropriada do método de autenticação e a identificação dos seus elementos arquiteturais proporcionam melhorias nas garantias necessárias para o desenvolvimento contínuo das redes de comunicação e dos sistemas de transporte viário, tão importante para mobilidade mundial (LIANHAI et al., 2019).

A Figura 16 apresenta o cenário da Internet Veicular (IoV), no contexto da tecnologia de comunicação 5G. Nessa representação, são evidenciadas as diversas camadas, elementos de comunicação, dispositivos participantes, centrais de processamento, alocação de serviços e aplicações. O mecanismo SIMA-IoV baseado na OB teve como objetivo a análise das características deste ambiente, utilizando exclusivamente a tecnologia de comunicação móvel de quinta geração no modo (*standalone* - *SA*). Outras tecnologias de redes sem fio, como 4G, WiFi 6, Bluetooth, LoRa e ZigBee, não foram analisadas nas simulações e resultados apresentados nesta pesquisa.

Dentro do escopo desta tese, a arquitetura implementada foi elaborada tendo como possíveis usuários os fornecedores, distribuidores e integradores de redes móveis veiculares, bem como os concessionários de vias públicas ou privadas que oferecem serviços e aplicações que dependam desta modalidade de comunicação no ambiente da IoV. O objetivo principal do mecanismo concebido SIMA-IoV é selecionar qual o método de autenticação é mais apropri-

ado a determinado cenário, isto, levando em conta a quantidade de veículos e a capacidade da infraestrutura, isto considerando o impacto direto que este tipo de serviço têm no tempo inicial de acesso dos usuários ao ambiente IoV.

Figura 16 – Ambiente IoV - Centralizada(TA) / Descentralizada(BC)



Fonte: O autor (2020)

A Figura 16 apresenta no topo da hierarquia, a camada da nuvem (*cloud*) que hospeda os serviços, aplicações e recursos mais robustos, normalmente virtualizados e escaláveis, mas com atrasos maiores e qualidade de serviços mais baixos por questões relacionadas às distâncias destes serviços e aplicações com relação aos usuários finais. A Figura 16 considera apenas um operador fornecedor do ambiente de telecomunicação 5G/IoV para ilustrar o ambiente escopo desta pesquisa.

Na camada do núcleo (*core*) do fornecedor residem as aplicações e serviços disponibilizados aos seus clientes e parceiros, já o núcleo da rede 5G consiste em uma série de módulos funções de serviços localizados no centro de dados do fornecedor, abrangendo todos os serviços necessários a tecnologia de comunicação de quinta geração. Estes serviços e aplicações podem funcionar de maneira centralizada ou distribuída entre vários centro de dados do operador, ou de parceiros terceirizados. Para esta tese a Autoridade de Confiança (TA) foi implementada, no núcleo do fornecedor, podendo estender ou integrar outras funções e serviços de rede.

A camada de nevoeiro (*fog*) pode incluir infraestrutura de centro de dados reduzida para

serviços virtualizados, localizada em estações de comunicação(BS) mais próximas dos dispositivos finais posicionados na borda (*edge*). Esta camada pode implementar o plano de controle e demais aplicações tais como a mineração e o consenso da *Blockchain*. Nesta tese, a arquitetura é flexível, mas para fins de análise e avaliação dos métodos de autenticação, as simulações consideraram a TA no núcleo com a latência negligível no *backhaul*, entre borda e núcleo e a *Blockchain* distribuída na camada (*fog*) com suas principais atividades e consenso.

A concepção do mecanismo SIMA-IoV é baseada nos atributos pertinentes as arquiteturas de implementação (centralizados e descentralizados), funcionalidades (autenticação, gerência de chaves e validação de mensagens) e demais especificidades (infraestrutura, tecnologia de comunicação, dispositivos ativos e passivos de rede, número de usuários, número de veículos e tipos de tráfego) no ambiente da IoV.

4.2 ARQUITETURA DO MECANISMO SIMA-IOV

O SIMA-IoV introduz o mecanismo inteligente de seleção projetado para selecionar o método de autenticação mais eficiente, levando em consideração as condições específicas de diferentes cenários. Essa abordagem é baseada em inteligência artificial, aplicando especificamente a técnica de otimização bayesiana, que visa resolver os problemas onerosos da busca e otimização. A tese objetivou fornecer um mecanismo para que as corporações das telecomunicações possam determinar o método mais apropriado de autenticação em um determinado cenário do ambiente IoV. O SIMA-IoV tem como propósito facilitar a tomada de decisão, buscando selecionar o método de autenticação melhor para determinado cenário/localidade. A responsabilidade pela escolha de determinado método de autenticação e validação das informações, não é uma tarefa simples, pois existe uma complexidade nos parâmetros do ambiente, altamente dinâmico. O profissional de Tecnologia da Informação deve considerar uma ampla heterogeneidade de variáveis, parâmetros e elementos, fluxo de tráfego, tecnologias de comunicação, quantidade de nós, dispositivos, infraestrutura disponível, serviços e aplicações, além de outras particularidades relacionadas às questões de arquitetura de redes e custos de cada organização (RIMBA et al., 2020).

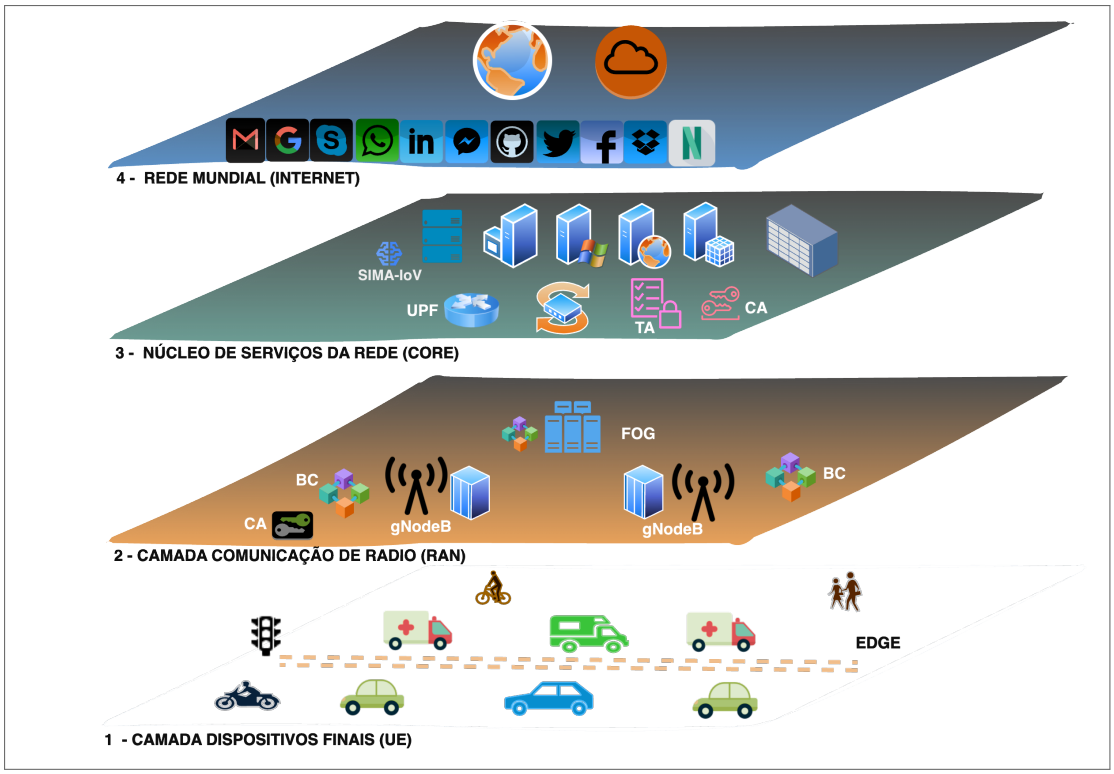
A aplicação de algoritmos inteligentes de otimização, permite uma compreensão mais aprofundada das relações de desempenho e eficácia dos métodos de autenticação com relação a vários cenários no ambiente da IoV, bem como o comportamento de cada um destes parâmetros a medida que dinamicamente se modificam durante o processo de aprendizado no decorrer das

simulações (CASTILLO; IBAñEZ, 2018a).

A indevida aplicação de um método de autenticação centralizada pode acarretar sobrecarga em um ambiente IoV com alta densidade de nós e a utilização intensiva de determinados serviços ou aplicações. Por outro lado, a adoção de uma solução de autenticação descentralizada em um ambiente IoV com baixa densidade de nós ou infraestrutura insuficiente pode resultar em escassez de recursos para a execução eficiente das atividades distribuídas. A principal questão reside em como selecionar o melhor método de autenticação, dado que os ambientes e cenários de uma IoV podem ser consideravelmente diversificados, diferindo significativamente de outros ambientes onde é possível mapear os parâmetros de forma regular, como em uma rede corporativa de fácil compreensão e definição dos serviços e aplicações.

A Figura 17 apresenta uma visão da arquitetura de análise do mecanismo inteligente SIMA-IoV para definição da escolha do melhor método de autenticação, baseada na técnica de otimização bayesiana. A arquitetura é definida em 4 camadas, sendo a primeira camada dos dispositivos finais (*Edge/UE*) que utilizam a rede de comunicação e seus serviços, a segunda camada é do nevoeiro (*Fog*), já a terceira camada é a do núcleo de uma operadora (*Core Network*) e a quarta e última é a camada da nuvem com todos os seus recursos e aplicações abrangentes (*Internet*).

Figura 17 – Visão Geral da Arquitetura de Análise do SIMA-IoV



Fonte: O autor (2023)

Na camada 1, dos usuários finais (UE), se encontram todos os dispositivos que demandam recursos do ambiente IoV. Os dispositivos presentes nesta camada iniciam o processo de autenticação com determinado método realizando a troca das chaves de validação das transações, possibilitando a utilização dos serviços, aplicações e compartilhamento de informações. No caso do serviço da *Blockchain*(BC) distribuída, os dispositivos da camada 1, possuem a atividade de resguardar suas chaves privadas e podem participar de atividades relacionadas a seleção dos mineradores ou das unidades responsáveis pela realização da atividade do consenso. Com relação ao serviço da Autoridade de Confiança(TA), os dispositivos desta camada, possuem a atividade de resguardar suas chaves privadas para o funcionamento do sistema de autenticação e troca de mensagens.

A camada 2, rede de acesso rádio (RAN), dispõe dos equipamentos de comunicação de redes e as estações base (*gNodeBs*) do padrão 5G. Estes dispositivos podem auxiliar e participar do processo de autenticação e gerência de chaves a depender dos seus recursos e da maneira como a arquitetura é definida e implementada. Na *Blockchain* desta tese, a camada 2, assume as atividades de mineração, consenso e distribuição dos registros de blocos válidos para os nós.

Na camada 3, Núcleo de Serviços da Rede *Core*, foram implementados os métodos de autenticação e o Mecanismo Inteligente de Seleção do Método de Autenticação no Contexto IoV (SIMA-IoV), conforme apresentado pela Figura 17. Na camada *Core*, encontram-se os serviços essenciais para o fornecimento e acesso a *streaming*, banco de dados, informações de conta e perfil, entre outros, disponibilizados por cada operadora. As informações obtidas das três primeiras camadas e seus respectivos serviços são analisadas pelo mecanismo SIMA-IoV, buscando apoiar a tomada de decisão sobre o método de autenticação mais apropriado a determinado ambiente IoV examinado. Nesta camada, também são gerenciados os encaminhamentos das solicitações para as aplicações requisitadas na camada superior da Internet. A camada 4 pode ser utilizada pelos métodos de autenticação de forma complementar, como a terceirização de uma possível CA ou outro serviço ou aplicação na nuvem.

4.3 CONCEPÇÃO SIMA-IOV

O SIMA-IoV (Seleção Inteligente do Método de Autenticação) para Internet dos Veículos, é um mecanismo projetado para otimizar a seleção e apoiar a tomada de decisão de gestores e administradores de redes sobre qual método de autenticação utilizar em determinado ambiente da Internet Veicular(IoV). Com a crescente complexidade e dinamismo das redes veiculares,

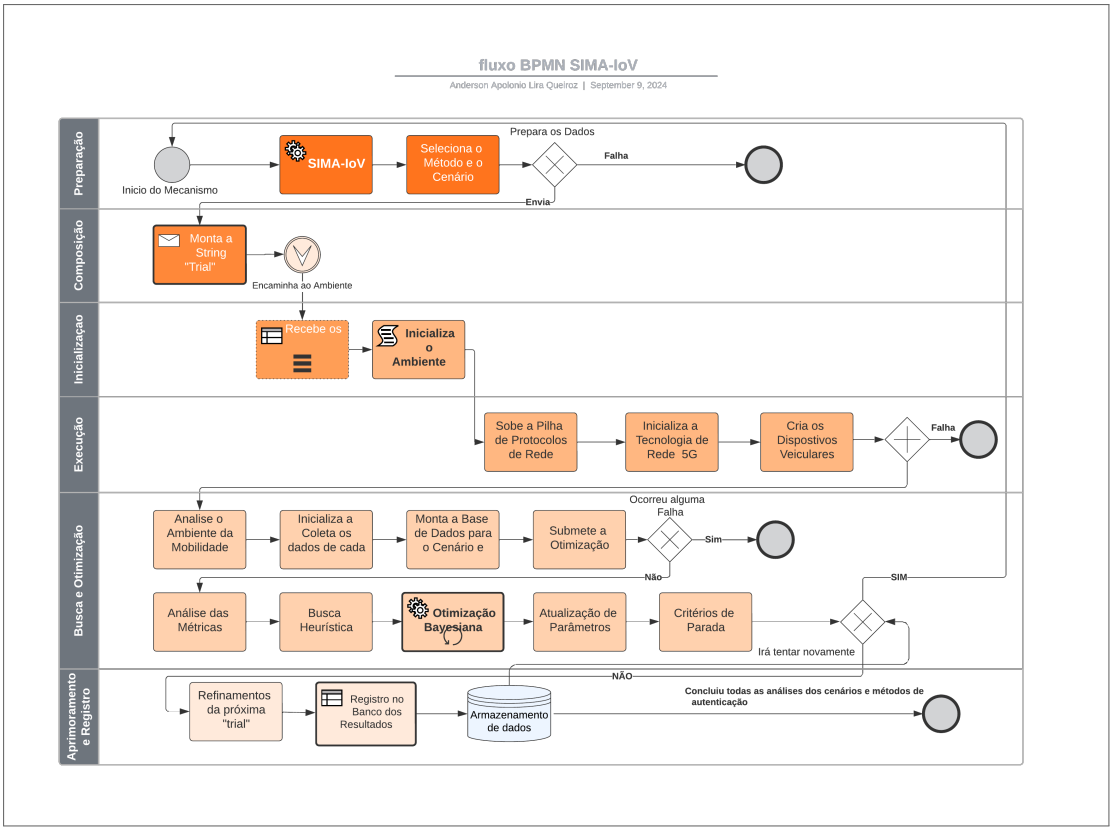
a necessidade por métodos eficientes de autenticação tornou-se crítica. O SIMA-IoV utiliza técnicas avançadas de otimização e aprendizado de máquina para analisar e selecionar o melhor método de autenticação com base em múltiplos parâmetros e condições específicas de cada cenário, sejam eles urbanos ou rurais. Tetando soluções de autenticação centralizados, como a *Trust Authority* (TA), e descentralizados, como a *Blockchain*, o mecanismo busca proporcionar um equilíbrio entre eficiência e resiliência, adaptando-se às demandas dinâmicas das redes veiculares modernas.

A Figura 18, apresenta o fluxo de trabalho do mecanismo de seleção do método de autenticação na Internet de Veículos. O processo inicia no bloco "Mecanismo de Seleção do Método de Autenticação", responsável por selecionar o método de autenticação, TA, BC(PoW ou PoS) e o cenário de simulação, com 2, 4 ou 6 *gNodeBs*, com variação de 100 a 1000 veículos. Em seguida, monta-se a *"trial"* inicial, cujos parâmetros são enviados ao simulador. O simulador então inicializa a pilha de protocolos, instancia a rede 5G, cria os dispositivos veiculares e inicializa a simulação no ambiente. Durante a simulação, as mobilidades realizadas dentro do espaço cenário definido e os dados resultantes são coletados. Estes dados são, então, submetidos à análise das métricas e busca heurística dentro da etapa de otimização bayesiana. Caso os critérios de parada não sejam atingidos, os parâmetros são atualizados e refinamentos são feitos com base nos resultados obtidos. Este ciclo continua até que os critérios de parada sejam atendidos, culminando na seleção do método de autenticação mais apropriado para o cenário especificado.

Para o desenvolvimento e implementação da abordagem nesta tese, que visa a otimização inteligente da seleção do método de autenticação apropriado a cada cenário, objetivando a melhoria da gestão e administração das redes na IoV, foram adotadas algumas ferramentas e bibliotecas para suporte do mecanismo: ns-3, OMNeT++, SUMO, 5GLena, SIMU5G, INET, VeinS e Crypto++. Além disso, foram testados vários métodos de inteligência artificial, incluindo redes neurais recorrentes (RNN), unidades recorrentes fechadas (GRU), memória de longo prazo recorrente (LSTM), bandidos multi-armados (MAB), bandidos multi-armados contextuais (CMAB), algoritmos genéticos e, por fim, otimização bayesiana (OB). Entre estas técnicas de IA, a OB mostrou-se satisfatória para atender aos objetivos estabelecidos, proporcionando resultados mais robustos e eficazes (YE, 2022).

A implementação do mecanismo inteligente no ambiente IoV em Redes 5G (SIMA-IoV), visa selecionar entre os métodos de autenticação, Autoridade de Confiança (TA) e *Blockchain* utilizando dois algoritmos de consenso (PoW/PoS), qual método é mais apropriado para um

Figura 18 – BPMN do SIMA-IoV

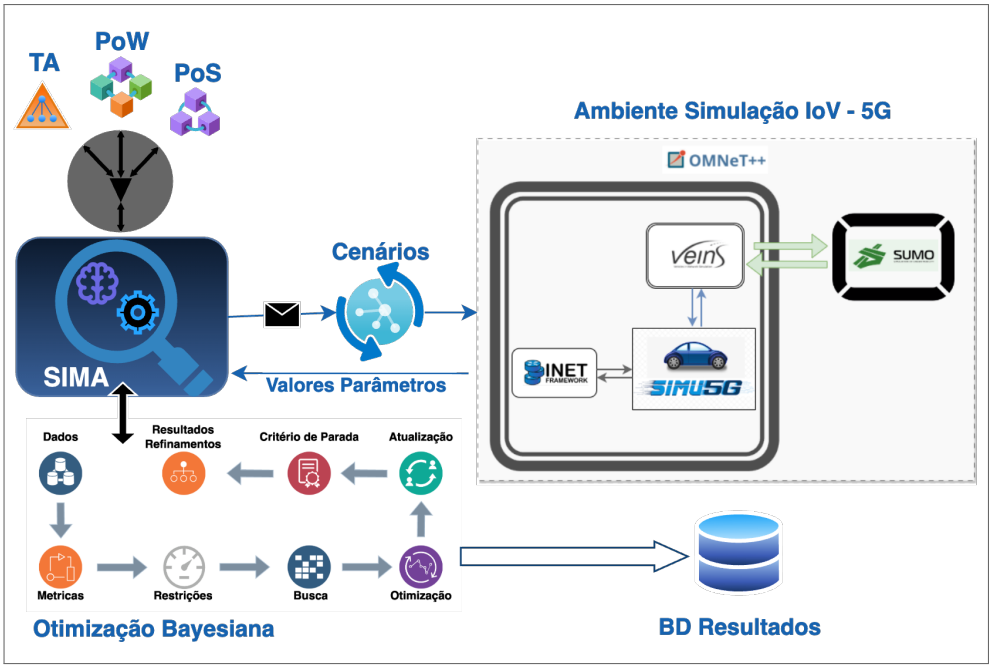


Fonte: O autor (2023)

determinado cenário das redes veiculares. Esse mecanismo baseia-se na análise dos parâmetros de cada cenário para selecionar qual método apresenta melhores resultados conforme os requisitos estabelecidos. O objetivo é garantir que a escolha seja criteriosa e precisa, superando uma decisão baseada exclusivamente em julgamento técnico humano ou inspeção básica dos dados (DAI et al., 2019b).

A Figura 19, apresenta o SIMA-IoV com o Ambiente Simulação da IoV e no padrão de comunicação 5G, utilizando a técnica de Otimização Bayesiana com heurística, que seleciona um dos possíveis métodos de autenticação, Autoridade de Confiança (TA), *Blockchain* PoW ou PoS. A partir disso, o SIMA-IoV monta um possível cenário, passando os parâmetros do número de carros (100 a 1000) e infraestrutura (2,4 e 6) para o simulador OMNeT++, este recebe as informações por meio do *"trial"*, iniciando a rodada de simulação com base no cenário e método de autenticação selecionado. Ao final, o simulador do Ambiente IoV/5G retorna os valores dos parâmetros requisitados pelo SIMA-IoV. Estes valores são então submetidos novamente à técnica de otimização bayesiana para análise das métricas, busca, otimização, atualização e refinamentos dos próximos valores a serem simulados. O objetivo é encontrar,

Figura 19 – Concepção do SIMA-IoV



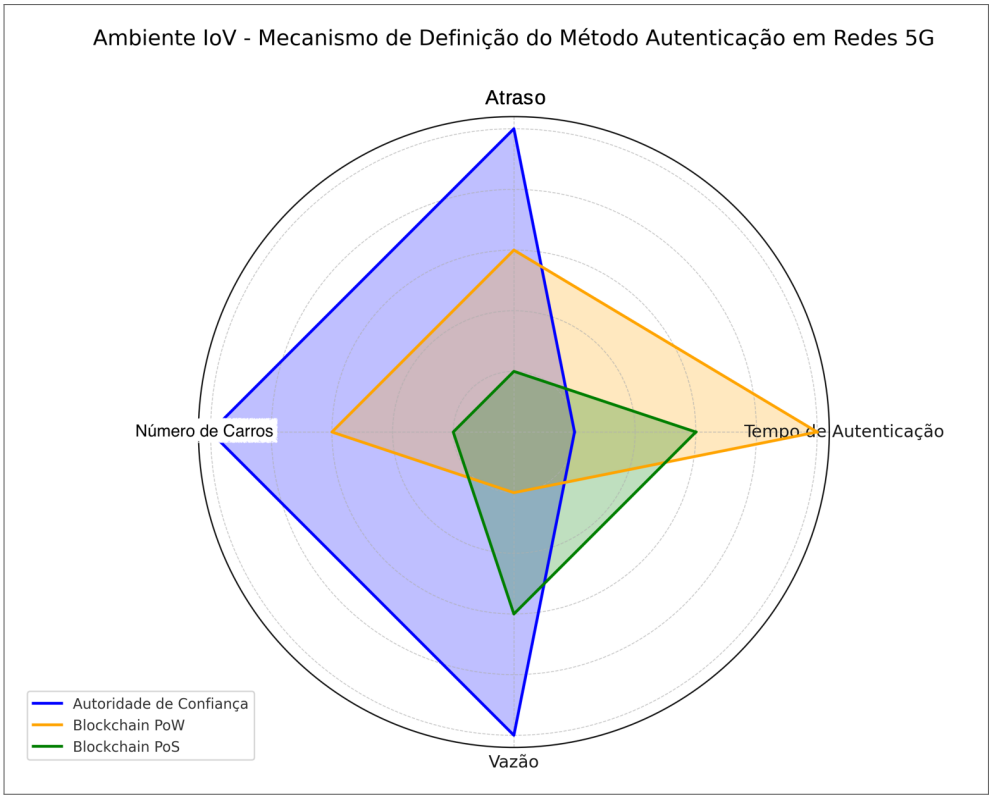
Fonte: O autor (2023)

para cada cenário e método, o melhor valor possível que será registrado no banco de dados. Este processo cíclico continua até que os critérios de parada sejam atendidos, garantindo a otimização contínua de todos os cenários simulados.

Para alcançar maior precisão, foi empregada as técnicas de IA associada ao método de otimização bayesiana com esquema heurístico de busca do modo *Panzen Tree Estimator* (TPE). O ambiente considerado nesta tese não possui um *dataset* preexistente para treinamentos e testes, por isso os dados foram gerados com base nos requisitos dos processos e nas métricas dos dispositivos e da rede. Esses dados foram obtidos a partir da escolha de um método de autenticação específico combinado com um cenário de simulação estabelecido na subseção 4.4 (TA/BC).

O SIMA-IoV selecionará uma das soluções de autenticação possíveis TA ou BC(PoW/PoS), para cada rodada de simulação, onde o problema é delimitado, a atuação se direciona para a função de decisão, que, com base nos pesos atribuídos as variáveis dos parâmetros, seleciona algum método dentre os disponíveis. Logo, este método passa por um processo de busca, avaliação e otimização, resultando em um valor definido pelo mecanismo de otimização, este processo ocorre a cada rodada da simulação no ambiente. Este processo tem o intuito de encontrar o melhor resultado, de um método em relação aos requisitos de cada cenário do ambiente examinado.

Figura 20 – Aspectos de Análise dos Parâmetros do Ambiente IoV



Fonte: O autor (2024)

A Figura 20, ilustra o comportamento das métricas gerais do ambiente. O SIMA-IoV coleta as informações solicitadas e as envia para as funções de otimização que avaliam o estado e detectam a necessidade de possíveis atualizações dos valores e dos resultados a cada rodada do processo. Para possibilitar as devidas simulações do SIMA-IoV foi implementado todo o ambiente IoV integrado no simulador OMNeT++, com a utilização da linguagem de programação e compilação do C++ para desenvolvimento dos métodos de autenticação, as bibliotecas de simulação, os protocolos de redes, os simuladores veiculares, de mobilidade e da tecnologia de comunicação 5G. As funções inteligentes de **otimização *bayesiana*** foram desenvolvidas em *Python*, com a comunicação entre estes processos sendo realizada via *Application Programming Interface* (API) de *Sockets* (interfaces de comunicação entre processos utilizando a rede). Após obter informações sobre o ambiente e a infraestrutura da IoV, esses dados foram utilizados para configurar o mecanismo inteligente que seleciona o método de autenticação, levando em conta a análise dos parâmetros correlacionados do ambiente IoV examinado.

4.3.1 Funções Bayesiana

A metodologia adotada baseia-se na otimização bayesiana, que ajusta os hiperparâmetros de maneira eficiente e aprimora o desempenho do modelo ao explorar diferentes métodos de forma criteriosa. Ao contrário dos algoritmos genéticos, que frequentemente envolvem um alto custo computacional devido à grande quantidade de simulações e cruzamentos, além do risco de ficarem presos em uma única solução sub ótima, a OB utiliza um modelo probabilístico para explorar de forma inteligente o espaço de busca. Dessa forma, ela encontra rapidamente configurações otimizadas com menos iterações, equilibrando exploração e exploração, o que a torna mais adequada para problemas com restrições computacionais. Esta etapa da solução é descrita pelo Algoritmo 1, onde a Função Objetivo sorteia um *trial*, define um método de autenticação, o número de veículos, infraestrutura e cenário de avaliação, gerando um comando com esses parâmetros que servirão de entrada para a execução da simulação. Ao final da execução desta função é retornado o estado como valor da função objetivo.

Algoritmo 1: Função Objetivo

Entrada: Cenário de Avaliação (trial)
Saída: Valor da Função Objetivo
Função `Objetivo (trial)`**:**

$solução \leftarrow SelecionarSolução(trial)$;
$num\_carros \leftarrow DefinirNúmeroCarros(trial)$;
$num\_infras \leftarrow DefinirNúmeroInfraestruturas(trial)$;
$Comando \leftarrow CriarComando(solução, num\_carros, num\_infras)$;
$estado \leftarrow ExecuçãoDoAmbiente(Comando)$;
return $estado$;

Fonte: O autor (2024).

A OB é uma técnica eficiente para otimizar funções de custo ou desempenho que são relativamente caras de se avaliar, empregando estratégias probabilísticas para prever o comportamento da função objetivo com base em amostras anteriores, geralmente um processo gaussiano, para criar uma função de aquisição que sugere os próximos pontos a serem avaliados. Essa função de aquisição equilibra a exploração de áreas desconhecidas com a exploração de áreas promissoras, maximizando a eficiência da busca pelo ótimo global. Sendo assim, a otimização do estudo envolve a execução iterativa da Função Objetivo com diferentes parâmetros para encontrar a melhor configuração que otimiza o desempenho do ambiente. O processo

de otimização é descrito pelo Algoritmo 2, é inicialmente criado um estudo de otimização com a configuração definindo o nome do estudo e o banco de dados para armazenamento.

Algoritmo 2: Otimização do Estudo

Input: Configuração do Estudo
Output: Resultados Otimizados
Function `Otimização (trial):`

$study \leftarrow InicializarEstudo(configuração)$;
$DefinirCritériosOtimização(study)$;
$study.optimize($**objective**$, n\_trials = 60)$;
$resultados \leftarrow CarregarResultados(study)$;
foreach *tentativa* $\in resultados$ **do**
  $ExibirParâmetrosValores(tentativa)$;
$melhores\_resultados \leftarrow ObterMelhoresTentativas(study)$;
foreach *melhor* $\in melhores\_resultados$ **do**
  $ExibirParâmetrosValores(melhor)$;
return $melhores\_resultados$;

Fonte: O autor (2024).

Em seguida são definidos os seguintes critérios de otimização: minimizar o tempo de autenticação, maximizar o consumo do link, minimizar o atraso médio e maximizar o número de carros. A partir disso, inicializa-se o processo de otimização usando a Função Objetivo, realizando 60 tentativas. Uma vez que esse processo é encerrado, os resultados das tentativas de otimização são carregados e retornados os melhores resultados encontrados.

O ajuste é realizado dinamicamente aplicando a OB para os parâmetros do ambiente, visando maximizar a eficiência do sistema. A Função Objetivo avalia diferentes cenários, enquanto o processo de otimização busca a melhor configuração para o cenário do ambiente simulado, garantindo os melhores resultados conforme os critérios definidos.

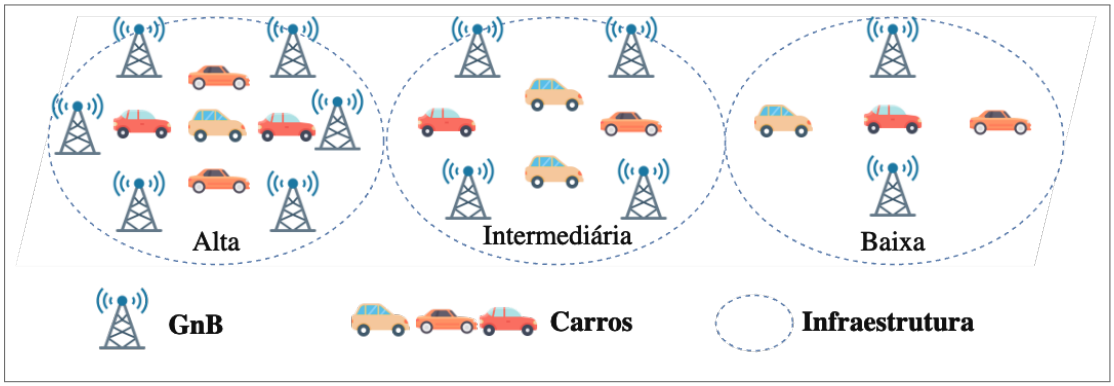
Fonte: O autor (2024).

4.4 CENÁRIOS SIMA-IOV

O mecanismo SIMA-IoV foi utilizado em diferentes configurações de cenários. Os episódios visam abstrair características do ambiente, onde cada operadora possui diferentes infraestruturas a depender de cada região. A definição, considerará as capacidades de Infraestrutura,

categorizando como Baixa, Intermediária e Alta com base na quantidade de estações de comunicação, para Quantidade de Veículos, representada pela Figura 21.

Figura 21 – Abstração de Ambientes IoV



Fonte: O autor (2022)

A partir da definição destes ambientes será gerada a variação no número de veículos ao longo do tempo conforme as cargas representadas na Tabela 6. Através da otimização buscamos obter a decisão sobre qual método de autenticação TA ou BC(PoW/PoS), melhor explora os recursos de comunicação móvel e da infraestrutura disponíveis em determinado cenário IoV examinado.

Tabela 6 – Cenários Simulação

Cenário	Infraestrutura	*N* gNodeBs	*N* Veículos	*N* Transações
1	**Baixa**	**2**	**100-1000**	49200
2	**Intermediária**	**4**	**100-1000**	49200
3	**Alta**	**6**	**100-1000**	49200

Fonte: O autor (2024).

As IoVs podem abranger diferentes tipos de dispositivos e tecnologias, estes elementos influenciam diretamente nos diversos parâmetros e requisitos que compõe o ambiente. Para obtenção das métricas aplicadas nesta tese foram utilizados os seguintes parâmetros da IoV apresentados na Tabela 7.

4.4.1 Elaboração do Ambiente de Simulação para o SIMA-IoV

O ambiente de simulação inicial utilizou a ferramenta Simulador de Redes v.3 (ns-3), conhecida por sua robustez e flexibilidade na simulação de ambientes de redes. No entanto,

Tabela 7 – Conjunto de Dados Base

Consumo da Banda de Redes	Percentual de utilização da largura de banda total disponível da tecnologia de rede
Infraestrutura	Número de *gNodeBs*.
Número de carros	Quantidade de veículos conectados na rede em determinado ambiente
Tempo de resposta	Intervalo de tempo que leva para um dispositivo enviar um pedido de comunicação e receber uma resposta do dispositivo de destino na mesma rede
Tempo de Autenticação	Intervalo de tempo total que leva um dispositivo para se autenticar em determinado método no ambiente IoV

Fonte: O autor (2024).

durante o desenvolvimento e a execução das simulações na tese, identificamos limitações significativas relacionadas à reprodução da mobilidade e especificamente na funcionalidade de *"handover"*, não disponível na biblioteca de comunicação 5G do projeto 5GLena[1] a única compatível com o simulador ns-3. A capacidade de gerenciar a transição de conexões móveis entre diferentes pontos de acesso entre as estações base é crucial para a precisão das simulações em cenários de redes móveis avançadas como os da IoV.

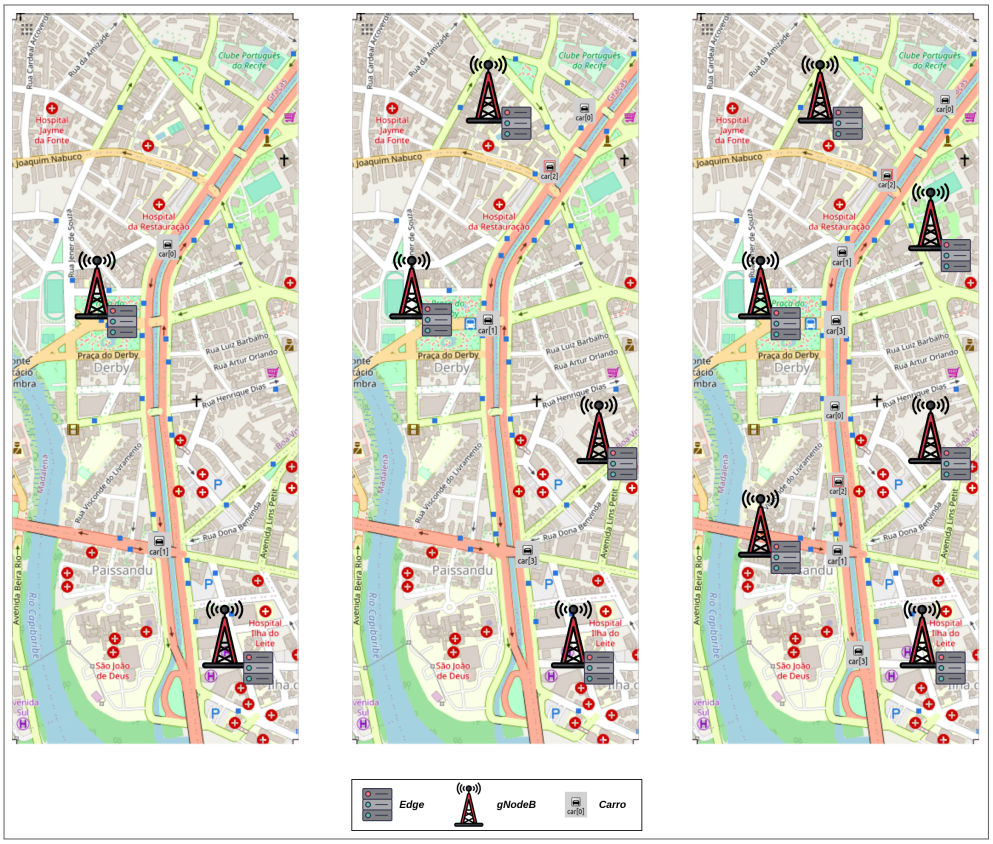
Devido a essas limitações, tornou-se necessário migrar para um ambiente de simulação capaz de superar tais restrições. Esse novo ambiente não apenas supera as barreiras de mobilidade e *"handover"*, mas também incorpora uma gama de tecnologias e atributos gráficos de simulação essenciais para parametrização correta do ambiente IoV. Com isso, pudemos implementar as simulações de forma mais detalhada, incluindo cenários avançados de mobilidade próximos aos reais (Av. Agamenon Magalhães, Recife PE - BR). Com todos estes atributos e requisitos foi possível o desenvolvimento do ambiente IoV, garantindo que as simulações pudessem atender aos requisitos de desempenho necessários para o SIMA-IoV.

A partir das ferramentas apresentadas nas seções anteriores, o primeiro passo foi implementar o ambiente de mobilidade, apresentado na subseção 4.2. Os veículos transitam em determinada região, Av. Agamenon Magalhães na Cidade do Recife PE Brasil, comunicando-se com as estações da infraestrutura de telecomunicações *gNodeBs* alocadas no trajeto conforme cada cenário predefinido para as simulações. Após a criação e parametrização do ambiente no arquivo *.NED* do Simulador de Eventos Discretos (OMNeT++), foram configurados os dispositivos dos veículos (OBUs) de maneira que utilizassem a estrutura prototipada, e os

[1] Instituto CTTC (Centre Tecnológic de Telecomunicacions de Catalunya)

métodos de autenticação, conforme apresentado na Figura 22.

Figura 22 – Ambientes Simulação



Fonte: O autor (2024)

Tabela 8 – Parâmetros utilizados na simulação.

Banda (f_c)	3.5 GHz
Número de *Resource Blocks*	25
Numerologia	0
gNB txpower	43dBm
UE's txpower	18dBm
Tempo de simulação	250s
Número de Veículos	100-1000
Número de gNBs	2|4|6
Área em M2	575.048,19

Fonte: O autor (2023)

A Tabela 8 apresenta os valores dos parâmetros adotados na simulação. Os parâmetros de Banda, Número de Blocos de Recursos e Numerologia foram definidos conforme preconiza os dados das normatizações do 3GPP para o padrão 5G convencional. Além disso, a Tabela apresenta as potências de transmissão tanto das estações-base quanto dos dispositivos finais.

Os valores para o tempo de simulação, bem como o número de veículos e estações-base, foram selecionados com base nos dados disponíveis no conjunto de dados disponibilizado pela prefeitura da cidade do Recife PE Brasil 4.5, objetivando encontrar os limites de capacidade para validação do mecanismo SIMA-IoV no ambiente.

4.5 CONJUNTO DE DADOS DA REGIÃO SELECIONADA

A compreensão e interpretação das características dos dados públicos da região são cruciais para uma definição precisa dos parâmetros do ambiente na IoV. Diante da busca por esta concepção, a escolha do número de veículos para o ambiente simulado foi baseada nos dados dos Sensores de Tráfego localizados na área selecionada de uma das maiores Avenidas da Cidade Recife em Pernambuco Brasil, Figura 23 no período dos meses de janeiro a dezembro do ano de 2019, sendo estes os últimos dados publicados, desconsiderando o ano de 2020 (período pandêmico). Os dados estão disponíveis no site da Companhia Transito e Transporte Urbano da Cidade do Recife (CTTU PCR/PE-BR) [2].

Figura 23 – Av. Agamenon Magalhães, Recife-PE Brasil

Fonte: (Google Street View, 2024)

O monitoramento do fluxo de veículos em áreas urbanas é crucial para a gestão do tráfego, planejamento urbano e implementação de políticas públicas eficientes neste segmento. Na Cidade do Recife, dados coletados por alguns sensores instalados nas principais vias oferecem uma visão do comportamento do tráfego em determinada região. A análise dos dados do fluxo

[2] http://dados.recife.pe.gov.br/dataset/velocidade-das-vias-quantitativo-por-velocidade-media-2019

de veículos proporcionou extrair as informações necessárias para parametrizações do conjunto de dados simulados a serem gerados nesta tese.

A Metodologia teve como suporte os dados que contemplam informações sobre a contagem de veículos em diferentes faixas de velocidade em intervalos de minutos. Os registros estão estruturados da seguinte forma:

- Identificador
- Mês
- Equipamento de medição
- Faixa da via
- Data e hora
- Intervalo de minutos
- Quantidade de veículos em diferentes faixas de velocidade

A tabela 9 apresenta o resumo dos dados obtidos no *dataset* citado, considerando as médias dos veículos por minuto e a quantidade máxima registrada, isto somando todos os segmentos dos grupos por categoria de velocidade (0-10,11-20,21-30,31-40,41-50,51-60,61-70,71-80,81-90,91-100 km/h).

Através dos parâmetros da data/hora e quantidade de veículos por segmento foi possível analisar os padrões de tráfego e os horários de pico e os períodos de menor movimento. A escolha do número de veículos para simulação foi baseada nos dados coletados pelo sensor 002 do ano de 2019 da Av. Agamenon Magalhães na Cidade do Recife sentido Cidade de Olinda, ambas localizadas em Pernambuco, Brasil. A capacidade na granularidade dinâmica para refinar a busca de hiperparâmetros com base nas observações anteriores e a segmentação dos dados por velocidades forneceram uma base de suporte sobre o dimensionamento do ambiente de simulação e alguns parâmetros relacionados a infraestrutura viária e de telecomunicação, além da delimitação das quantidades de veículos a serem simulados (100,200 e 300).

Com referência ao dimensionamento das infraestruturas de comunicação representadas na Figura 24, os dados foram obtidos por meio das informações disponibilizadas pelo site Conexis Brasil Digital [3], levando em consideração a disponibilidade mínima de 2 antenas e máxima de 5 antenas na área por operadora (2-OI, 3-TIM, 3-CLARO e 5-VIVO).

[3] https://conexis.org.br/numeros/mapa-de-antenas/

Tabela 9 – DataSet - Av. Agamenon Magalhães Recife PE / 2019

Mês	Total Veículos/Mês	Média por Minuto	Máximo por Minuto
Dezembro	1.485.711,00	32,12	233
Novembro	1.250.446,00	26,11	147
Outubro	2.122.336,00	37,81	256
Setembro	1.179.644,00	29,10	182
Agosto	1.611.225,00	34,78	201
Julho	1.854.501,00	35,87	216
Junho	2.264.663,00	39,05	283
Maio	2.104.278,00	36,74	212
Abril	1.743.255,00	34,83	185
Março	1.474.554,00	26,72	210
Fevereiro	1.923.687,00	36,11	244
Janeiro	2.246.986,00	38,19	258

Fonte: O autor (2023)

No entanto, devido à dificuldade de encontrar um conjunto de dados (*dataset*) que atendesse aos objetivos desta pesquisa, foi gerado um conjunto de dados para o treinamento da IA utilizando todo o ambiente integrado de simulação. A primeira etapa envolveu a determinação do deslocamento dos veículos, que necessitam percorrer uma região específica da cidade, apresentada na Figura 24, com métricas de rede capturadas a cada 200 milissegundos ao longo de 250 segundos de simulação.

Figura 24 – Região de Deslocamento dos Veículos - Av. Agamenon Magalhães, Recife-PE Brasil

Fonte: O autor / SUMO (2023)

4.6 CENÁRIOS DE CASOS DE USO EM PROVEDORES IOV

No intuito de ilustrar funcionalidade e a flexibilidade do sistema SIMA-IoV, podemos considerar um cenário de caso de uso em que uma empresa de transporte urbano adota duas soluções de autenticação para os veículos conectados: uma solução centralizada e outra descentralizada. Ambas as soluções estão operando em tempo real e são capazes de fornecer autenticação para os veículos na rede de Internet of Vehicles (IoV). O SIMA-IoV, que é um mecanismo inteligente de gerenciamento de autenticação, tem a capacidade de analisar continuamente os parâmetros do ambiente real — como o número de veículos, a composição do tráfego, a velocidade média, entre outros dados coletados em tempo real. Com base nesses dados, o SIMA-IoV consegue identificar qual método de autenticação (centralizada ou descentralizada) é o mais eficiente e adequado para o cenário específico em questão.

Por exemplo, em uma situação de alto tráfego, com muitos veículos conectados, o método de autenticação centralizada pode ser mais rápido e eficaz para garantir uma resposta ágil. Por outro lado, em um cenário com menos veículos ou onde a latência não é um fator crítico,

a autenticação descentralizada pode ser preferível, oferecendo maior segurança e resiliência a ataques. Ao selecionar automaticamente o método de autenticação mais apropriado para cada situação, o SIMA-IoV proporciona aos provedores de serviço uma solução dinâmica e adaptável. Dessa forma, os provedores podem utilizar a opção de autenticação que melhor atende às necessidades e características do ambiente em tempo real, garantindo eficiência operacional, segurança e flexibilidade na gestão da rede de veículos conectados.

4.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo 4 apresentou o SIMA-IoV, projetado para selecionar o método de autenticação mais apropriado para diferentes cenários em um ambiente IoV. O mecanismo da tese emprega a otimização bayesiana para selecionar o melhor método de autenticação com base nas informações dos parâmetros coletados no contexto do ambiente. Isso possibilita uma utilização mais eficiente dos recursos disponíveis para cada localidade examinada.

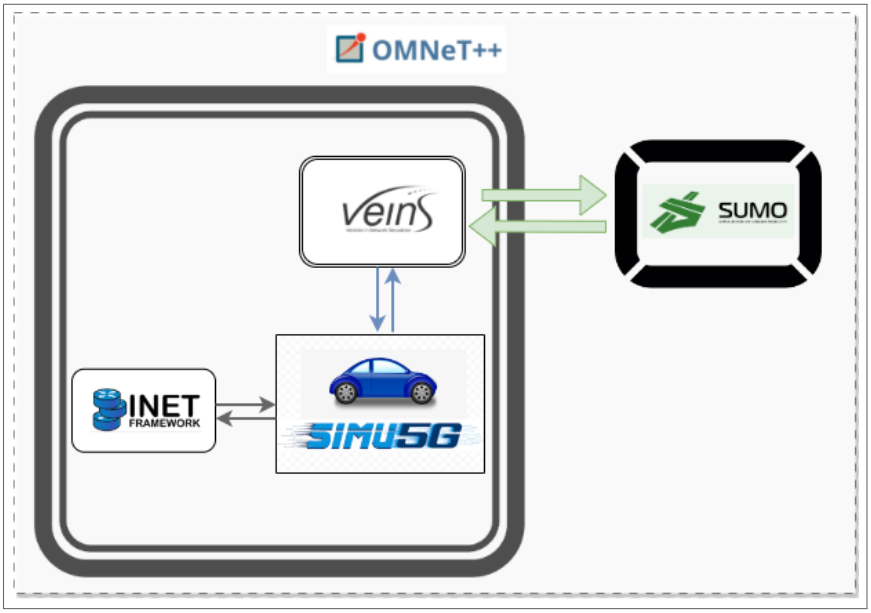
5 CAPÍTULO AMBIENTES DE SIMULAÇÃO E TREINAMENTO

Este capítulo 5, detalha os ambientes de simulação e treinamento desta pesquisa, destacando as ferramentas e suas integrações, bem como, as definições das escolhas dos modelos e parâmetros que impactam os resultados.

5.1 AMBIENTES DE SIMULAÇÃO

Para o desenvolvimento da tese, foram integradas diversas ferramentas, com o objetivo de representar o ambiente IoV e obter resultados próximos aos de um ambiente real. A Figura 25 ilustra a arquitetura dos módulos que compõem o ecossistema de tecnologias utilizadas para realizar e executar as simulações de forma integrada.

Figura 25 – Ferramentas do Ambiente Simulação



Fonte: O autor (2023)

5.1.1 OMNeT++

É um ambiente integrado que fornece um *framework* estruturado de simulação e integração de tecnologias baseado na linguagem C++, apoiada por componentes para a construção de simuladores de rede de comunicação com e sem fio. A plataforma é baseada na IDE de desenvolvimento Eclipse e a estende com novos editores, assistentes e funcionalidades

adicionais como suporte para modelagem de desempenho em redes fotônicas, sem fio, sensores, internet das coisas, redes veiculares terrestres ou aéreas.

Como principais componentes da ferramenta podemos mencionar, biblioteca do kernel de simulação (C++), linguagem de descrição da topologia NED IDE de simulação baseado no Eclipse, GUI de tempo de execução de simulação interativa (Qtenv), interface de linha de comando para execução de simulação (Cmdenv), utilitários adicionais (ferramenta de criação de *makefile*), integração ao GIT e outros recursos para modelagem UML (OMNETPP, 2023a).

5.1.2 INET Framework

É uma biblioteca de modelo de código aberto para o ambiente de simulação dos protocolos de redes. Foi construído em torno do conceito de módulos que se comunicam por passagem de mensagem. Agentes e protocolos de rede são representados por componentes, que podem ser livremente combinados para formar *hosts, bridges, routers, switches* e outros dispositivos de rede. O INET Framework fornece protocolos, agentes e outros modelos para as redes de comunicação.

Suas principais funcionalidades são implementadas na camada OSI (física, camada de enlace, rede, transporte, aplicação), pilha de rede IPv4/IPv6, protocolos da camada de transporte: TCP, UDP, SCTP, protocolos de roteamento (ad-hoc e com fio), interfaces de redes com/sem fio (Ethernet, PPP, IEEE 802.11, etc.), camada física com nível de detalhe escalonável (unidade de rádio de disco para modelos de propagação detalhados, nível de quadro para representação de nível de bit/símbolo, etc.), redes overlay/peer-to-peer, LTE/5G e suporte a emulação e mobilidade de rede (OMNETPP, 2023b).

5.1.3 Simu5G

O Simu5G é uma biblioteca que permite a simulação de comunicações no plano de dados no padrão de implantação de quinta geração "5G New Radio". A biblioteca do Simu5G é modelada e baseada no OMNeT++, implementando todas as camadas do protocolo, bem como suas funcionalidades e seus requisitos com base no 3GPP (NARDINI DARIO SABELLA; VIRDIS, 2020).

O Simu5G pode ser usado para avaliar os serviços de Computação de Borda Multi Serviços (MEC) e redes veiculares terrestres e aéreas, internet das coisas, cidades inteligentes,

entre outras aplicações que utilizam como meio de comunicação em uma rede 5G. A biblioteca permite também a coexistência e transição dos padrões de comunicação 4G-LTE/5G, bem como a modelagem de cenários comuns de comunicação cliente servidor até os modelos complexos fim-a-fim envolvendo computação de borda e comunicação multiacesso (NARDINI GIOVANNI STEA; SABELLA, 2020).

5.1.4 Veins

Simulador de redes para ambientes veiculares possui uma estrutura abrangente de modelos, buscando apresentar um espaço de simulação totalmente adequado com relação aos requisitos da alta mobilidade, densidade, desempenho e tratamento a falhas de entregas e segurança na rede de comunicação. Assim como o INET o Veins se integra ao OMNeT++ através da IDE do Eclipse para realizar interações de chamadas remotas ao simulador de tráfego urbano e rodoviário SUMO 5.1.5 utilizando a porta 9999 e o protocolo TCP. O Veins foi concebido de um projeto de pesquisa de mesmo nome que hoje abrange 5 continentes e envolve universidades, institutos de pesquisa, órgãos governamentais e a indústria automobilística (SOMMER; DRESSLER, 2011).

Para modelagem das interações complexas das redes veiculares todas as atividades relacionadas ao tráfego são realizadas pelo SUMO, já a parte da simulação de rede e camada física, tais como modelos de rádio fica a cargo do OMNeT++/INET, restando ao simulador Veins a modelagem da construção específica dos domínios dos dispositivos para as redes veiculares fornecendo uma estrutura adequada e, ao mesmo tempo, abrangente para ser aplicada nas simulações realizadas (VEINS.ORG, 2023).

5.1.5 SUMO

O SUMO é um simulador de tráfego que admite diversos tipos de modelagens intermodais de mobilidade, sejam elas de tráfego de pedestres, veículos particulares, de transporte público, automotores ou não, veículos aéreos tripulados ou não, em áreas urbanas ou rurais. A primeira versão do simulador foi disponibilizada em 2001, sendo de código aberto e em constante crescimento das funcionalidades e ferramentas de apoio a simulação com objetivo de melhor suportar a criação, execução e avaliações dos diferentes tipos de ambientes e contextos de tráfego.

O simulador fornece modelos básicos, aprimorados e personalizados implementados através de suas APIs como forma de controlar e melhorar todo o processo da simulação móvel. Os componentes do SUMO podem ser utilizados em vários tipos de projetos e para diversas questões de pesquisa, tais como: avaliação do desempenho de dispositivos de trânsito, escolha de rotas, avaliação de aspectos ecológicos, algoritmos de decisão entre outros. Como exemplo, o SUMO foi utilizado para fornecer previsões de tráfego na cidade de Colônia durante uma visita do Papa em 2005 e também para a análise do fluxo coletivo na Copa do Mundo de Futebol em 2006.

Ele foi empregado para compreender o comportamento simulado de telefonia veicular para avaliação do desempenho da vigilância baseada em redes sem fio de longas distâncias. Este tipo de simulador é largamente utilizado pela comunidade da Internet das Coisas e Rede Veiculares no intuito de compreender comportamentos relacionados a mobilidade e suas demandas (SUMO, 2023).

5.2 ETAPA DE TREINAMENTO

A ideia é que o mecanismo seja capaz de interpretar os dados e métricas da rede e obtenha desta forma uma melhora na análise das mudanças dos dados dos parâmetros no qual a rede está em simulação. Para isso, o mecanismo de análise inicial do ambiente obtém o estado, enviando para o SIMA-IoV, no formato apresentado pelo pacote JSON no Código Fonte 1.

Código Fonte 1 – Pacote enviado pelo mecanismo de análise para o algoritmo de otimização.

```
{
  "Tempo de Autenticacao": value,
  "Consumo do Link": value,
  "Atraso Medio": value,
  "Numero de carros": value
}
```

Na etapa de treinamento, foi utilizado cenários distintos que variavam a solução de autenticação, a quantidade de dispositivos e a infraestrutura de comunicação. Cada cenário incluía diferentes configurações de *gNodeBs* estações base de quinta geração e diversas quantidades de veículos (dispositivos OBUs). Especificamente, os cenários variavam entre 2, 4 e 6 *gNodeBs*, e de 100 a 1000 veículos. Para cada combinação dessas configurações, foram alternados os métodos de autenticação BC (PoS/PoW) e TA. A definição desses cenários ocorria dentro da Função Objetivo apresentada na seção anterior. O Algoritmo 3 apresenta o funcionamento

das execuções das simulações. A execução é baseada no comando gerado pelo trial da função objetivo, onde são enviados para o Omnet++ a especificação das configurações e a simulação é inicializada. Uma vez que a simulação chega ao fim, o estado da rede para aquele trial é armazenado em um banco de dados *(SQLite)* e retornado para a função objetivo, para poder ser utilizado no processo de otimização contínua.

Algoritmo 3: Execução do Ambiente

Input: Comando (Solução, Número de Carros, Número de Infraestruturas), DataBase
Output: Métricas de Desempenho (Tempo de Autenticação, Consumo de Link, Atraso Médio, Número de Carros)
Function `ExecuçãoDoAmbiente (Comando)`**:**
    $estado = run(Comando)$;
    $AtualizarBasedeDados(estado)$;
    return $estado$;

Fonte: O autor (2023)

Cada registro no banco de dados contém os seguintes campos, *Solution*: Tipo do método de autenticação aplicada na simulação; *NumInfra*: Número de gNodeBs utilizados no cenário; *NumCars*: Número de veículos presentes em cada cenário simulado; *AuthTime*: Tempo médio de autenticação de cada método em milissegundos; *Throughput*: Taxa de transferência média em Kbps (Kbits por segundo); *Delay*: Atraso médio em milissegundos;

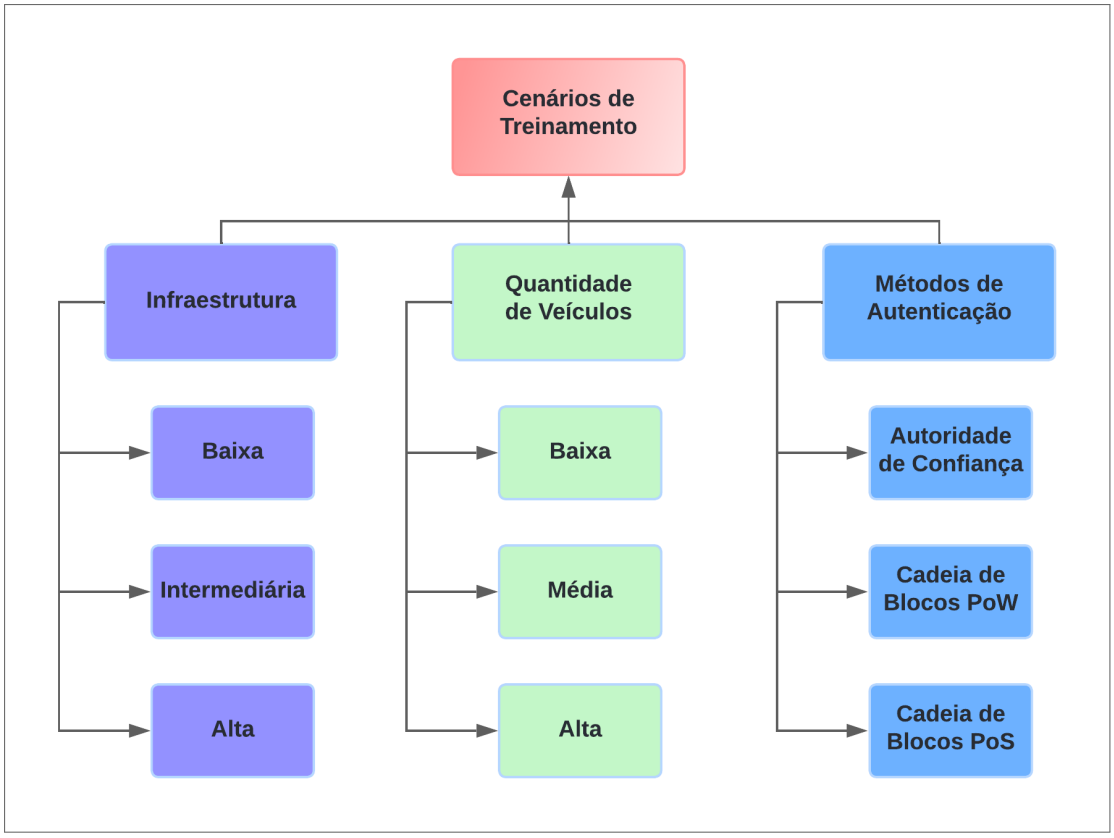
É importante mencionar que para os cenários de treinamentos propostos foram elaboradas três possibilidades de configuração, conforme apresentado na Figura 26:

Tabela 10 – Descrição da Estrutura do Banco de Dados

Campo	**Descrição**
Solution	Tipo de autenticação aplicada na simulação.
NumInfra	Número de gNodeBs utilizados no cenário.
NumCars	Número de veículos presentes no cenário.
AuthTime	Tempo médio de autenticação em milissegundos.
Throughput	Vazão média em Mbps (kbps por segundo).
Delay	Atraso médio em milissegundos.

Fonte: A autora (2024)

Figura 26 – Cenários de Treinamentos



Fonte: O autor (2023)

A Tabela 10 apresenta a descrição da estrutura do banco de dados utilizado para etapa de treinamento desta pesquisa. O campo "Solution"especifica o tipo de solução de autenticação aplicada na simulação. O campo "NumInfra"indica o número de gNodeBs (estações rádio base) utilizados no cenário. "NumCars"refere-se ao número de veículos presentes no cenário. O campo "AuthTime"mede o tempo médio de autenticação em milissegundos. O "Throughput"registra a vazão média em *kilobits* por segundo (Kbps), enquanto "Delay"representa o atraso médio em milissegundos.

5.3 IMPLEMENTAÇÃO DOS MÉTODOS DE AUTENTICAÇÃO

Este capítulo 5.3, apresenta a implementação das soluções de autenticação, Autoridade de Confiança e *Blockchain*. Os métodos foram desenvolvidos na linguagem de programação *c++*(v17), suas execuções e comportamentos simulados nas ferramentas de simulação ns-3 e OMNeT++. Para atender aos requisitos de proteção, utilizamos a biblioteca *crypto++* (v5.6.4). Em relação ao protocolo de transporte, optamos pelo TCP (*Transmission Control*

Protocol) (POORZARE; AUGé, 2020), devido às frequentes perdas de pacotes contendo dados sensíveis, como chaves criptográficas, ao tentamos utilizar o UDP (*User Datagram Protocol*). Essas perdas comprometem a integridade das chaves durante o processo de estabelecimento e validação de dispositivos no ambiente IoV, resultando em impactos negativos no desempenho e na operação dos serviços de autenticação e validação de mensagens em ambos os métodos (TA/BC).

5.3.1 Esquemas de Proteção dos Métodos TA e BC

Para o atendimento das funcionalidades de proteção e validação dos dados nos métodos desenvolvidos, TA e BC, utilizamos especificamente os algoritmos criptográficos assimétricos (*Elliptic Curve Diffie–Hellman Key Exchange-ECDH, Elliptic Curve Digital Signature Algorithm-ECDSA*), simétrico (*Advanced Encryption Standard-AES*) e de *hash*(*Secure Hash Algorithm-SHA*). O Algoritmo 4 apresenta um trecho de exemplo da função para geração das chaves públicas e privadas utilizando o esquema de curvas elípticas.

Algoritmo 4: Geração das Chaves *ECDH*

```
Data: Elliptic Curve Domain Parameters, Random Number Generator
Result: Public and Private Key Pair, Public Key of Party B
Function EllipticCurveDiffieHellman():
    Initialize curve parameters;
    Initialize RNG;
    Initialize ECDH domain with curve parameters;
Function GenerateKeyPair():
    Allocate memory for private and public keys;
    Generate key pair using ECDH domain and RNG;
    Store private key in m_privkey;
    Store public key in m_pubkey;
    return m_privkey,m_pubkey;
Function SetPubKeyB(pubB):
    Store pubB in m_pubkeyB;
```

Fonte: O autor (2024).

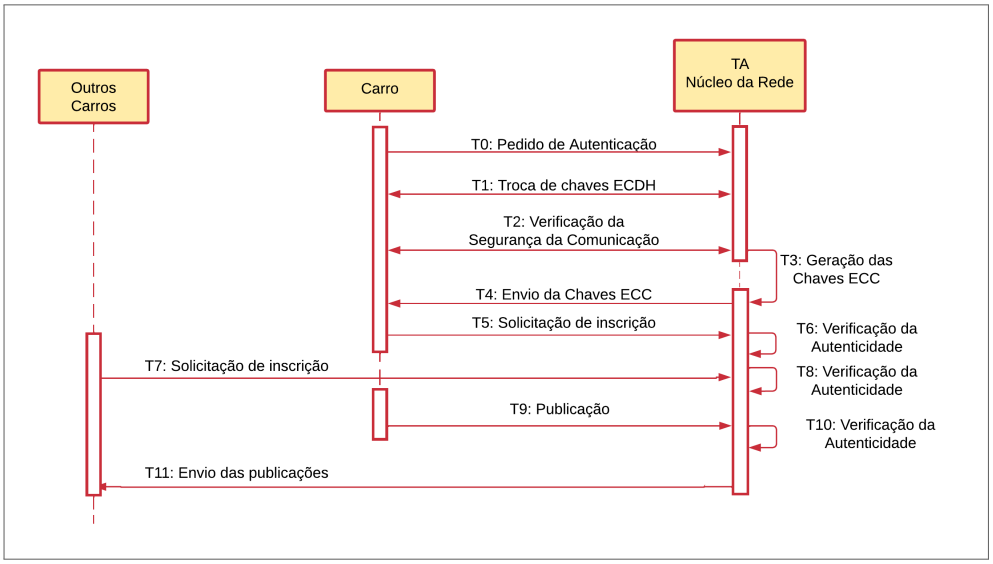
O algoritmo assimétrico ECDH foi utilizado para troca das chaves privadas através do canal inseguro da IoV, possibilitando a proteção da chave que será enviada ao veículo. O algoritmo ECDSA tem as funções da autenticidade e do não repúdio das mensagens dos

veículos remetidas à rede. O algoritmo AES, simétrico, é responsável por manter a privacidade e confidencialidade dos dados presentes nas mensagens dos dispositivos no ambiente da IoV. A função de condensação SHA512 possui a atribuição da geração dos códigos de verificação da integridade das chaves na TA e dos blocos encadeados na BC.

5.3.2 Autoridade de Confiança (TA)

A elaboração do método da TA levou em consideração o fluxo de dados apresentado na Figura 27, as especificidades da tecnologia, modo de implementação e suas adaptações para o ambiente simulado. Suas principais funções implementadas são a autenticação, gerenciamento dos usuários, geração e revogação das chaves e validação das mensagens publicadas no ambiente da IoV. A Autoridade de Confiança, desenvolvida para esta tese, opera no modelo de arquitetura Cliente/Servidor (LIANHAI et al., 2019), estando localizada na camada do núcleo da rede do ambiente da IoV. Optamos pela gestão de certificados locais, não terceirizados, realizando as atividades da Autoridade Certificadora(CA) no próprio método da TA. A Figura 27 apresenta a ordem de comunicação para atividade de autorização e permissão de uso da rede e dos serviços e sistemas entre o veículo integrante e a TA.

Figura 27 – Diagrama de Sequência (TA) - *CORE/EDGE/FOG*



Fonte: O autor (2020)

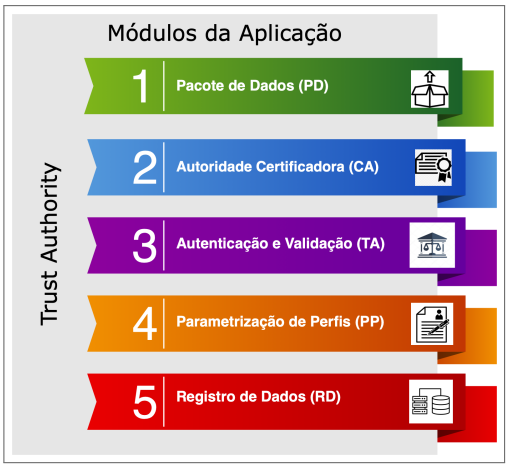
O diagrama 27, mostra a sequência de mensagens trocadas entre o carro e a TA (Núcleo da Rede), com foco na autenticação, troca de chaves, inscrição e publicação de informações. No instante T0 ocorre o Pedido de Autenticação, onde o carro envia um pedido de autenticação a

TA. Em T1 é iniciada a Troca das chaves ECDH (Elliptic-curve Diffie–Hellman) para proteção futura das chaves ECC. T2, faz a Verificação da Segurança da Comunicação, onde o veículo verifica a segurança da comunicação após a troca de chaves. T3, realiza a Geração das Chaves ECC (Elliptic Curve Cryptography). T4, Envia as Chaves ECC geradas para o carro. T5, efetua a Solicitação de inscrição a TA. T6, executa a Verificação da Autenticidade do pedido de inscrição do carro. T7, procede a Solicitação de inscrição. T8, confere a Verificação da Autenticidade dos carros que solicitaram inscrição. T9, realiza a Publicação. T10, verifica a Autenticidade da Publicação e T11 efetiva o Envio das publicações para outros carros.

5.3.2.1 Estruturação da TA

Para o devido funcionamento do método a TA desenvolvida foi modularizada para possibilitar uma melhor conformidade de implementação ao ambiente IoV , conforme Figura 28.

Figura 28 – Módulos da Aplicação - *Trust Authority* (TA)



Fonte: O autor (2020)

1 (PD) Pacote de Dados - Estrutura os tipos de pacotes: autenticação, certificados, transações e mensagens. Define os formatos e campos de cada tipo de pacote de dados e sinalização;

2 (AC) Autoridade Certificadora - Determina e gerencia os certificados e as chaves de proteção, determinando os padrões e tecnologias criptográficas bem como suas aplicabilidades e uso;

3 (TA) Autenticação e Validação - Obtém as requisições de autenticação e realiza a validação dos entes e suas mensagens na rede ou sistema através das consultas realizadas a CA;

4 (PP) Parametrização de Perfis - Caracteriza os perfis e os relaciona com as devidas autorizações, permissões e atividades;

5 (RB) Registro de Dados - Grava de forma estruturada as informações no sistema de gerenciado de banco de dados da TA.

Um exemplo da *Trust Authority*(TA) desenvolvida é exibida no Algoritmo 5.

Algoritmo 5: Aplicação *Trust Authority* IoV

```
Data: Node configuration, Socket type, Port number
Result: Application startup with appropriate callbacks and bindings
Function TrustAuthorityApplication():
    recvSocket = CreateSocket ();
    addr = Node.GetAddress ();
    local = InetSocketAddress(addr, port);
    if recvSocket.Bind(local) == -1 then
        Fatal error "Fail to bind socket";
    recvSocket.Listen();
    recvSocket.SetRecvCallback(TrustAuthorityApplication::Receive);
    recvSocket.SetRecvPktInfo(true);
    recvSocket.SetAcceptCallback( MakeNullCallback(bool, Ptr<Socket>, const
     Address), MakeCallback(TrustAuthorityApplication::HandleAccept, this));
    recvSocket.SetCloseCallback(
     MakeCallback(TrustAuthorityApplication::HandleClose, this),
     MakeCallback(TrustAuthorityApplication::HandlePeerError, this));
    if isCar then
        Schedule(interval * nodeId,
          TrustAuthorityApplication::SendAuthenticationRequest, this);
```

Fonte: O autor (2024).

5.3.3 *Blockchain* (BC)

A aplicação da *Blockchain* desenvolvida levou em consideração todo o ambiente descrito na seção 2, suas funções foram baseadas em pesquisas similares publicadas (HU et al., 2019; GUO et al., 2019b), levando em consideração as diferenças do ambiente e as características

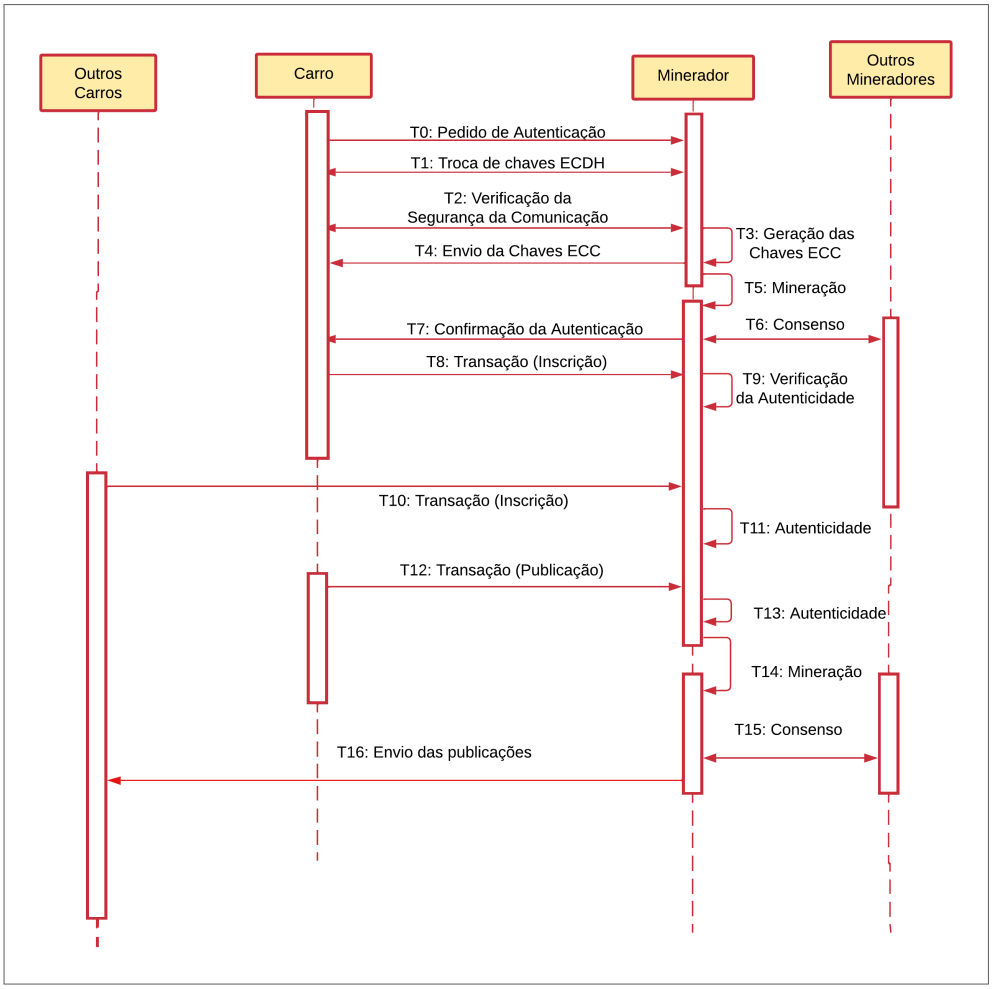
foco da arquitetura IoV desta tese. A *Blockchain* opera na arquitetura distribuída (ZHANG; LI; CUI, 2018a), seu desenvolvimento teve como base as propriedades de um sistema autônomo, colaborativo, processamento compartilhado, imutabilidade de registros e o consenso das transações. A aplicação foi projetada para funcionar na camada do nevoeiro da IoV onde as RSU's atuam como mineradores e decisores da *Blockchain*. O mecanismo desta tese implementa dois algoritmos de consenso: PoW e *Proof of Participation* (PoS), em um sistema de arquitetura *peer-to-peer*. Esses algoritmos são usados para tomada de decisões compartilhadas e participativas e na validação das transações. As próximas subseções apresentam o funcionamento dos métodos PoW e PoS desenvolvidos nesta pesquisa.

5.3.3.1 Prova de Trabalho - PoW

O mecanismo de consenso PoW (*Proof of Work*), ou Prova de Trabalho, foi implementado na *Blockchain* da criptomoeda *Bitcoin* para validar transações e adicionar novos blocos à cadeia. Nesta técnica, os nós da rede competem para resolver complexos problemas matemáticos, conhecidos como *hashes* de validação, para verificar e registrar as transações. A Figura 29, ilustra a ordem de comunicação para autenticação e permissão entre o veículo e a *Blockchain* utilizando o algoritmo de consenso PoW.

O diagrama 29, apresenta a sequência de mensagens trocadas entre o carro, o minerador PoW e outros mineradores localizados no nevoeiro (*gNodebs*), com foco na autenticação, troca de chaves, inscrição, publicação, mineração e consenso. Em T0, o veículo envia um Pedido de Autenticação ao minerador. T1, o minerador responde ao carro e ocorre a troca de chaves ECDH. T2, verifica a segurança da comunicação após a troca de chaves. T3, o minerador gera as chaves ECC. T4, o minerador envia as chaves ECC geradas para o carro. T5, realiza o processo de mineração. T6, outros mineradores participam do consenso para validar a mineração. T7, o minerador confirma a autenticação ao carro. T8, realiza uma transação de inscrição. T9, o minerador verifica a autenticidade da transação de inscrição. T10, outros carros realizam transações de inscrição. T11, os mineradores verificam a autenticidade das transações. T12, realiza uma transação de publicação. T13, os mineradores verificam a autenticidade das transações de publicação. T14, os mineradores realizam o processo de mineração para as novas transações. T15, os outros mineradores participam do consenso para validar as novas transações. T16, o veículo envia as publicações para os outros carros.

Figura 29 – Diagrama de Sequência (BC/PoW)



Fonte: O autor (2021)

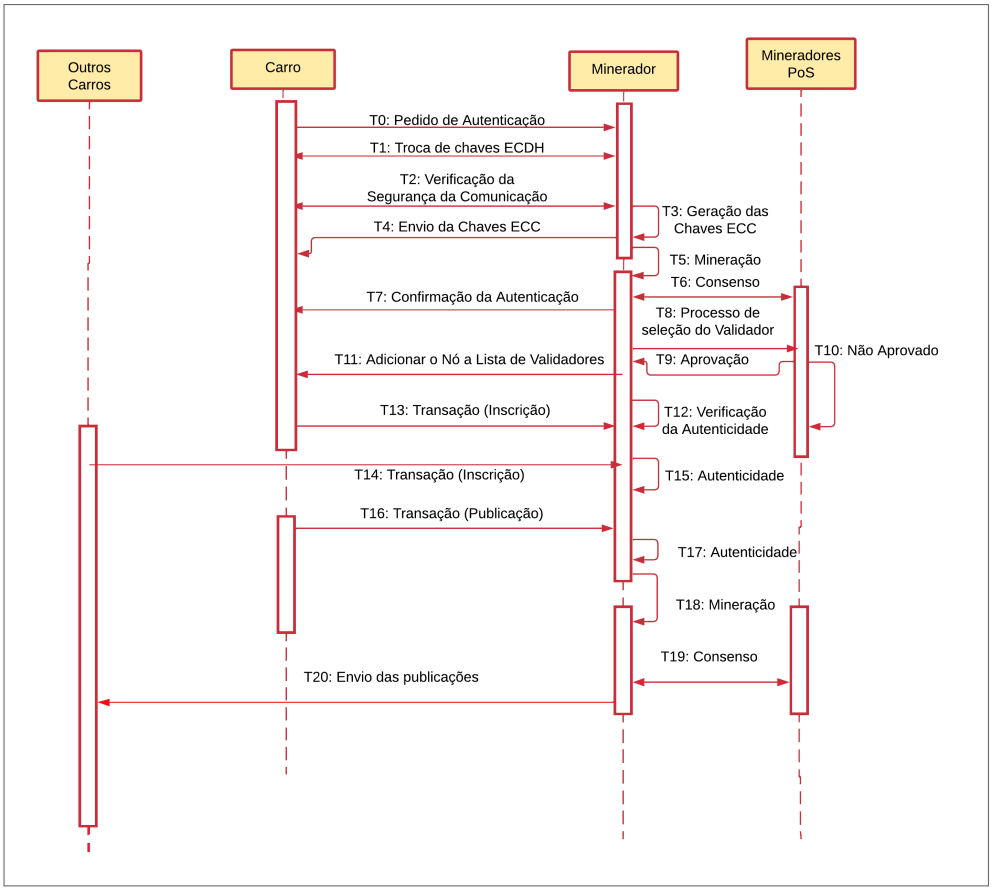
5.3.3.2 Prova de Participação - PoS

O PoS é considerado mais eficiente energeticamente e oferece melhor desempenho e escalabilidade, pois não requer o poder computacional necessário para resolver desafios matemáticos complexos, conforme mencionado no 2. O método de autenticação *Blockchain* utilizando consenso PoS tende a ser mais eficiente e escalável (YAGA PETER MELL; SCARFONE, 2011). A Figura 30 apresenta a ordem de comunicação para atividade de autenticação entre o veículo e a BC utilizando o algoritmo de consenso PoS na IoV.

O diagrama 30, exibi a sequência de mensagens trocadas entre o carro, o minerador PoS localizado no nevoeiro (*gNodebs*), com foco na autenticação, troca de chaves, inscrição, publicação, mineração, seleção de validadores, aprovação e consenso.

Todo o processo é iniciado também no T0 com um pedido de autenticação ao minerador. Em T1, o minerador responde ao carro e ocorre a troca de chaves ECDH. T2, verifica a

Figura 30 – Diagrama de Sequência (BC/PoS)



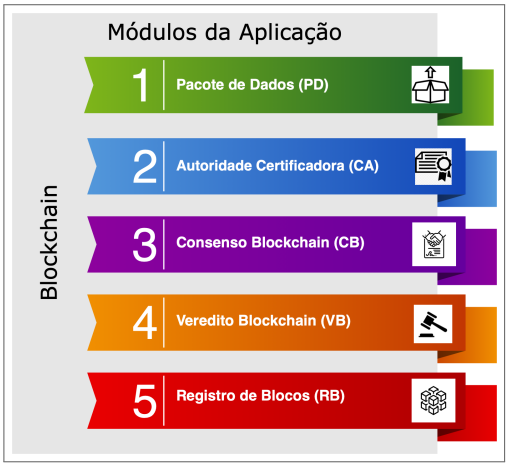
Fonte: O autor (2023)

segurança da comunicação após a troca de chaves. T3, o minerador gera as chaves ECC. T4, envia as chaves ECC geradas para o carro. T5, o minerador realiza o processo de mineração. T6, os mineradores participam do consenso para validar a mineração. T7, confirma a autenticação ao carro. T8, inicia o processo de seleção do validador. T9, o validador aprova a transação. T10, caso o validador não aprove, a transação não é descartada. T11, o veículo é adicionado à lista de validadores. T12, verifica a autenticidade da transação. T13, o veículo realiza uma transação de inscrição. T14, outros veículos realizam transações de inscrição. T15, os mineradores verificam a autenticidade das transações de inscrição. T16, realiza uma transação de publicação. T17, os mineradores verificam a autenticidade das transações de publicação. T18, realiza o processo de mineração para as novas transações. T19, ocorre o consenso para validar as novas transações. T20, envia as publicações para os outros carros.

5.3.3.3 Estruturação da BC

Para o devido funcionamento a BC foi desenvolvida modularizada possibilitando um ajustamento a arquitetura IoV da tese, conforme apresentado na Figura 31.

Figura 31 – Módulos da Aplicação - *Blockchain* (BC)



Fonte: O autor (2022)

1 (PD) Pacote de Dados - Estrutura os tipos de pacotes: autenticação, certificados, transações de bloco e mensagens de alerta. Define os formatos e campos de cada tipo dos pacotes de dados;

2 (AC) Autoridade Certificadora - Gera e administra os certificados e as chaves de proteção, determinando os padrões das tecnologias criptográficas bem como suas aplicabilidades e uso;

3 (CB) Consenso *Blockchain* - Estabelece os critérios e regras para as decisões participativas e compartilhadas sobre as transações publicadas na rede;

4 (VB) Veredito *Blockchain* - Delibera sobre as decisões obtidas através do consenso para aprovação de um novo bloco a ser registrado no ambiente da IoV;

5 (RB) Registro do Bloco - Inclui conjunto de registros propostos e aprovados como novo bloco na cadeia e replicados aos demais nós participantes do sistema distribuído.

Um exemplo do método de autenticação distribuído *Blockchain* desenvolvido é exibida no Algoritmo 6.

Fonte: O autor (2023).

Algoritmo 6: *Blockchain* IoV

Data: Node configuration, Socket type, Port number
Result: Application startup with appropriate callbacks, bindings, and initial blockchain block
Function `BlockchainApplication()`:

```
recvSocket = CreateSocket ();
addr = Node.GetAddress ();
local = InetSocketAddress(addr, port);
if recvSocket.Bind(local) == -1 then
    Fatal error "Fail to bind socket";
recvSocket.Listen();
recvSocket.SetRecvCallback(BlockchainApplication::Receive);
recvSocket.SetRecvPktInfo(true);
recvSocket.SetAcceptCallback( MakeNullCallback(bool, Ptr<Socket>, const
 Address), MakeCallback(BlockchainApplication::HandleAccept, this));
recvSocket.SetCloseCallbacks(
 MakeCallback(BlockchainApplication::HandlePeerClose, this),
 MakeCallback(BlockchainApplication::HandlePeerError, this));
Block block = Block(1, 0, currentTransactions, 0, 0, 0, "0000");
block.hash = "0000";
chain.pushback(block);
rand = CreateObject(UniformRandomVariable);
randomOffset = MicroSeconds(rand->GetValue(100, 2000));
ECDHKey.KeyPairGeneration();
ECDHIv.KeyPairGeneration();
```

5.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

O capítulo 5, exemplificou o ambiente de simulação para a avaliação de diferentes cenários e métodos de autenticação na Internet dos Veículos (IoV) em redes 5G. A criação de ambientes de simulação permitiu a replicação das condições reais, viabilizando uma análise precisa dos métodos de autenticação e do mecanismo concebido nesta tese. O treinamento das simulações destacou a necessidade de um processo sistemático para garantir a validade e a confiabilidade dos resultados obtidos. A implementação dos métodos de autenticação aplicáveis no contexto da Internet dos Veículos (IoV) em redes 5G para compreender as variações nos resultados entre ambientes é crucial para a adaptação e otimização dos métodos de autenticação.

6 CAPÍTULO DE AVALIAÇÃO DE DESEMPENHO

Este capítulo apresenta os resultados dos ambientes de simulação (preliminares e definitivos), onde os experimentos foram realizados. Os resultados são apresentados, bem como suas evoluções e mudanças. Estes resultados contemplam todo o processo para definição do SIMA-IoV, nos ambientes de simulação e de codificação utilizados.

6.1 AVALIAÇÃO DOS AMBIENTES DE SIMULAÇÃO

6.1.1 Network Simulator-3 (BCxTA)

Inicialmente os métodos de autenticação, TA e BC foram desenvolvidos no ambiente de simulação IoV implementado no ns-3. Os resultados preliminares obtidos foram objeto da publicação (QUEIROZ et al., 2023) e serão apresentados nesta subseção como análise preliminar do comportamento dos métodos sem o emprego da técnica de inteligência artificial como suporte a decisão.

Devido a limitações identificadas, oriundas do simulador ns-3 (mobilidade, controle e mapeamento de áreas) bem como das bibliotecas de comunicação 5GLena (*handover entre as gNodeBs*) e das dificuldades de integração com demais simuladores, os resultados preliminares apresentados serviram de base para uma melhor compreensão dos métodos de autenticação e para a evolução da pesquisa em um novo ambiente de simulação definido posteriormente.

A Tabela 11 exibe métricas, cenários e configurações de hardware e software definidos para as simulações. O tempo de simulação foi de 100s (linha1) e foram realizadas simulações considerando o parâmetro Quantidade de Veículos variando entre 20 e 100 unidades, conforme indicado na linha 2. A linha 3 exibe a tecnologia adotada 5G/NSA. O tamanho de pacote foi configurado com o valor 1.496 bytes sendo apresentado na linha 4.

Em relação ao número de gNBs, foram utilizadas 4 unidades (linha 5). Os veículos permanecem sob a mesma cobertura do início ao fim da simulação, portanto neste estudo não foi implementada mobilidade para os veículos. As configurações de *hardware* e *software* (linha 6). Os esquemas criptográficos simétricos, assimétricos e de *hash* utilizados foram indicados na linha 7. As soluções de segurança desenvolvidas, implementadas e analisadas, Autoridade de Confiança e *Blockchain* (linha 8). Por fim as métricas são apresentadas na linha 9 da tabela.

Tabela 11 – Bases da Simulação

1	Tempo de Simulação	100 segundos
2	Quantidade de Veículos	20, 40, 60, 80, 100
3	Tecnologia de Comunicação	5G *Non Standalone*
4	Tamanho dos Pacotes	1496 *Bytes*
5	Número de *gNodeB's*	4 (*mmWave*)
6	Servidores das Aplicações	Arquitetura Baseada em CPU - Intel Xeon CPU E5530 2.40GHz, 12 vCPU, 38GB RAM, HD 80GB e Ubuntu 20.08
7	Esquemas Criptográficos	ECDH, ECDSA, AES256, SHA512
8	Aplicações Desenvolvidas	Autoridade de Confiança, *Blockchain*
9	Métricas	Vazão, Atraso, Tempo de autenticação, Número de mensagens

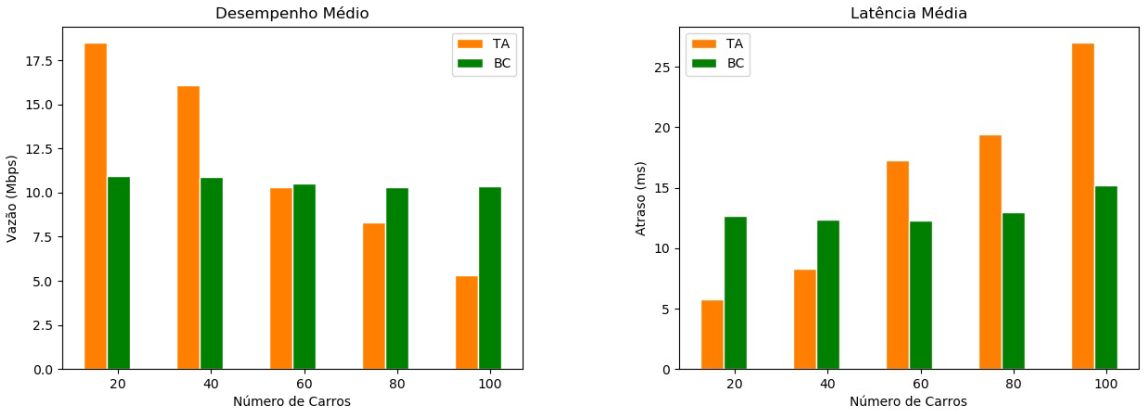
Fonte: O autor (2022)

6.1.1.1 Avaliação Comparativa

Explicitamos os resultados das simulações executadas no ambiente da IoV na borda em redes 5G, conforme parâmetros indicados na Tabela 11. Os dados apresentados são referentes às análises nos diferentes cenários de carga no ambiente desta tese. Nas Figuras 32a e 32b realizamos a avaliação da comunicação centralizada que será utilizada para a aplicação da TA (na cor laranja) no núcleo da rede e a comunicação distribuída na borda que será utilizada pela aplicação da *Blockchain* (na cor verde).

A Figura 32a apresenta a vazão da rede 5G para as arquiteturas centralizada e distribuída, variação no número de veículos entre 20 e 100, intercalações de 20 unidades e envios de pacotes entre 10mil a 52mil por cenário de simulação. A arquitetura centralizada alcançou vazões iguais a 18,74Mbps e 5,29Mbps para os cenários com 20 e 100 veículos, respectivamente. Em relação à mesma simulação na arquitetura descentralizada obtivemos taxas de desempenho entre 10,96Mbps(20 veículos) a 10,29Mbps(100 veículos). Levando em conta o mesmo número de pacotes já citados, podemos observar na Figura 32b, que na arquitetura centralizada os atrasos variaram entre 5,79ms e 26,97ms, ocorrendo degradação do atraso à medida que o número de veículos aumenta, culminando em um valor de atraso bastante elevado quando o número máximo de veículos é simulado. A arquitetura descentralizada, novamente se manteve

Figura 32 – Ambiente Centralizado x Descentralizado



(a) Vazão Cent vs Desc

(b) Atraso Cent vs Desc

Fonte: O autor (2022)

estável, com variação entre 12,26ms à 15,18ms.

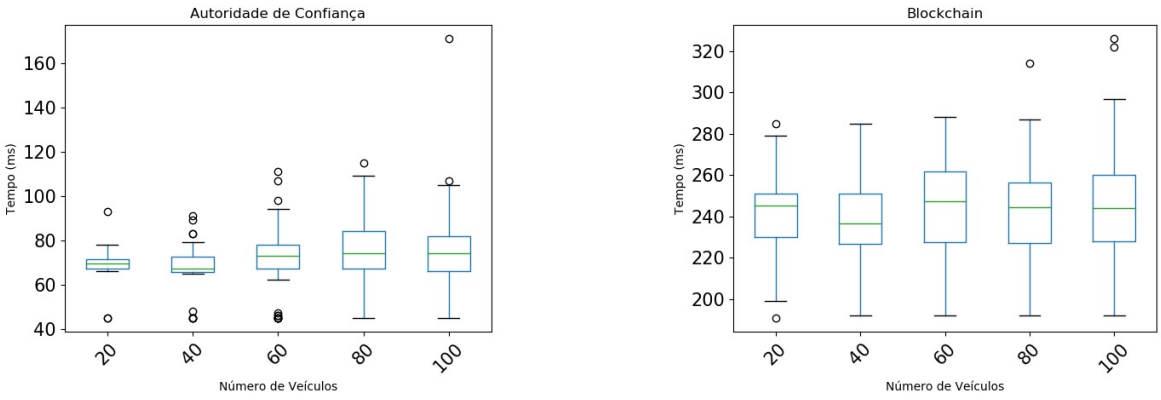
O desempenho da rede em termos de vazão, para a arquitetura centralizada, atingiu taxas superiores de transmissão quando o número de veículos ficou entre 20 e 40. Isso se dá por conta da menor quantidade de requisições para unidade centralizada, facilitando o atendimento e assim aumentando a vazão e diminuindo o atraso. À medida que ampliamos o número de veículos o desempenho tende, gradativamente a se deteriorar, chegando ao patamar de 30% de sua vazão e aumento aproximado de 470% no seu atraso, isto comparado aos valores iniciais. Na arquitetura descentralizada, constatamos maior regularidade no comportamento do desempenho, pode-se observar que não há pico, nem queda da vazão com relação à variação da quantidade de veículos.

As Figuras 33a e 33b representam a média de tempo da atividade de autenticação totalmente concluída nas aplicações, TA e BC. Nota-se uma diferença significativa nestes tempos, com variação aproximada de 150ms, ou 400%, entre os melhores e piores valores obtidos. Para explicar tal discrepância, devemos considerar que a BC necessita submeter as transações de autenticação ao processo de consenso, este procedimento requer maiores recursos computacionais, além de exigir intensa comunicação entre os mineradores impactando no desempenho global da arquitetura.

Podemos ver que a *Blockchain* novamente apresenta regularidade com poucos *outliers*, enquanto a TA tem uma amplitude interquartil muito menor. Porém, mesmo com estes valores atípicos, quando da comparação entre a TA com 100 veículos e a BC com 20 veículos, o tempo de autenticação no cenário com TA é mais eficiente. Isto ocorreu devido à baixa carga de 100

veículos na aplicação centralizada, o que não foi suficiente para sobrecarregar o sistema a ponto de degradar seu desempenho. Além disso, é fundamental considerar o impacto das operações de mineração e consenso realizadas na *Blockchain*, uma vez que esses processos podem introduzir latência e consumir recursos computacionais significativos. Isso pode influenciar diretamente no desempenho do método de autenticação, especialmente em redes onde a alta demanda por validação de transações.

Figura 33 – Tempo de Autenticação



(a) *Trust Authority* (b) *Blockchain*

Fonte: O autor (2022)

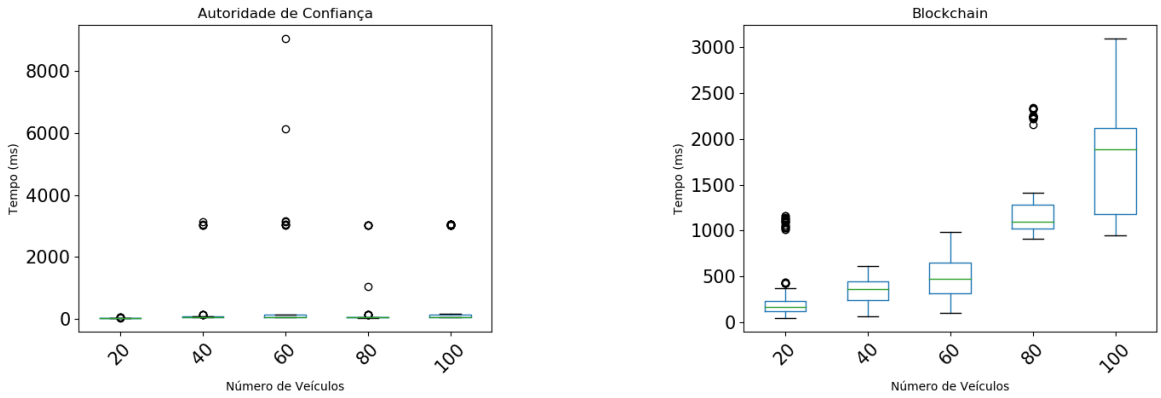
As Figuras 34a e 34b apresentam o envio de mensagens de trânsito. O propósito da implementação da aplicação para trocas de mensagens, foi medir o desempenho das soluções, TA e BC, durante o processo de validação e armazenamento das mensagens. A aplicação da TA, em média, obteve desempenho superior da vazão para a atividade de propagação e validação das mensagens, considerando que a BC precisa tanto validar a mensagem enviada, quanto atingir o consenso para armazenar cada mensagem nos blocos.

6.1.2 OMNeT++ Simulator (TAxBC)

A mudança do simulador ns-3 para o OMNeT++ foi motivada pela necessidade da melhoria em cenários de mobilidade e *handover*. Novas simulações com OMNeT++ foram importantes para criação do novo conjunto de dados, permitindo uma análise aprofundada em relação aos resultados preliminares obtidos com ns-3, subseção 6.1.1. Esta abordagem auxiliou o nosso entendimento dos dados gerados, garantindo a consistência dos resultados.

A tabela 12 apresenta os diversos campos utilizados para avaliar o desempenho dos métodos

Figura 34 – Tempo de Validação de Informações de Trânsito



(a) Troca de Mensagens TA (b) Troca de Transações BC

Fonte: O autor (2022)

de autenticação no contexto da Internet dos Veículos (IoV) em redes 5G. A **Solução**, refere-se ao método de autenticação analisado. A coluna Infra indica o número de antenas disponíveis no cenário. **Carros**, representa a quantidade de veículos presentes no cenário. **Temp-Aut**, (Tempo de Autenticação) é o tempo que cada veículo precisou para se autenticar e entrar na rede. **Transmissão**, refere-se de envio e recebimento dos dados utilizando determinado método de autenticação. **CQI**, (Índice de Qualidade de Canal) é o nível da qualidade do sinal do veículo em relação à antena com a qual está realizando o processo de autenticação. **Atraso**, é o tempo total necessário para que uma determinada requisição seja atendida. Por fim, **Nível de Serviço** é o impacto do número de usuários na qualidade do serviço, mensurado pelo tempo de fila. Este campo mede o *buffer* que armazena as solicitações e o tempo que elas demoram para serem atendidas.

Os dados da Tabela 12 representam resultados médios das simulações de diferentes configurações dos métodos de autenticação com relação à infraestrutura (2, 4 e 6) BS e número de veículos (100, 200 e 300), utilizando a tecnologia de redes 5G.

- Variáveis: Existem três métodos representados nos dados: PoS (*Proof of Stake*), PoW (*Proof of Work*) e a TA (*Trust Authority*). A capacidade das infraestruturas simuladas, com opções de 2, 4 e 6 estações base de transmissão 5G. Quantitativo do número de veículos OBU simulados, variando em 100, 200 e 300 dispositivos.

- Métricas: Tempo médio de autenticação. Vazão média, que indica a quantidade de dados relativos ao serviço de autenticação que podem ser transmitidos em uma unidade de tempo. CQI, índice de aferição da qualidade do sinal no canal. Atraso, tempo total

Tabela 12 – Tabela das Médias dos Dados das Simulações no OMNeT++

Solução	Infra	Carros	Temp-Aut	Transmissão	CQI	Atraso	Nível-Serviço
PoS	2	100	122.118	68.609	12.655	0.004608	1.318601e-08
PoS	2	200	123.067	40.853	12.403	0.004633	2.609199e-08
PoS	2	300	123.664	30.446	12.352	0.004697	3.946867e-08
PoS	4	100	122.448	68.620	13.494	0.004659	1.984018e-08
PoS	4	200	124.025	41.996	13.256	0.004726	3.990698e-08
PoS	4	300	124.573	30.106	12.995	0.004800	5.962043e-08
PoS	6	100	123.045	67.915	13.750	0.004722	2.014463e-08
PoS	6	200	124.317	40.985	13.477	0.004814	4.011311e-08
PoS	6	300	124.408	30.487	13.174	0.004860	6.089781e-08
PoW	2	100	137.549	68.433	12.553	0.004699	1.994759e-08
PoW	2	200	153.288	39.863	12.411	0.004626	2.621466e-08
PoW	2	300	189.647	30.607	12.306	0.004674	3.930956e-08
PoW	4	100	121.910	67.305	13.697	0.004631	1.923848e-08
PoW	4	200	142.177	40.792	13.249	0.004722	3.942700e-08
PoW	4	300	176.901	30.091	12.966	0.004793	5.977882e-08
PoW	6	100	121.330	65.069	13.855	0.004685	1.963928e-08
PoW	6	200	183.284	41.599	13.477	0.004804	4.009362e-08
PoW	6	300	190.892	30.205	13.163	0.004852	6.061353e-08
TA	2	100	71.643	717.561	12.582	0.004852	1.607935e-07
TA	2	200	79.111	490.041	12.540	0.004912	3.388647e-07
TA	2	300	79.509	382.547	12.547	0.005011	5.261948e-07
TA	4	100	64.959	717.175	13.595	0.005025	1.412753e-07
TA	4	200	70.422	490.604	13.603	0.005151	2.919449e-07
TA	4	300	86.412	381.509	13.585	0.005299	4.477317e-07
TA	6	100	70.238	716.740	13.873	0.005073	1.320651e-07
TA	6	200	69.364	490.314	13.871	0.005206	2.729879e-07
TA	6	300	79.360	381.982	13.882	0.005374	4.159080e-07

Fonte: O autor (2024)

médio de transmissão de um pacote entre a origem e o destino. Por fim, o Nível de Serviço, que representa a medida de eficiência e eficácia do método de autenticação para o atendimento aos veículos.

- Tendências: Há uma distinção entre os resultados das três soluções. Por exemplo, em geral, a solução TA apresentou tempos de autenticação mais baixos e vazão mais altos em comparação com as soluções PoS e PoW. Pode-se observar também que, à medida

que o número de infraestruturas aumenta, certas métricas na TA (como o tempo médio de autenticação e o atraso médio) também aumentam o que não ocorre de forma tão linear nem no PoW muito menos no PoS. O crescimento maior no número de dispositivos e infraestrutura impacta bastante no desempenho do método PoW quando comparada ao PoS e TA.

- Implicações: A seleção de uma solução vai depender dos requisitos específicos (por exemplo: velocidade de autenticação, vazão, atraso e qualidade do serviço) desejados, pois diferentes métodos de autenticação centrlizada ou descentralizada podem ser mais apropriados conforme os objetivos de cada operadora ou fornecedora do serviço IoV.

- Escalabilidade: A análise dos dados da Tabela 12 sugere que o desempenho dos métodos de autenticação podem variar dependendo do tamanho da infraestrutura, e número de dispositivos, o que indica possíveis implicações de escalabilidade e otimização dos recursos no ambiente IoV.

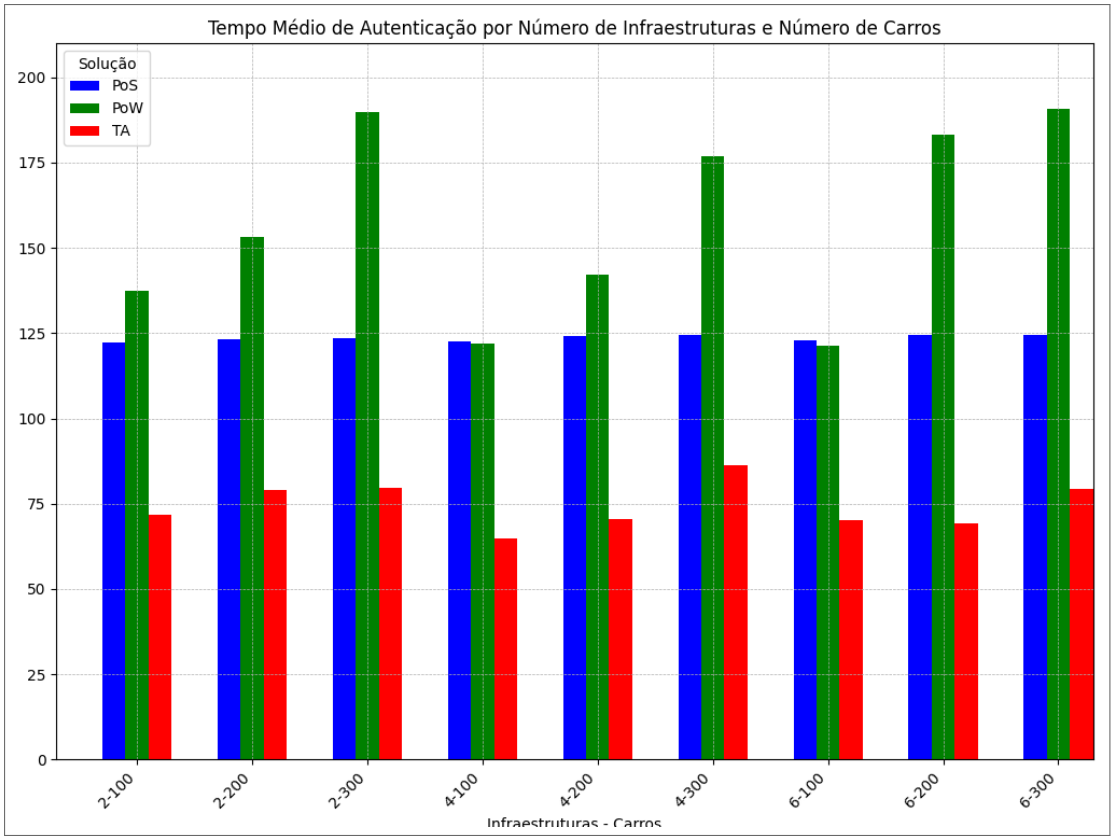
As simulações realizadas com o simulador do OMNeT++ e demais ferramentas e bibliotecas integradas forneceram uma visão inicial dos dados do ambiente IoV utilizando as redes 5G. Com essas simulações, foi possível criar um conjunto de dados com 144.401 registros, levando em consideração todas as simulações combinadas dos métodos, infraestruturas e quantidade de veículos por cenários definidos. Os dados de número de veículos, composição de tráfego e velocidade por faixa foram obtidos a partir de um conjunto de dados fornecido pela empresa de transporte urbano da cidade do Recife, coletados pelo sensor localizado na Av. Agamenon Magalhães, sentido Boa Viagem–Olinda.

6.1.2.1 Avaliação Comparativa

Com base na Figura 35, podemos observar as seguintes variações nos tempos de autenticação para cada solução em relação ao número de carros (100, 200 e 300) e ao número da infraestrutura (2, 4 e 6):

(i) Solução PoS - Tempo de autenticação variou entre 122 e 124 ms. Está solução não apresentou uma variação significativa nos tempos de autenticação em relação ao número de carros ou antenas.

Figura 35 – Tempo de Autenticação



Fonte: O autor (2024)

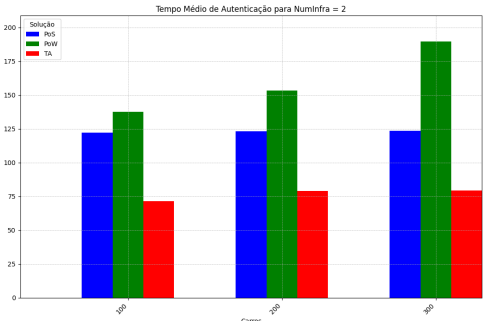
(ii) Solução PoW - Tempo de autenticação variando entre 121 e 197 ms. Foi observada uma variação considerável nos tempos de autenticação, com o tempo máximo sendo quase o dobro do tempo mínimo. Isto se dá pela relação complexa entre o número de carros e a infraestrutura.

(iii) Solução TA - Tempo de autenticação variando entre 64 e 83 ms. Os tempos de autenticação são significativamente menores em comparação com as demais soluções. Os métodos centralizados simplificam o processo, eliminando a necessidade de algoritmos complexos, enquanto sistemas distribuídos exigem ações adicionais para garantir a consistência e disponibilidade entre os nós.

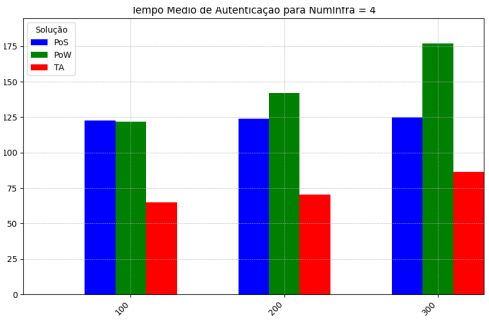
Sistemas distribuídos precisam de mecanismos de coordenação e consenso entre vários nós para garantir a consistência dos dados de autenticação, o que introduz sobrecarga adicional e aumenta o tempo de processamento.

As análises sugerem que o método PoW tem uma variação significativa nos tempos de autenticação, enquanto a TA apresenta os tempos de autenticação mais baixos. Já o método

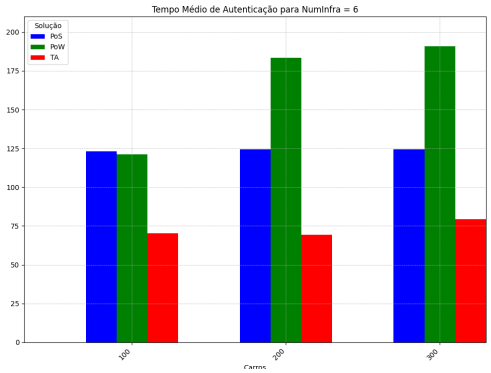
Figura 36 – Tempo de Autenticação das Soluções por Cenários



(a) Infra 02



(b) Infra 04



(c) Infra 06

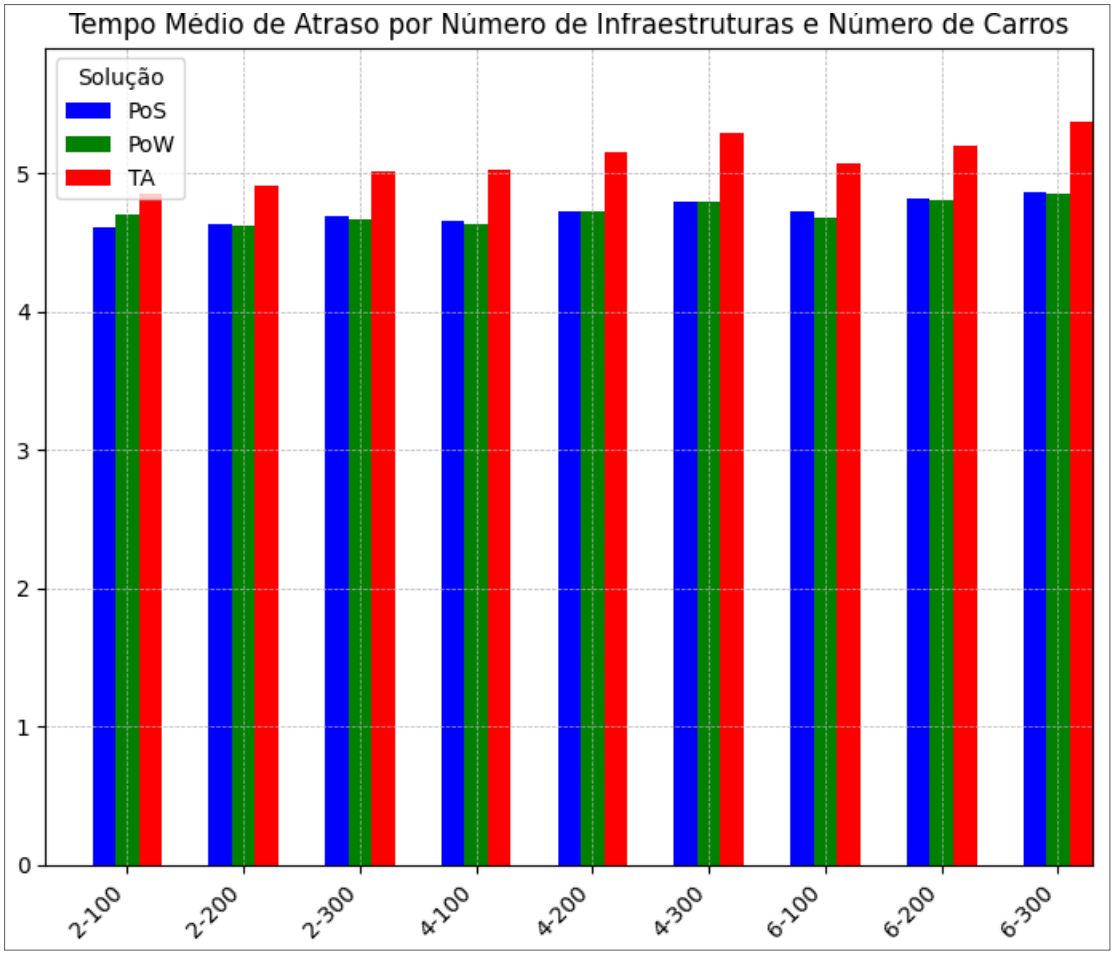
Fonte: O autor (2024)

PoS, obteve os tempos de autenticação mais estáveis, porém mais altos que a TA e mais baixos quando comparados ao PoW.

A Figura 37, exibi o tempo médio de atraso de transmissão, em milissegundos(ms) para os três métodos de autenticação (PoS, PoW e TA), nos diferentes cenários.

(i) Análise dos Métodos: A TA (Trust Authority), apresenta o maior tempo de atraso médio entre as soluções. O tempo de atraso aumenta de forma mais acentuada conforme toda infraestrutura aumenta. Observa-se um crescimento quase linear e constante no tempo de atraso médio conforme a densidade de veículos e o número de antenas 5G aumentam. A BC com o método de consenso PoS (Proof of Stake), tem um desempenho intermediário comparado às outras duas soluções. O tempo de atraso médio é relativamente estável, com algumas flutuações iniciando com menor tempo médio ao longo dos diferentes cenários. Apresenta um baixo aumento no tempo de atraso conforme o número de veículos e infraestrutura cresce. Já a BC com o consenso PoW (Proof of Work), apresenta a menor média de tempo de atraso entre as soluções, devido à natureza mais participativa do seu consenso.

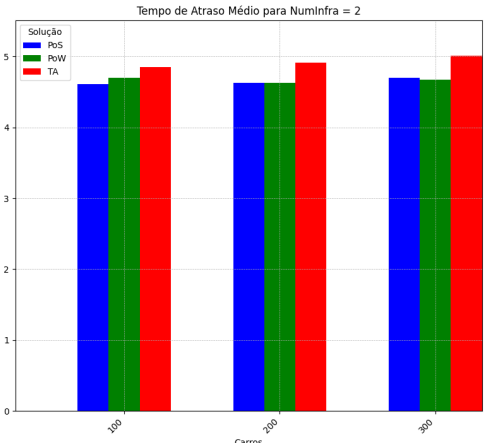
Figura 37 – Tempo Médio de Atraso



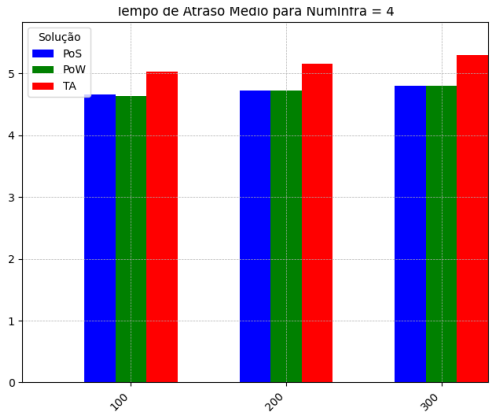
Fonte: O autor (2024)

(ii) Análise por Cenários: Com a infraestrutura 26 ruim, baseada em 2 gNodeBs e 100 dispositivos veículos, o método da BC utilizando o consenso PoS obteve o menor valor de atraso médio de 4.6 ms, já o consenso PoW atingiu 4.7 ms e o método TA ficou em 4.8 ms. No cenário com 200 Carros e a mesma infraestrutura o PoS apresentou um leve crescimento 4.7 ms, o PoW 4.6 ms (menor atraso) e a TA 4.9 ms (maior atraso). O comportamento dos dados permanece quando há um aumento no número de veículos para 300, conforme apresenta os dados, PoS 4.7s, PoW 4.6 ms e a TA com 4.9 ms. Quando consideramos a infraestrutura boa, com 4 gNodeBs, e o número de 100 veículos na BC com o consenso PoS 4.7 ms, no consenso PoW 4.6 ms (menor atraso) e no método da TA centralizada 4.9 ms (maior atraso). Com o acréscimo na quantidade de veículos para 200, os resultados obtidos e apresentados foram PoS 4.7 ms, PoW 4.7 ms (neste cenário os consensos empataram) e TA com 5.1 ms. Ao simularmos novamente o aumento para 300 dispositivos veiculares, os valores foram PoS 4.8 ms, PoW 4.8

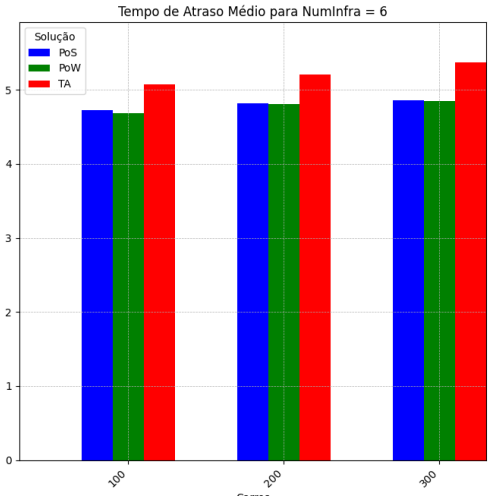
Figura 38 – Tempo Médio de Atraso dos Métodos por Cenários



(a) Infra 02



(b) Infra 04



(c) Infra 06

Fonte: O autor (2024)

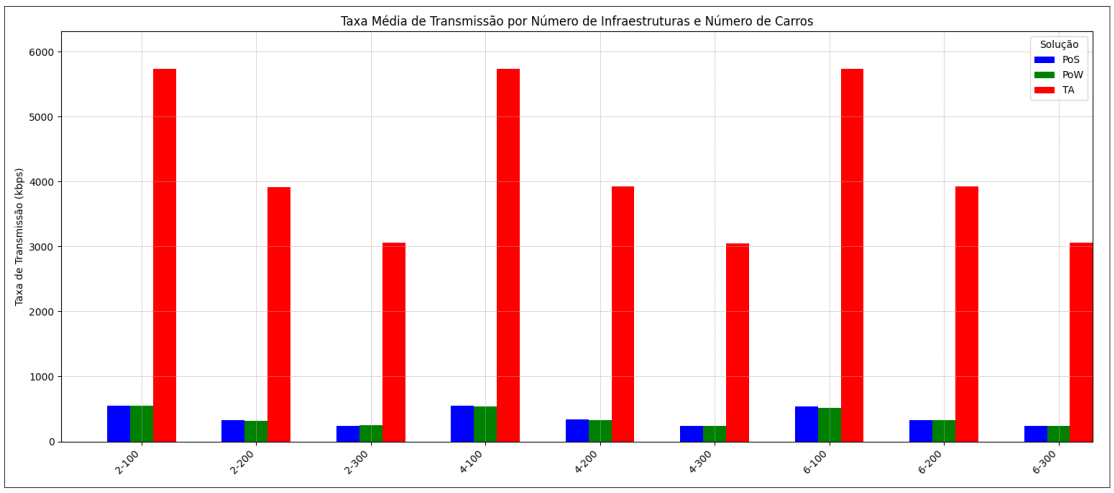
ms (empate novamente) e TA 5.2 ms (maior atraso). Por fim, com a infraestrutura excelente, ou seja, 6 gNodeBs, com 100 veículos alcançamos os seguintes tempos de atraso na BC utilizando o PoS 4.7 ms e o PoW 4.6 ms (menor atraso), com o método da TA 5.0 ms (maior atraso). Com a capacidade da infraestrutura mantida e adicionando o dobro de veículos (200) o PoS 4.8 ms, PoW 4,7 ms e TA 5.2 ms (maior atraso). Por fim, nesta infraestrutura com os 300 veículos na capacidade máxima da simulação tivemos os seguintes resultados de dados de atraso PoS 4.9 ms, PoW 4.8 ms (menor atraso) e TA 5.3 ms (maior atraso).

(iii) Análise do Desempenho: A solução PoW tem o menor tempo médio de atraso, seguida por PoS que se manteve estável e a TA apresentando os piores tempos médios de atraso para qualquer cenário simulado. Com relação à escalabilidade, o aumento do número de

carros e infraestrutura 5G geralmente resulta em um aumento no tempo de atraso médio para todos os métodos, mas esse aumento é mais acentuado no método de autenticação da Autoridade de Confiança(TA). Já ao analisarmos a variabilidade, a solução PoW apresenta uma maior variabilidade nos tempos de atraso, enquanto PoS mostra um padrão mais consistente e previsível que os demais.

A Figura 39 apresenta a taxa média de transmissão (kbps), dos métodos de autenticação TA, PoW e PoS no ambiente da IoV.

Figura 39 – Taxa Média de Transmissão (kbps)



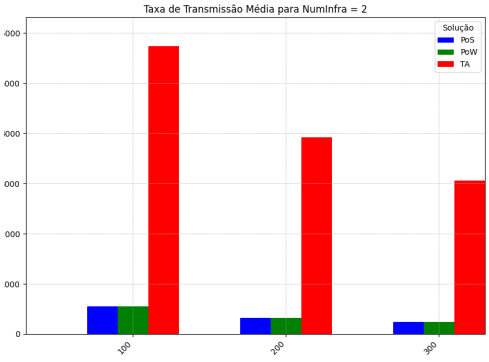
Fonte: O autor (2024)

1. Solução TA: Apresenta taxas de transmissão significativamente maiores em comparação com as outras duas soluções. As taxas de transmissão do método de autenticação TA oscilaram entre aproximadamente 3200 kbps e 5600 kbps. Esta solução demonstra um padrão com tempos de transmissão atingindo picos em intervalos e depois diminuindo, criando um padrão de comportamento. Isso se deve à relação com o número de requisições do serviço centralizado, no qual os dispositivos de borda não contribuem para a melhoria do desempenho.

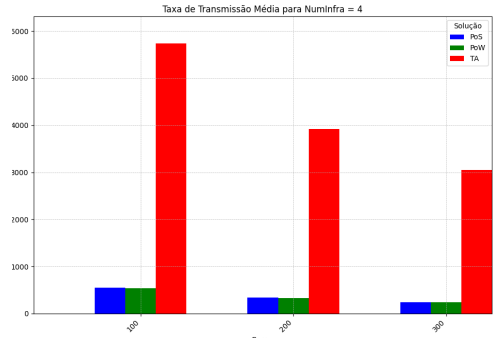
2. Solução PoS e PoW: Ambas as soluções têm tempos de transmissão muito mais baixos e relativamente estáveis em comparação com a solução da TA. Os tempos de transmissão para ambos giram aproximadamente na faixa entre 220 a 550 kbps. Existem pequenas flutuações, mas não há picos ou vales significativos nos consensos utilizando a BC.

3. A partir das análises dos cenários IoV do ambiente de simulação, como, por exemplo, com 2 ou 4 gNodeBs, 100 ou 200 veículos, aplicando o método da TA a solução inicia com

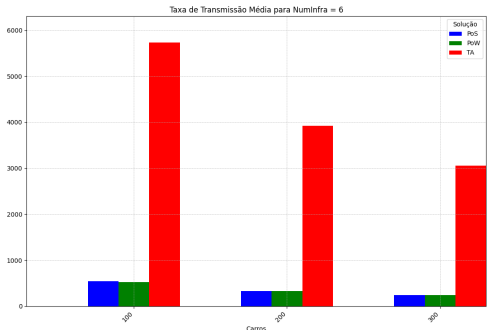
Figura 40 – Taxa Média de Transmissão das Soluções por Cenários



(a) Infra 02



(b) Infra 04



(c) Infra 06

Fonte: O autor (2024)

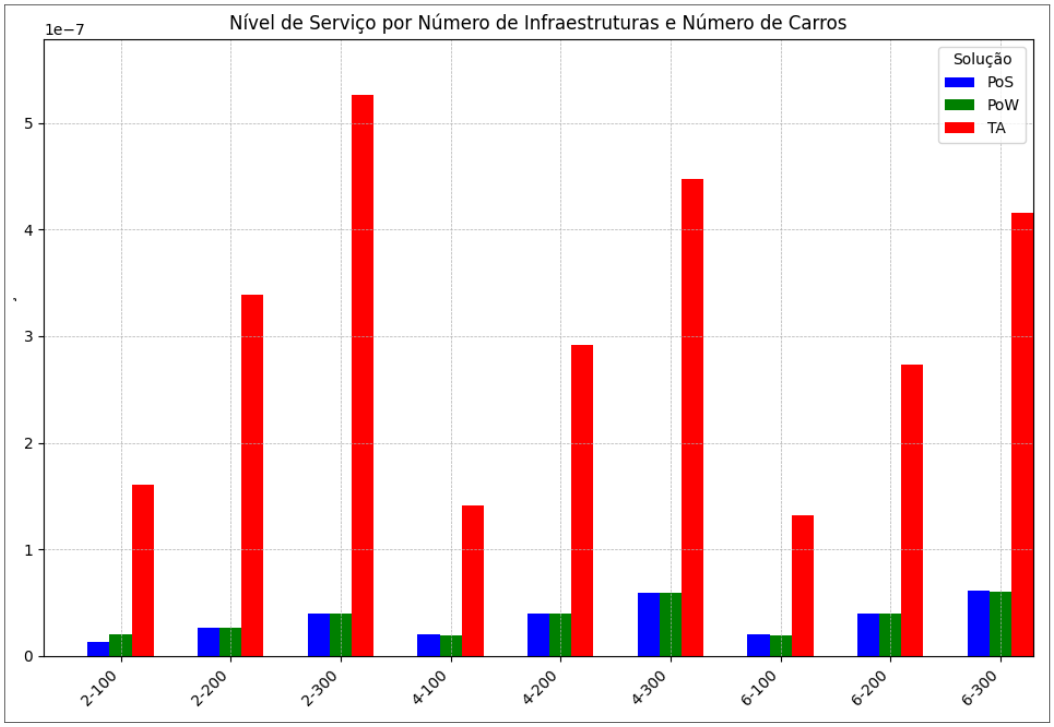
uma taxa de transmissão alta, em torno de 5600 kbps. Os métodos PoS e PoW começam em torno de 400 kbps. Já no cenário com 6 gNodeBs, 300 veículos a TA termina com a taxa aproximada de 3200 kbps. Os métodos PoS e PoW permanecem estáveis com relação as suas capacidades de transmissão independente da elasticidade do ambiente. Os mineradores localizados na borda da rede, mais próximos dos usuários, auxiliam na comunicação e no processo de autenticação, absorvendo o impacto no crescimento do número de usuários.

O método de autenticação da TA apresenta tempos de transmissão altos com flutuações perceptíveis, indicando que pode ser menos eficiente em ambientes de alta concorrência em comparação com PoS e PoW neste contexto. Os métodos PoS e PoW mantêm tempos de transmissão mais estáveis, indicando maior consistência durante a escalabilidade do ambiente IoV simulado. Para aplicações que exigem tempos de transmissão mais estáveis em ambientes densos, o métodos da BC PoS pode ser mais oportuno.

A Figura 41 apresenta o Nível de Serviço em diferentes cenários de número de carros e infraestrutura 5G. As soluções representadas são PoS, PoW e TA. Uma análise dos dados são

apresentados na Figura 41.

Figura 41 – Média do Nível de Serviço dos Métodos



Fonte: O autor (2024)

(i) Análise do Nível de Serviço: O método de autenticação TA, apresenta um comportamento muito variável, com picos significativos nos níveis de serviço comparado às outras soluções. Estes picos ocorreram nos cenários com 2 gNodeBs e 200 veículos e o método TA, 4 gNodeBs 200 veículos e o método TA e 6 gNodeBs 200 veículos e o método TA, indicando uma possível sensibilidade a infraestrutura em relação ao número de carros. A BC utilizando o PoS (*Proof of Stake*), apresentou um nível de serviço mais estável e consistente ao longo dos cenários simulados. No PoS o nível de serviço apresenta pouca variabilidade, mostrando um leve aumento em alguns cenários, indicando menor impacto comparado à solução da TA. Já a BC utilizando o PoW (*Proof of Work*), apresenta uma leve tendência de aumento dos níveis em certos cenários.

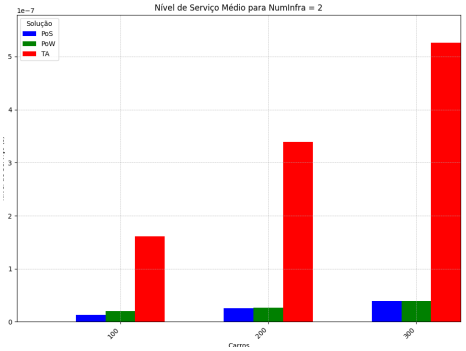
(ii) Análise dos Dados por Cenários:

(A) 2 gNodeBs: 100 Veículos - PoS: 0.4e-7 - PoW: 0.4e-7 - TA: 2e-7 (pico);

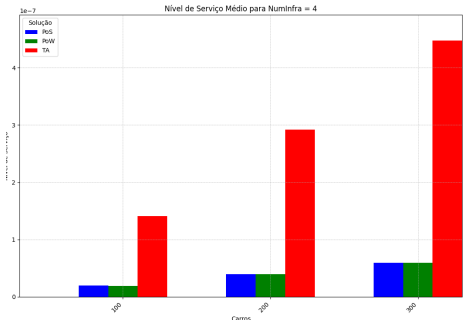
(B) 2 gNodeBs: 200 Veículos - PoS: 0.6e-7 - PoW: 0.5e-7 - TA: 5e-7 (pico);

(C) 2 gNodeBs: 300 Veículos - PoS: 0.7e-7 - PoW: 0.6e-7 - TA: 0.2e-7;

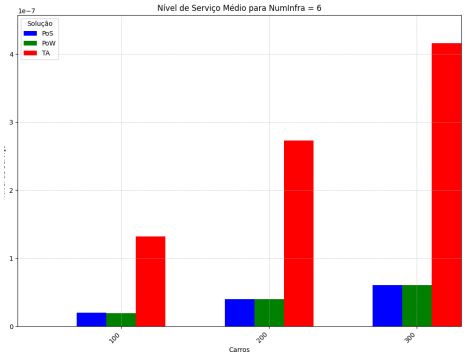
Figura 42 – Tempo Médio do Nível de Serviço por Cenários



(a) Infra 02



(b) Infra 04



(c) Infra 06

Fonte: O autor (2024)

(D) 4 gNodeBs: 100 Veículos - PoS: 0.8e-7 - PoW: 0.7e-7 - TA: 2e-7 (pico);

(E) 4 gNodeBs: 200 Veículos - PoS: 1.0e-7 - PoW: 0.9e-7 - TA: 4e-7 (pico);

(F) 4 gNodeBs: 300 Veículos - PoS: 1.1e-7 (pico); - PoW: 1.0e-7 - TA: 0.2e-7;

(G) 6 gNodeBs: 100 Veículos - PoS: 1.2e-7 - PoW: 1.1e-7 - TA: 2e-7 (pico);

(H) 6 gNodeBs: 200 Veículos - PoS: 1.3e-7 - PoW: 1.2e-7 - TA: 3e-7 (pico);

(I) 6 gNodeBs: 300 Veículos - PoS: 1.4e-7 (pico) - PoW: 1.3e-7 - TA: 0.2e-7.

Com relação à estabilidade e consistência, os métodos de consenso da BC PoS e PoW apresentam níveis de serviço mais estáveis, com pouca variação ao longo dos diferentes cenários. Já questões relacionadas a variabilidade, a TA mostra uma grande instabilidade no nível de serviço, com picos altos em determinados cenários.

Esta análise sugere que, enquanto a solução TA pode alcançar níveis de serviço muito altos em certas condições, PoS e PoW da BC oferecem um desempenho mais estável e confiável. A escolha entre estes métodos deve considerar as necessidades das variáveis do ambiente

IoV observando todos os parâmetros em condições específicas de configurações da rede de comunicação e do ecossistema de cada operadora de telecomunicação.

6.2 AVALIAÇÃO DO SIMA-IOV NO AMBIENTE 5G

Esta seção demonstra os resultados das análises do Mecanismo Inteligente de Seleção do Método de Autenticação Baseado no Contexto IoV em Redes 5G (SIMA-IoV), focando em quatro parâmetros essenciais: número de veículos, tempo de autenticação, vazão e atraso da rede.

Os resultados obtidos apresentam que o método de autenticação baseado na TA apresentou melhor desempenho nos requisitos do tempo de autenticação e taxa de vazão no ambiente IoV, especialmente quando se trata de quantidades menores de veículos, na faixa de 100 a 200 dispositivos. Esta eficiência pode ser atribuída à capacidade da TA de realizar a autenticação de forma mais direta e objetiva sem necessidade de processamentos e processos adicionais, melhorando assim a comunicação entre os veículos e a infraestrutura da rede. A rapidez no processamento centralizado da TA torna-o especialmente vantajoso em cenários onde a quantidade de veículos é limitada, sem gargalos, permitindo um desempenho otimizado e respostas mais imediatas.

Por outro lado, o método PoW demonstrou ser o menos eficiente, apresentando os tempos de autenticação mais elevados, mesmo na menor faixa de veículos (100-200). A ineficiência do PoW pode ser explicada pelo alto custo computacional inerente ao seu mecanismo de consenso, que requer a resolução de complexos problemas matemáticos. Este processo não só consome mais tempo, mas também demanda maior poder computacional e energético, tornando-o menos capaz para as situações onde a velocidade de autenticação é crítica.

O PoS, embora apresente tempos de autenticação maiores que o TA, ainda é mais eficiente que o PoW. Este método mostrou-se particularmente eficaz em faixas intermediárias de veículos (350-700), onde o TA não obteve resultados satisfatórios no requisito do atraso segundo os dados do SIMA-IoV. A consistência nos tempos de autenticação do PoS em diferentes faixas pode ser atribuída ao seu mecanismo de consenso baseado no perfil dos validadores, o que reduz significativamente a necessidade de processamento intensivo e, portanto, o tempo de autenticação. A estabilidade do PoS nestas condições indica que ele pode ser uma solução viável para redes de tamanho médio, onde o equilíbrio entre eficiência e escalabilidade é crucial.

Em suma, a análise dos métodos de autenticação TA, PoW e PoS revela que cada um

Tabela 13 – Tabela Análises do SIMA no Ambiente IoV

Solução	Carros	Infra	Temp Autenticação	Vazão	Atraso
TA	102	6	68.6633	697.3163	0.005549
TA	182	6	69.8304	511.7096	0.005441
TA	247	6	68.3383	426.8610	0.005593
TA	255	6	68.7341	418.5852	0.005624
TA	199	4	65.4722	491.4218	0.005336
TA	211	4	67.3895	474.5903	0.005424
TA	212	4	68.0379	473.2472	0.005434
TA	226	4	62.7316	455.5009	0.005199
TA	852	2	78.0469	390.6415	0.005653
TA	890	2	82.7311	382.0520	0.005223
TA	983	2	82.9762	380.1094	0.005325
TA	997	2	84.6869	377.3849	0.005968
PoW	116	2	161.604	53.4750	0.004581
PoW	128	2	157.919	57.0756	0.004602
PoW	232	2	159.464	33.1546	0.005027
PoW	297	2	161.901	32.6936	0.004828
PoW	221	4	226.872	38.7999	0.004847
PoW	555	4	219.613	31.3716	0.004985
PoW	590	4	229.822	31.7625	0.004959
PoW	693	4	239.278	29.9296	0.004773
PoW	141	6	207.748	52.4989	0.004613
PoW	209	6	214.590	38.3752	0.004759
PoW	238	6	215.363	36.1043	0.004967
PoW	118	6	213.184	68.1254	0.004669
PoS	474	2	132.457	29.6101	0.004669
PoS	468	2	132.088	33.7797	0.004716
PoS	665	2	129.486	27.9824	0.004773
PoS	833	2	127.918	27.3883	0.004815
PoS	219	4	156.734	36.8327	0.004730
PoS	662	4	155.405	29.7545	0.004771
PoS	871	4	158.160	28.6874	0.004861
PoS	473	4	158.013	32.2994	0.004928
PoS	300	6	189.016	33.9429	0.005029
PoS	928	6	172.051	33.6369	0.005042
PoS	147	6	175.188	50.2619	0.004944

Fonte: O autor (2024).

possui características específicas a depender da quantidade de veículos e *gNodeBs* na rede. O TA destaca-se em cenários com menor quantidade de veículos, o PoS em faixas intermediárias, e o PoW, apesar de sua ineficiência temporal, pode ainda ser considerado em contextos onde a segurança baseada em alto poder computacional é prioritária. Os resultados demonstram a importância de selecionar o método de autenticação conforme o contexto e os requisitos específicos de cada rede veicular, garantindo assim um desempenho otimizado e eficiente com base nas características de cada ambiente.

Os dados da Tabela 13 representam resultados das análises do SIMA-IoV no ambiente simulado da IoV/5G, considerando nesta análise um intervalo maior de veículos de 100 a 1000, ampliando assim as possibilidades, escalabilidade e a capacidade para obter e refinar os resultados buscando desta forma maior assertividade em cada método de autenticação com relação à infraestrutura, as redes de comunicação e a densidade de dispositivos móveis.

Com relação ao tempo de autenticação é essencial garantir que os veículos possam se conectar e trocar informações de maneira segura e eficiente, mantendo os requisitos de desempenho para cada cenário do ambiente IoV. O SIMA-IoV demonstrou a capacidade de buscar e selecionar o melhor método de autenticação com base nas informações obtidas no ambiente com uma redução significativa no tempo de análise para cada método, um fator crucial para o ambiente dinâmico da IoV e das redes de quinta geração. Com tempos de autenticação reduzidos em determinados cenários, os veículos podem iniciar suas comunicações utilizando os serviços e aplicações mais rapidamente, resultando em uma experiência de uso e operação mais fluída no Ambiente.

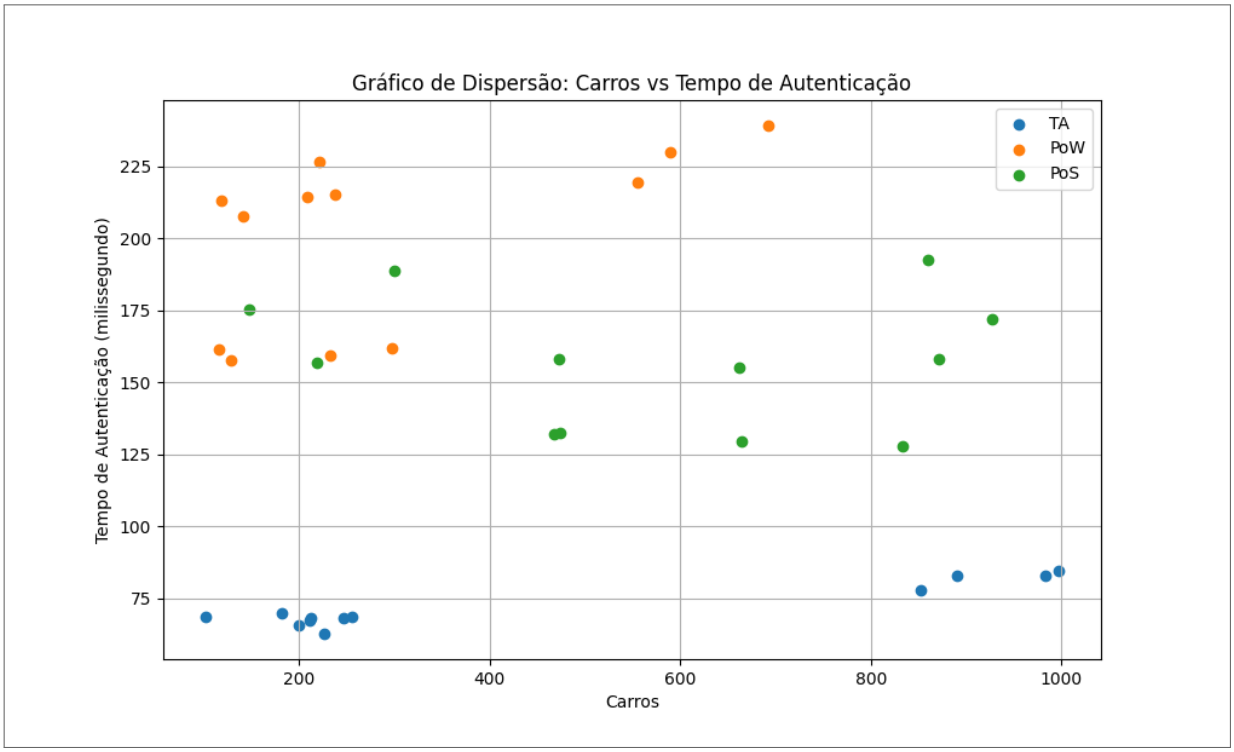
A vazão da rede é um indicador de quanta informação pode ser transmitida através da rede em um determinado tempo. No contexto da IoV, uma maior vazão permite que mais dados sejam trocados entre os veículos e a infraestrutura, possibilitando uma comunicação mais eficiente. As análises do SIMA-IoV apresentam quais os métodos de autenticação utilizando a rede 5G, conseguem melhores vazões de dados para os cenários analisados. Esta seleção é de suma importância para aplicações em tempo real, visão computacional, mapas, alerta de trânsito e compartilhamento de arquivos de áudio e vídeo na rede.

O atraso da rede, ou latência, refere-se ao tempo que leva para um dado percorrer entre os veículos ou entre o veículo e a infraestrutura. Para um ambiente IoV, onde as decisões precisam ser determinadas em frações de segundos, a baixa latência deve ser um objetivo primordial. Os resultados das análises indicam que o SIMA-IoV conseguiu selecionar os menores atrasos dos métodos de autenticação na rede de forma eficaz. Com menor latência, os veículos podem

reagir mais rapidamente às mudanças no ambiente, melhorando a segurança e a eficiência do ecossistema de tráfego nas vias.

O mecanismo de otimização SIMA-IoV pode ser empregado para análise de ambientes, aprimorando as tomadas de decisão sobre qual método de autenticação deve ser aplicado a determinado cenário local. Com a seleção do método correto, teremos também a redução do tempo de autenticação, aumento da vazão da rede, diminuição do atraso e a melhora na qualidade de serviço, potencializando os requisitos dos serviços e da comunicação, promovendo um ambiente mais eficiente. Esses avanços não apenas melhoram a experiência do usuário, mas também contribuem para os serviços e aplicações que fazem parte do ecossistema ITS. Os resultados das análises do SIMA-IoV realizadas e apresentadas, destacando o potencial de cada método de autenticação conforme os parâmetros de cada cenário da rede IoV/5G.

Figura 43 – Análise do SIMA-IoV - Dispersão dos Tempos de Autenticação



Fonte: O autor (2024)

A Figura 43 apresenta uma dispersão dos melhores tempos de autenticação obtidos pelo SIMA-IoV para os diferentes números de veículos e infraestrutura entre os três métodos disponíveis: TA, BC(PoW e PoS). No eixo X estão o número de veículos, no eixo Y o tempo de autenticação. Os pontos representam os tempos de autenticação para cada solução: TA (azuis), PoW (laranja) e o PoS (verdes).

A TA é método mais eficiente em termos de tempo de autenticação, especialmente para

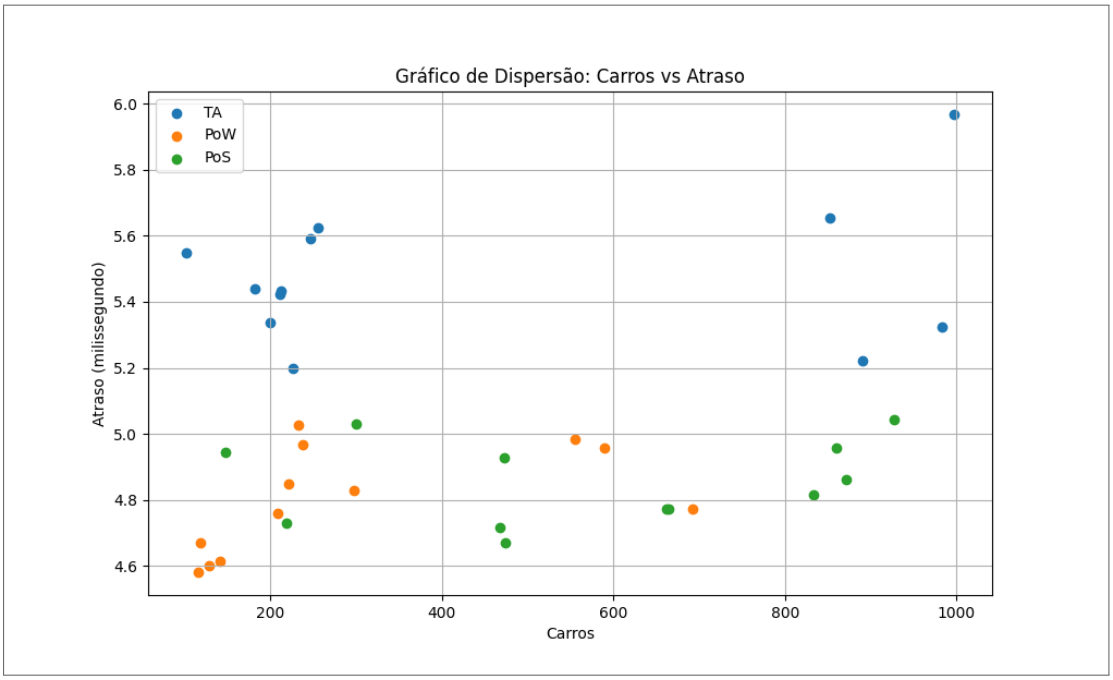
Tabela 14 – Autenticação (TA, PoW, PoS) em função do número de veículos e tempos de autenticação.

Solução	Número de Veículos	Tempo de Autenticação (ms)
TA	50 – 250, 850 – 1000	62 – 85
PoW	116 – 693	157 – 239
PoS	100 – 1000	127 – 193

quantidades menores de veículos (100-200), apresentando neste espaço o melhor desempenho. O PoW é o método que apresentou menos eficiência, com os tempos de autenticação mais elevados mesmo na faixa menor de veículos (100-200). Já o PoS revelou tempos de autenticação maiores do que a TA, só que menores do que o PoW. Isto revela uma boa eficiência para faixas intermediárias de veículos (350-700), onde a TA não conseguiu obter resultados satisfatórios segundo o SIMA-IoV, o PoS também apresentou consistência nos tempos para as demais faixas demonstrada na Figura 43.

A TA se destacou como a mais eficiente, enquanto PoW apresenta os maiores desafios com tempos de autenticação elevados e maior variabilidade devido a seu alto custo energético e computacional. O PoS permaneceu como meio-termo para os tempos de autenticação, com razoável condição de variabilidade do número de veículos (100-1000) e da infraestrutura (2-6) no ambiente no IoV/5G.

Figura 44 – Análise da SIMA-IoV - Dispersão do Tempo de Atraso na Comunicação



Fonte: O autor (2024)

Para realizar uma análise detalhada dos dados apresentados na Figura 44, considerando o

método da TA*(Trust Authority)*, pontos azuis, estão distribuídos principalmente na faixa de atraso entre 0.0052ms a 0.0057ms, para o número de veículos, na faixa de (200 a 300). O atraso da TA é mais alto quando comparado aos valores dos métodos PoW, pontos laranja, e PoS, pontos verde, durante toda a simulação nos cenários de análise do mecanismo. O método da Autoridade de Confiança (TA) pode enfrentar dificuldades de escalabilidade em ambientes com um grande número de veículos. À medida que a quantidade de veículos cresce, a sobrecarga associada ao processo de autenticação também tende a crescer proporcionalmente a depender dos recursos, resultando em atrasos significativos para este método. Portanto, é crucial considerar alternativas ou aprimoramentos no método TA para manter a eficiência e a qualidade do serviço em cenários de alta densidade veicular.

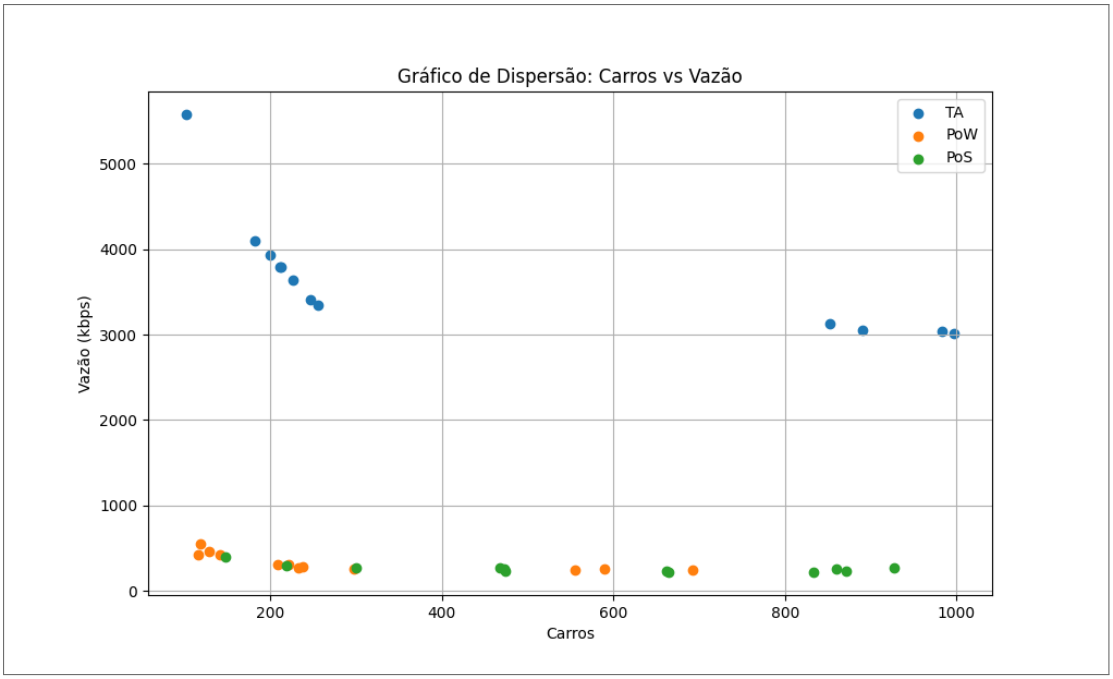
Os métodos da BC, PoW e PoS, pontos laranja e verde, estão principalmente na faixa de atraso entre 0.0045ms e 0.0051ms. Estes métodos apresentam um atraso menor em comparação com a TA. No PoW há uma concentração de pontos em torno de 100 a 300 carros com atraso próximo a 0.0045ms, indicando que esta solução pode obter melhores resultados com um número menor de veículos em termos de atraso. O método do PoS os pontos estão distribuídos na faixa de atraso entre 0.0047ms e 0.0051ms. O atraso no PoS é semelhante ao da solução PoW, ambos mostrando menor atraso quando comparados à TA que variou entre 0051ms e 0059ms. Considerando que o PoW e PoS mostram um bom desempenho em termos de atrasos com um aumento gradual em função do número de carros, mantendo-se abaixo da faixa de 0.0051ms.

Ao compararmos os métodos da BC PoW e PoS distrbuídos, ambos apresentam um menor atraso sugerindo que são mais eficientes em termos de tempo de resposta na IoV em relação à TA que apresentou um maior atraso talvez por sua característica arquitetural centralizada, especialmente à medida que o número de carros aumenta, o que pode indicar uma menor escalabilidade ou eficiência no tempo de resposta em cenários de alta demanda.

Os métodos PoW e PoS são superiores à solução TA em termos de atraso na rede 5G no ambiente IoV, há uma reduzida diferença encontrada pelo SIMA-IoV, mas que permite definir que o PoW obtém resultados mais relevantes na faixa dos 100 aos 300 veículos e o PoS é melhor quando atua na faixa com maior densidade de dispositivos dos 800 a 1000. A solução TA, exibiu limitações em termos de atraso à medida que a carga na rede aumenta, o que pode estar relacionado com sua natureza centralizada, onde podem ocorrer enfileiramentos para atendimento de demandas.

A Figura 45 apresenta a dispersão das capacidades de vazão para diferentes números de

Figura 45 – Análise da SIMA-IoV - Dispersão Vazão dos Métodos de Autenticação



Fonte: O autor (2024)

carros e infraestruturas, categorizadas por três diferentes métodos de autenticação. No eixo X, os veículos e no eixo Y estão os valores da Vazão. Os pontos azuis representam a TA, o PoW são os pontos laranja e o PoS são pontos verdes.

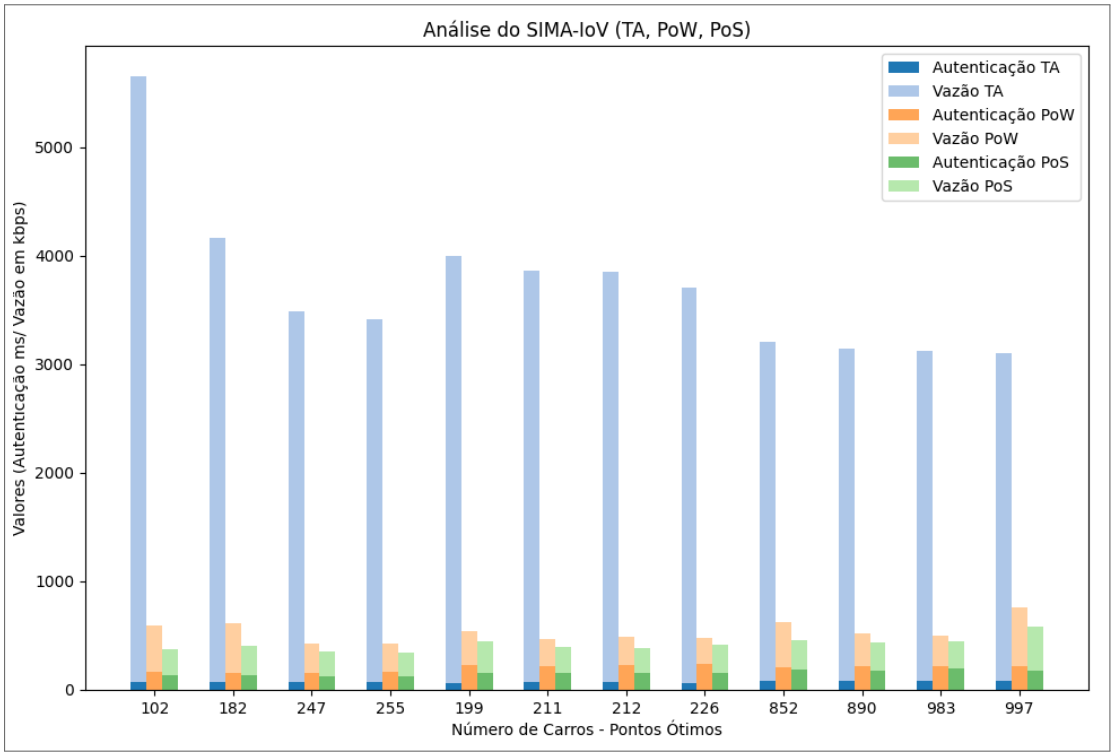
O método da TA, obtém o melhor desempenho em duas faixas principais de carros: de 100 a 200 e de 800 a 1000 com leve queda quando comparado a primeira faixa. Na faixa de 100 a 200 veículos, a vazão varia entre aproximadamente 3.280 a 5.600 Kbps. Para a faixa de 800 a 1000 veículos, a vazão é aproximadamente 3.040 Kbps. A TA apresentou uma vazão significativamente maior em comparação com PoW e PoS. Ao analisarmos, o método do PoW predominou na faixa de carros iniciais entre 100 e 200, e a sua vazão variou em aproximadamente 520 kbps. Desta forma o método PoW apresentou uma eficiência de vazão muito baixa.

Já a BC com o consenso PoS o mecanismo SIMA-IoV localizou alguns pontos de vazão durante toda a faixa de veículos da simulação para qualquer cenário. Sua vazão variou em aproximadamente 408 Kbps. Logo, o PoS tem sua vazão similar à de PoW, mas com uma distribuição mais dispersa. Isso indica um comportamento linear da vazão neste método. A TA é o método mais eficiente, com uma vazão muito superior em comparação com PoW e PoS. O PoW é menos eficiente que a TA e um pouco mais eficiente que o PoS. Isso indica possíveis gargalos no modo de atuação do método PoS na rede IoV simulada. Já o PoS apresenta uma

eficiência de vazão um pouco abaixo que PoW e significativamente menor que TA.

A Figura que apresenta a dispersão 45 destaca a superioridade do método da TA em termos de vazão, enquanto PoW e PoS apresentam vazões significativamente menores e mais variáveis. Isso sugere que, para aplicações que exigem alta eficiência de vazão, TA é o melhor método, enquanto PoW e PoS tiveram maior distribuição na variabilidade dos cenários, necessitam de otimizações substanciais para aprimorarem suas capacidades de vazão na rede conforme selecionadas pelo SIMA-IoV, demonstrando assim toda carga que os algoritmos de consenso e os processos de controle de integridade resultam para o requisito da vazão.

Figura 46 – Análise da SIMA-IoV - Tempo de Autenticação / Vazão



Fonte: O autor (2024)

A Figura 46 apresenta os dados referente ao tempo de autenticação e à vazão dos três métodos (TA, PoW e PoS) durante o processo de simulação e análise em tempo real do SIMA-IoV. É importante observar as diferenças de desempenho entre eles em termos de tempo de autenticação (ms) e vazão (Kbps) para cada cenário observado. A TA (Azul Escuro), conseguiu os menores tempos em todos os pontos ótimos do limite mínimo e máximo de veículos, indicando que TA é mais eficiente em termos de tempo de autenticação. O método da BC utilizando o PoW (Laranja) mostrou tempos maiores do que os obtidos pela TA e a BC utilizando o PoS. Isto era esperado, uma vez que o consenso PoW é computacionalmente mais intenso devido ao processo de mineração, o que leva a maiores tempos de autenticação.

A BC com PoS (Verde), apresenta tempos de autenticação que são intermediários entre TA e PoW. Ele demonstra um desempenho de autenticação razoável, isto tanto nos cenários de simulação (100, 200 e 300) quanto de análise do SIMA-IoV (100-1000) veículos.

Ao observarmos novamente na Figura 46 a TA (Azul Claro), apresenta a maior vazão entre os três métodos, sendo capaz de processar mais dados por unidade de tempo. A sua vazão é mais estável e alta, o que é benéfico para redes que necessitam de alta capacidade de transferência de dados. O PoW (Laranja Claro) é significativamente menor que a TA. Isto reflete na maior carga computacional e tempo necessário para resolver os problemas desta forma de consenso "prova de trabalho", o que limita a quantidade de dados que podem ser processados rapidamente. Por fim, a Figura 46 apresenta as análises da vazão do método da BC com o consenso PoS (Verde Claro), indicando ser uma boa solução intermediária, equilibrando entre tempo de autenticação e capacidade de transferência de dados. O PoS apresenta uma boa vazão, superior a PoW, mas inferior a TA.

Com base nas análises do SIMA-IoV, a TA, em ambos os parâmetros de tempo de autenticação e vazão examinados, foi mais eficiente. No entanto, a *Blockchain* com o PoS oferece um equilíbrio entre os dois métodos extremos Ta e PoW. Apresenta tempos de autenticação mais baixos que o PoW e uma vazão com comportamento linear independente do cenário analisado, embora ainda inferior à TA. Além disso, a BC com PoS obteve atrasos aceitáveis e com boa escalabilidade para todos os cenários, o que a torna mais efetivo para aplicações que se desejam se beneficiar da imutabilidade, privacidade, e maior disponibilidade características dos serviços distribuídos. O PoW, por outro lado, tem o pior desempenho em termos de tempo de autenticação, refletindo sua natureza de trabalho intensiva e computacionalmente custosa. Em resumo, enquanto a TA se destaca em eficiência para autenticação e vazão, a BC com PoS oferece uma alternativa viável, combinando tempos de autenticação aceitáveis com os benefícios adicionais de segurança e privacidade proporcionados pela imutabilidade da *Blockchain*.

6.3 ANÁLISES ESTATÍSTICAS DOS DADOS DO SIMA-IOV

Os resultados de uma análise de variância (ANOVA) realizada para avaliar as diferenças nos valores médios de três grupos diferentes: TA, PoW e PoS. A análise foi realizada com um nível de confiança de 95% e inclui testes de comparações múltiplas de Tukey para identificar quais grupos diferem significativamente entre si.

Análise de Variância (ANOVA) é uma técnica estatística utilizada para comparar as médias de três ou mais grupos independentes. Ela avalia se há diferenças estatisticamente significativas entre as médias desses grupos, analisando a variabilidade dos dados e dividindo a variação total em componentes atribuíveis às diferentes fontes de variação. Em essência, a ANOVA permite determinar se as diferenças observadas entre os grupos são devido a efeitos reais ou se podem ser atribuídas ao acaso (MONTGOMERY, 2017).

Optar pela ANOVA ao invés de outras técnicas, como o teste t ou MANOVA, pode ser justificado por algumas razões:

- Teste t: O teste t é efetivo para comparar as médias de apenas dois grupos. Quando há três ou mais grupos, a aplicação de múltiplos testes t aumenta a probabilidade de cometer erros do Tipo I (falsos positivos). A ANOVA, por outro lado, permite a comparação de múltiplos grupos simultaneamente, mantendo o controle sobre a taxa de erro global.
- MANOVA (Multivariate Analysis of Variance): Enquanto a MANOVA é utilizada para testar diferenças em múltiplas variáveis dependentes simultaneamente, a ANOVA é focada em uma única variável dependente. Se o objetivo do estudo é comparar as médias de uma única variável entre vários grupos, a ANOVA é mais apropriada e menos complexa do que a MANOVA. Além disso, a ANOVA é mais fácil de interpretar quando se está lidando com uma única variável de interesse.

A escolha da Análise de Variância (ANOVA) é apropriada quando se pretende comparar as médias de uma única variável dependente entre três ou mais grupos. Esse método evita as limitações associadas à realização de múltiplos testes t e a complexidade adicional envolvida na Análise de Variância Multivariada (MANOVA).

6.3.1 Análise para Infraestrutura com 2 Antenas

Tabela 15 – Resultados da ANOVA

F Variação	GL	Soma Quadrados	Média Quadrática	Valor F	Pr(>F)
Grupos	2	12435	6217	1143	1.48×10^{-11} ***
Resíduos	9	49	5		

Fonte: O autor (2024).

Os resultados da ANOVA, Tabela 15 indicam que há uma diferença significativa entre os grupos (Valor-p $=$ 1.48e-11), sugerindo que pelo menos um dos grupos difere significativa-

mente dos outros em termos de valores médios.

Tabela 16 – Comparações múltiplas das médias de Tukey com nível de confiança de 95%

Comparação	Diferença de Média	Limite Inferior	Limite Superior	Valor-p
PoW - PoS	29.73475	25.13107	34.33843	1×10^{-7}
TA - PoS	-48.37697	-52.98066	-43.77329	0
TA - PoW	-78.11172	-82.71541	-73.50804	0

Fonte: O autor (2024).

A tabela 16, apresenta as comparações múltiplas de Tukey entre os grupos, com um nível de confiança de 95%. Os resultados das comparações de Tukey mostram que todos os grupos diferem significativamente entre si (Valor-p < 0.001 para todas as comparações).

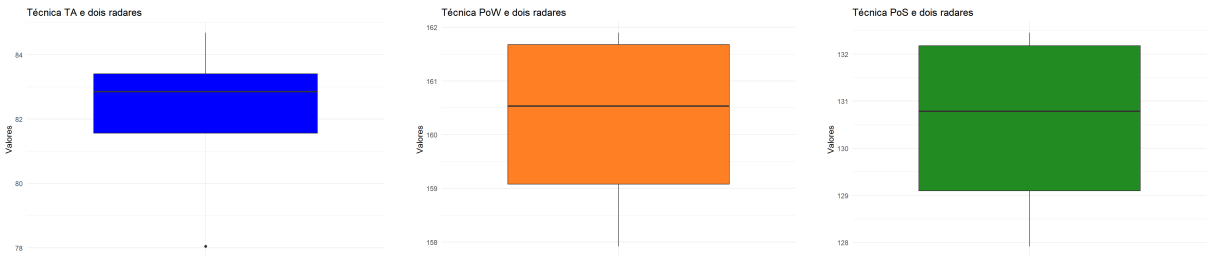
Tabela 17 – Médias dos grupos. O grupo TA tem a menor média.

Grupo	Índice	Média
TA	1	82.11028
PoW	2	160.222
PoS	3	130.4872

Fonte: O autor (2024).

Os resultados da Tabela 17, indicam que o grupo TA tem a menor média (82.11028), seguido pelo grupo PoS (130.4872) e o grupo PoW (160.222). As diferenças entre as médias dos grupos são todas estatisticamente significativas.

Figura 47 – Análise de Variância para Infraestrutura com 2 Antenas



(a) TA - Infra 02 (b) PoW - Infra 02 (c) PoS - Infra 02

Fonte: O autor (2024).

A análise estatística de variância, exibidas nas Figuras 47a, 47b, 47c, revelaram as diferenças significativas nos valores médios entre os grupos TA, PoW e PoS. As comparações múltiplas

de Tukey confirmaram que todos os grupos diferem significativamente entre si. O grupo TA apresentou a menor média, enquanto o grupo PoW apresentou a maior média.

6.3.2 Análise para Infraestrutura com 4 Antenas

Tabela 18 – Resultados da ANOVA

F Variação	GL	Soma Quadrados	Média Quadrática	Valor F	Pr(>F)
Grupos	2	53380	26690	1087	1.85×10^{-11} ***
Resíduos	9	221	25		

Fonte: O autor (2024).

Os resultados da Tabela 18 ANOVA, indicam que há uma diferença significativa entre os grupos (Valor-p = 1.85e-11), sugerindo que pelo menos um dos grupos difere significativamente dos outros em termos de valores médios.

Tabela 19 – Comparações múltiplas de médias de Tukey com nível de confiança de 95%

Comparação	Diferença de Média	Limite Inferior	Limite Superior	Valor-p
PoW - PoS	71.81825	62.03766	81.59884	0
TA - PoS	-91.17020	-100.95079	-81.38961	0
TA - PoW	-162.98845	-172.76904	-153.20786	0

Fonte: O autor (2024).

A Tabela 19, apresenta as comparações múltiplas de Tukey entre os grupos, com um nível de confiança de 95%.

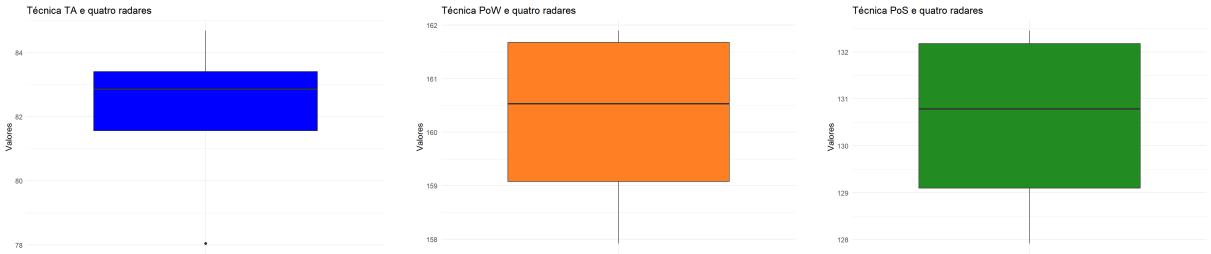
Tabela 20 – Médias dos grupos. O grupo TA tem a menor média.

Grupo	Índice	Média
TA	1	65.9078
PoW	2	228.8963
PoS	3	157.078

Fonte: O autor (2024).

Os resultados da Tabela 20, indicam que o grupo TA tem a menor média (65.9078), seguido pelo grupo PoS (157.078) e o grupo PoW (228.8963). As diferenças entre as médias dos grupos são todas estatisticamente significativas.

Figura 48 – Análise de Variância para Infraestrutura com 4 Antenas



(a) TA - Infra 04 (b) PoW - Infra 04 (c) PoS - Infra 04

Fonte: O autor (2024).

A análise de variância das Figuras, 48a, 48b, 48c, revelaram as diferenças significativas nos valores médios entre os grupos TA, PoW, PoS e um grupo não especificado. As comparações múltiplas de Tukey confirmaram que todos os grupos diferem significativamente entre si. O grupo TA apresentou a menor média, enquanto o grupo PoW apresentou a maior média.

6.3.3 Análise para Infraestrutura com 6 Antenas

Tabela 21 – Resultados da ANOVA

F Variação	GL	Soma Quadrados	Média Quadrática	Valor F	Pr(>F)
Grupos	2	45941	22970	605.6	2.54×10^{-10} ***
Resíduos	9	341	38		

Fonte: O autor (2024).

Os resultados da Tabela 21 ANOVA, indicam que há uma diferença significativa entre os grupos (Valor-p = 2.54e-10), sugerindo que pelo menos um dos grupos difere significativamente dos outros em termos de valores médios.

Tabela 22 – Comparações múltiplas de médias de Tukey com nível de confiança de 95%

Comparação	Diferença de Média	Limite Inferior	Limite Superior	Valor-p
PoW - PoS	30.53275	18.37372	42.69178	0.0001651
TA - PoS	-113.29697	-125.45601	-101.13794	0.0000000
TA - PoW	-143.82972	-155.98876	-131.67069	0.0000000

Fonte: O autor (2024).

A Tabela 22, apresenta as comparações múltiplas de Tukey entre os grupos, com um nível de confiança de 95%.

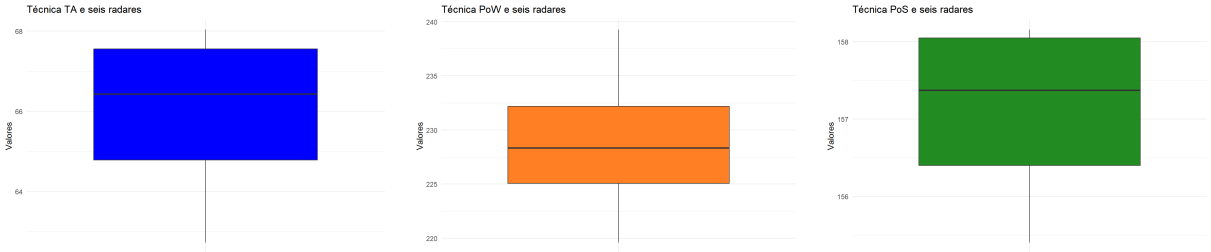
Tabela 23 – Médias dos grupos. O grupo TA tem a menor média.

Grupo	Índice	Média
TA	1	68.89153
PoW	2	212.7212
PoS	3	182.1885

Fonte: O autor (2024).

Os resultados da Tabela 23, indicam que o grupo TA tem a menor média (68.89153), seguido pelo grupo PoS (182.1885) e o grupo PoW (212.7212). Todas as diferenças entre as médias dos grupos são estatisticamente significativas.

Figura 49 – Análise de Variância para Infraestrutura com 6 Antenas



(a) TA - Infra 06 (b) PoW - Infra 06 (c) PoS - Infra 06

Fonte: O autor (2024).

A análise de variância apresentadas nas Figuras, 49a, 49b, 49c revelaram as diferenças significativas nos valores médios entre os grupos TA, PoW, PoS e outros não especificados. As comparações múltiplas de Tukey confirmaram que todos os grupos diferem significativamente entre si. O grupo TA apresentou a menor média, enquanto o grupo PoW apresentou a maior média.

6.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

O Capítulo 6 demonstrou os resultados das análises com o SIMA-IoV descritos no capítulo 4, conforme os cenários dos ambientes examinados. Foram apresentados os resultados de cada método e as relações com os parâmetros e requisitos de cada cenário, confrontados com as

métricas de desempenho do tempo de autenticação, vazão e atrasos durante as comunicações no ambiente IoV simulado.

Os resultados da análise estatística revelaram diferenças significativas nos valores médios entre os grupos TA, PoW e PoS. A ANOVA indicou que a variação entre os grupos é estatisticamente significativa ($F(2, 9) = 605.6$, $p < 0.001$), evidenciando que pelo menos um dos grupos difere dos outros em termos de médias. As comparações múltiplas de Tukey corroboraram essas diferenças, demonstrando que o grupo TA apresenta consistentemente a menor média (68.89153), seguido pelos grupos PoS (182.1885) e PoW (212.7212). Os resultados obtidos são essenciais para compreender as discrepâncias dos métodos de autenticação analisados, evidenciando a importância de da compreensão dos requisitos para diferentes contextos da Internet dos Veículos (IoV) em redes 5G.

7 CAPÍTULO CONCLUSÃO

7.1 CONSIDERAÇÕES FINAIS

Este trabalho demonstrou que o mercado de veículos conectados tem um papel vital na consolidação da Internet dos Veículos (IoV), facilitando a conectividade entre veículos, infraestrutura de transporte inteligente (ITS) e dispositivos de pedestres, além de fornecer acesso a serviços de Internet e computação em nuvem. Contudo, para que a IoV atinja todo o seu potencial, especialmente no contexto das cidades inteligentes, é essencial implementar estratégias robustas de segurança e privacidade. A autenticação é um elemento chave nesse contexto, garantindo que apenas veículos autorizados acessem a IoV, sem comprometer a qualidade dos serviços, como streaming de vídeo para passageiros e atualizações remotas de software automotivas (OTA). Esta tese analisou diferentes estratégias de autenticação para redes móveis de quinta geração (5G-V2X), incluindo abordagens centralizadas baseadas em autoridades de confiança (TA) e soluções distribuídas com *blockchain*, utilizando os algoritmos de consenso PoW (Proof of Work) e PoS (Proof of Stake).

Foi desenvolvida uma arquitetura em camadas para a IoV, integrando segurança através da computação de borda e nevoeiro, para isto foi desenvolvido um mecanismo inteligente para a escolha do método de autenticação utilizando otimização bayesiana. Os componentes e mecanismos desenvolvidos foram implementados e avaliados em simuladores de eventos discretos e de redes veiculares, utilizando dados reais de tráfego obtidos de sensores posicionados na Av. Agamenon Magalhães, Recife-PE, Brasil. Os resultados das simulações indicaram que a TA apresentou os melhores tempos de autenticação, variando entre 62-84ms, vazão média entre 377-697Kbps e atraso médio total de 5,1ms, sendo mais apropriada para cenários de baixa densidade de veículos. Por outro lado, a blockchain baseada em PoW mostrou-se instável, com tempos de autenticação elevados e dispersos (157-239ms), vazões entre 29-68Kbps e média de atraso de 4,8ms, especialmente em cenários de alta demanda. A blockchain baseada em PoS demonstrou um desempenho mais equilibrado e otimizado para diferentes cenários da IoV, com tempos de autenticação entre 127-192ms, vazões de 27-50Kbps e um atraso médio de 4,6ms.

A abordagem centralizada (TA) se mostrou mais eficiente em cenários de baixa densidade, enquanto a solução distribuída baseada em PoS oferece um desempenho mais estável e otimizado para variados cenários, incluindo aqueles com alta densidade de veículos e infraestrutura.

Em contraste, o PoW reforçou sua dificuldade para atender às exigências dos requisitos da IoV devido à sua instabilidade. Em resumo, este trabalho contribuiu para o avanço e compreensão dos impactos dos serviços de autenticação na Internet dos Veículos, apresentando uma solução eficaz SIMA-IoV, para a seleção do melhor método de autenticação de veículos em dado cenário, essenciais para o desenvolvimento seguro e sustentável das cidades inteligentes.

7.2 LIMITAÇÕES DA PESQUISA

Nossa pesquisa enfrentou algumas limitações relacionadas às ferramentas de simulação utilizadas com relação aos cenários simulados. As restrições das ferramentas de simulação e as bibliotecas ns-3, 5GLena, OMNeT++, INET, SIMU5G, Veins e SUMO impuseram desafios significativos tanto para integração na arquitetura quanto da sua complexidade para criar os cenários IoV. A pesquisa considerou um provedor de serviços de rede IoV, cenários mais complexos envolvendo múltiplas operadoras compartilhando recursos e links de rede para serviços de autenticação federados não foram explorados, a inclusão destes cenários confederados poderia fornecer uma visão mais abrangente e realista das operações distribuídas na IoV em ambientes multifacetados.

Além disso, o ambiente computacional utilizado para a simulação apresentou restrições de uso. A capacidade de processamento e a memória disponíveis limitaram a escala e tais restrições podem ter impactado nas análises e especialmente nos cenários com um número elevado de veículos e infraestrutura 5G densamente populada.

Uma limitação desta pesquisa é a não inclusão de outras tecnologias de redes sem fio, como 4G, WiFi 6, Bluetooth, LoRa e ZigBee, nas simulações e análises realizadas. Embora essas tecnologias sejam relevantes no contexto da Internet dos Veículos (IoV) e possam impactar a escolha e a eficácia dos métodos de autenticação, o foco desta pesquisa foi exclusivamente nas redes 5G devido ao seu papel emergente e seu potencial para suportar as altas demandas de conectividade e baixa latência necessárias para os cenários de veículos conectados.

Em resumo, a pesquisa enfrentou desafios devido às limitações acima mencionadas, superar estas limitações por meio de futuras investigações será crucial para obter uma compreensão mais completa e precisa das soluções de autenticação na Internet dos Veículos.

7.3 TRABALHOS PRODUZIDOS

- A Survey on Blockchain and Edge Computing applied to the Internet of Vehicles - IEEE International Conference on Advanced Networks and Telecommunications Systems - Workshop on New Advances on Vehicle-to-Everything (V2X) Communications and Networking, 2020 (QUEIROZ et al., 2020);
- Uma Proposta Open Source de Baixo Custo para Proteção e Geolocalização de Bebês em Veículos Automotores (SafeBaby) - Computer on the Beach, 2021 (BARBOSA et al., 2021);
- Autenticação com suporte à Computação de Borda 5G para a Internet de Veículos – WBlockchain SBRC, 2022 (QUEIROZ et al., 2023);
- Aero5GBS - Deep Learning-empowered ground users handover in Aerial 5G and Beyond Systems, IEEE Access Journal, 2023 (QUEIROZ; BARBOSA; DIAS, 2023).

7.4 TRABALHOS FUTUROS

Embora a presente pesquisa de tese tenha alcançado uma série de avanços relacionados à busca e seleção dos métodos de autenticação centralizados ou distrbuidos utilizando a TA ou a BC, há ainda uma série de oportunidades no que tange o ambiente complexo das IoVs e suas formas e modos de atuação.

Um avanço importante para esta pesquisa no futuro seria uma avaliação quantitativa dos recursos necessários, como CPU, memória, capacidade de processamento e requisitos da tecnologia de comunicação, nas diversas camadas arquiteturais que compõe o ambiente da IoV. Tal estudo, visa determinar a viabilidade prática da implantação de cenários e métodos específicos em uma rede veicular. A análise detalhada das capacidades computacionais e da rede fornecerá *insights* valiosos para a implementação eficaz de sistemas distribuídos baseados em *Blockchain* no ambiente IoV.

Além disso, a implementação do mecanismo SIMA-IoV em um ambiente real, por determinada operadora ou distribuidora de redes veiculares é essencial para uma análise mais precisa dos resultados. Comparar os dados obtidos em ambientes simulados com aqueles coletados em condições reais permitirá uma evolução dos parâmetros de análise do mecanismo. Esta abordagem visa ajustar o SIMA-IoV para uma maior compreensão de ambientes reais complexos

de serviços quem podem resultar em diferenças arquiteturais e de infraestruturas específicas de cada corporação. Tal comparação é importante para o aprimoramento do mecanismos e dos métodos de autenticação na Internet dos Veículos (IoV) em redes 5G.

Por fim, criar um sistema híbrido de autenticação que combine as vantagens da *Trust Authority* (TA) centralizada com as capacidades da *Blockchain* distribuída. Buscando explorar a eficiência e a rapidez da TA em ambientes com baixa latência e alta velocidade de processamento, ao mesmo tempo, em que se beneficiará da segurança, transparência e resiliência da *Blockchain* em cenários que requerem menor tempo de atraso e maior resistência a ataques e manipulações de dados.

REFERÊNCIAS

AL-SHAREEDA, M. A.; ANBAR, M.; MANICKAM, S.; HASBULLAH, I. H. Towards identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *IEEE Access*, v. 9, p. 113226–113238, 2021.

ALAM, M. S. K. M.; SADDIK, A. E. Toward social internet of vehicles: Concept, architecture, and applications. *IEEE ACCESS*, v. 3, p. 343–357, 2015. Doi: 10.1109/ACCESS.2015.2416657.

ALAM, M. S. K. M.; SADDIK, A. E. Internet of vehicles: An introduction. *International Journals of Advanced Research in Computer Science and Software Engineering*, v. 8, n. 1, p. 10–13, 2018. Doi: 10.23956/ijarcsse.v8i1.512.

ALAZEMI, F.; AL-MULLA, A.; AL-AKHRAS, M.; ALAWAIRDHI, M.; AL-MASRI, M.; OMAR, H.; ALSHAREEF, H. A trust management model in internet of vehicles. *International Journal of Data and Network Science*, v. 7, n. 2, p. 745–756, 2023.

ALI, A. H. I.; LI, F. Authentication and privacy schemes for vehicular ad hoc networks (vanets): A survey. *Vehicular Communications*, v. 16, p. 45–61, 2019. Doi: 10.1016/j.vehcom.2019.02.002.

ANIND, K. D.; GREGORY, D. A.; DANIEL, S. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction*, Taylor & Francis, v. 16, n. 2-4, p. 97–166, 2001. Disponível em: <https://doi.org/10.1207/S15327051HCI16234_02>.

ATZORI, A. I. L.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, p. 2787–2805, 2010. Doi: 10.1016/j.comnet.2010.05.010.

AYDIN, Y.; KURT, G. K.; OZDEMIR, E.; YANIKOMEROGLU, H. A flexible and lightweight group authentication scheme. *IEEE Internet of Things Journal*, v. 7, n. 10, p. 10277–10287, 2020.

BACK, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. [S.l.]: Oxford university press, 1996.

BALFE STEPHEN WOLTHUSEN, S. R. S. Robust and efficient communication overlays for trust authority computations. *IEEE Sarnoff Symposium*, 2009. Doi: 10.1109/SARNOF.2009.4850380.

BARBOSA, M.; QUEIROZ, A.; LIRA, W. L.; DIAS, K.; OLIVEIRA, E. Uma proposta open source de baixo custo para proteção e geolocalização de bebês em veículos automotores (safebaby). *Anais do Computer on the Beach*, v. 12, p. 095–101, 2021.

BARZEGAR, H. R.; IOINI, N. E.; LE, V. T.; PAHL, C. Wireless network evolution towards service continuity in 5g enabled mobile edge computing. In: *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.: s.n.], 2020. p. 78–85.

BATISTA, C.; SILVA, C. Um processo criativo de elicitação de contextos para sistemas sensíveis ao contexto. *SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO*, v. 11, p. 323–330, 2015. Doi: 10.5753/sbsi.2015.5833.

BROWNLEE, J. *Probability for machine learning: Discover how to harness uncertainty with Python*. [S.l.]: Machine Learning Mastery, 2019.

BRUNNER, C.; KNIRSCH, F.; UNTERWEGER, A.; ENGEL, D. A comparison of blockchain-based pki implementations. In: *ICISSP*. [S.l.: s.n.], 2020. p. 333–340.

BUCHMANN, J. *Introduction to Cryptography*. [S.l.]: Springer Science Business Media, 2004. v. 2.

CASTILLO, S. Z. J.; IBAñEZ, J. A. Context-aware data-driven intelligent framework for fog infrastructures in internet of vehicles. *IEEE ACCESS*, v. 6, p. 1–13, 2018. Doi: 10.1109/ACCESS.2018.2874592.

CASTILLO, S. Z. J.; IBAñEZ, J. A. Internet of vehicles: Architecture, protocols, and security. *IEEE INTERNET OF THINGS JOURNAL*, v. 5, n. 5, p. 3701–3709, 2018. Doi: 10.1109/JIOT.2017.2690902.

DAI, Y.; XU, D.; ZHANG, K.; MAHARJAN, S.; ZHANG, Y. Permissioned blockchain and deep reinforcement learning for content caching in vehicular edge computing and networks. In: *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. [S.l.: s.n.], 2019. p. 1–6.

DAI, Y.; XU, D.; ZHANG, K.; MAHARJAN, S.; ZHANG, Y. Permissioned blockchain and deep reinforcement learning for content caching in vehicular edge computing and networks. In: *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. [S.l.: s.n.], 2019. p. 1–6.

DOLUI, K.; DATTA, S. K. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In: *2017 Global Internet of Things Summit (GIoTS)*. [S.l.: s.n.], 2017. p. 1–6.

EZHILARASAN, E.; DINAKARAN, M. A review on mobile technologies: 3g, 4g and 5g. In: *2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM)*. [S.l.: s.n.], 2017. p. 369–373.

FU, Y.; YU, F. R.; LI, C.; LUAN, T. H.; ZHANG, Y. Vehicular blockchain-based collective learning for connected and autonomous vehicles. *IEEE Wireless Communications*, v. 27, n. 2, p. 197–203, 2020.

GAO, J.; AGYEKUM, K. O.-B. O.; SIFAH, E. B.; ACHEAMPONG, K. N.; XIA, Q.; DU, X.; GUIZANI, M.; XIA, H. A blockchain-sdn-enabled internet of vehicles environment for fog computing and 5g networks. *IEEE Internet of Things Journal*, v. 7, n. 5, p. 4278–4291, 2020.

GHOSH TANAY BHARTIA, S. K. A. S. C. B. C. Leveraging public-private blockchain interoperability for closed consortium interfacing. *IEEE Conference on Computer Communications*, 2021. Doi: 10.1109/INFOCOM42981.2021.9488683.

GOPE, P. Laap: Lightweight anonymous authentication protocol for d2d-aided fog computing paradigm. *Computer Security*, v. 83, p. 223–237, 2019. Doi: 10.1016/j.cose.2019.06.003.

GRANDISON, T.; SLOMAN, M. A survey of trust in internet applications. *IEEE Communications Surveys Tutorials*, v. 3, p. 2–16, 2000. Doi: 10.1109/COMST.2000.5340804.

GUO, S.; HU, X.; ZHOU, Z.; WANG, X.; QI, F.; GAO, L. Trust access authentication in vehicular network based on blockchain. *China Communications*, v. 16, n. 6, p. 18–30, 2019.

GUO, S.; HU, X.; ZHOU, Z.; WANG, X.; QI, F.; GAO, L. Trust access authentication in vehicular network based on blockchain. *China Communications*, v. 16, n. 6, p. 18–30, 2019.

GUPTA, A.; JHA, R. K. A survey of 5g network: Architecture and emerging technologies. *IEEE European Solid State Circuits Conference*, v. 13, p. 1206–1232, 2015. Doi: 10.1109/ACCESS.2015.2461602.

HARTENSTEIN, H.; LABERTEAUX, K. P. A tutorial survey on vehicular ad hoc networks. *IEEE Communications Magazine*, v. 46, p. 164–171, 2008. Doi: 10.1109/M-COM.2008.4539481.

HU, W.; HU, Y.; YAO, W.; LI, H. A blockchain-based byzantine consensus algorithm for information authentication of the internet of vehicles. *IEEE Access*, v. 7, p. 139703–139711, 2019.

HUANG, Q.; LI, N.; ZHANG, Z.; YANG, Y. Secure and privacy-preserving warning message dissemination in cloud-assisted internet of vehicles. In: *2019 IEEE Conference on Communications and Network Security (CNS)*. [S.l.: s.n.], 2019. p. 1–8.

JAVED KIRIL ANTEVSKI, J. M. L. G. F.; BERNARDOS, C. Distributed ledger technologies for network slicing: A survey. *IEEE Access*, v. 10, p. 19412–19442, 2022. Doi: 10.1109/ACCESS.2022.3151150.

JIANG, T.; FANG, H.; WANG, H. Blockchain-based internet of vehicles: Distributed network architecture and performance analysis. *IEEE Internet of Things Journal*, v. 6, n. 3, p. 4640–4649, 2019.

KANG, J.; YU, R.; HUANG, X.; WU, M.; MAHARJAN, S.; XIE, S.; ZHANG, Y. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet of Things Journal*, v. 6, n. 3, p. 4660–4670, 2019.

KRAUS, M. Using bayesian optimization to reduce the time spent on hyperparameter tuning. *gitgub. com*, 2019.

LI, H.; PEI, L.; LIAO, D.; SUN, G.; XU, D. Blockchain meets vanet: An architecture for identity and location privacy protection in vanet. *Peer-to-Peer Networking and Applications*, Springer, v. 12, p. 1178–1193, 2019.

LIANHAI, L.; YUJUE, W.; JINGWEI, Z.; QING, Y. A secure and efficient group key agreement scheme for vanet. *Sensors*, v. 19, p. 1–14, 2019. Doi: 10.3390/s19030482.

LIU, H.; ZHANG, Y.; YANG, T. Blockchain-enabled security in electric vehicles cloud and edge computing. *IEEE Network*, v. 32, n. 3, p. 78–83, 2018.

LIU, Y.; WANG, W.; ZHANG, M. Bayesian optimization for hyperparameter tuning in machine learning: A practical tutorial. *Journal of Machine Learning Research*, JMLR, v. 22, p. 1–20, 2021.

MAIO, V. D.; URIARTE, R. B.; BRANDIC, I. Energy and profit-aware proof-of-stake offloading in blockchain-based vanets. In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*. [S.l.: s.n.], 2019. p. 177–186.

MATTISSON, E. R. S. Overview of 5g requirements and future wireless networks. *IEEE European Solid State Circuits Conference*, v. 43, p. 5090–5025, 2017. Doi: 10.1109/ESSCIRC.2017.8094511.

MOHAMMADI, A.; SHEIKHOLESLAM, F. Intelligent optimization: Literature review and state-of-the-art algorithms (1965–2022). *Engineering Applications of Artificial Intelligence*, v. 126, p. 106959, 2023. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197623011430>.

MONTGOMERY, D. C. *Design and Analysis of Experiments*. 9th. ed. [S.l.]: John Wiley & Sons, 2017.

MURPHY, K. P. *Probabilistic machine learning: an introduction*. [S.l.]: MIT press, 2022.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. *https://bitcoin.org/bitcoin.pdf*, 2009.

NARDINI DARIO SABELLA, G. S. P. T. G.; VIRDIS, A. Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks. *IEEE Access*, v. 8, 2020. Doi: 10.1109/ACCESS.2020.3028550.

NARDINI GIOVANNI STEA, A. V. G.; SABELLA, D. Simu5g: A system-level simulator for 5g networks. *Proceedings of the 10th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, 2020. Doi: 10.5220/0009826400680080.

NISHU, G.; RAVIKANTI, M. et al. Authentication-based secure data dissemination protocol and framework for 5g-enabled vanet. *Future Internet*, v. 12, p. 63, 2020. Doi: 10.3390/fi12040063.

NúñEZ-GóMEZ, C.; CAMINERO, B.; CARRIóN, C. Hidra: A distributed blockchain-based architecture for fog/edge computing environments. *IEEE Access*, v. 9, p. 75231–75251, 2021.

OLIVEIRA, L. A. N.; ALENCAR, M. S.; LOPES, W. A. T. Evolução da arquitetura de redes móveis rumo ao 5g. *Revista de Tecnologia da Informação e Comunicação*, v. 8, n. 2, p. 43–50, Oct 2018. ISSN 2237-5104. Disponível em: <http://rtic.com.br/index.php/rtic/article/view/104>.

OMNETPP. Omnet++ documentation. *Portal OMNeT++*, 2023. Https://omnetpp.org/documentation/.

OMNETPP, I. N. What is inet framework. *Portal INET Framework OMNeTpp*, 2023. Https://inet.omnetpp.org/Introduction.html.

OMONIWA, B.; HUSSAIN, R.; JAVED, M. A.; BOUK, S. H.; MALIK, S. A. Fog/edge computing-based iot (feciot): Architecture, applications, and research issues. *IEEE Internet of Things Journal*, v. 6, n. 3, p. 4118–4149, 2019.

OSORIO, D. P. M.; AHMAD, I.; SáNCHEZ, J. D. V.; GURTOV, A.; SCHOLLIERS, J.; KUTILA, M.; PORAMBAGE, P. Towards 6g-enabled internet of vehicles: Security and privacy. *IEEE Open Journal of the Communications Society*, v. 3, p. 82–105, 2022.

PANWAR, N.; SHARMA, S.; SINGH, A. K. A survey on 5g: The next generation of mobile communication. *Physical Communication*, Elsevier, v. 18, p. 64–84, 2016.

POORZARE, R.; AUGé, A. C. Challenges on the way of implementing tcp over 5g networks. *IEEE Access*, v. 8, p. 176393–176415, 2020.

QIU, C.; YU, F. R.; XU, F.; YAO, H.; ZHAO, C. Blockchain-based distributed software-defined vehicular networks via deep q-learning. In: . New York, NY, USA: Association for Computing Machinery, 2018. p. 8–14. ISBN 9781450359641. Disponível em: <https://doi.org/10.1145/3272036.3272040>.

QUEIROZ, A.; OLIVEIRA, E.; BARBOSA, M.; DIAS, K. A survey on blockchain and edge computing applied to the internet of vehicles. In: *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. [S.l.: s.n.], 2020. p. 1–6.

QUEIROZ, A. A. L.; BARBOSA, M. K. S.; DIAS, K. L. Aero5gbs—deep learning-empowered ground users handover in aerial 5g and beyond systems. *IEEE Access*, v. 11, p. 120449–120462, 2023.

QUEIROZ, A. A. L.; OLIVEIRA, E. H. A. M. M.; SANTANA, M. K. de; LIRA, V. C. C. de; DIAS, K. L. Autenticação com suporte à computação de borda 5g para a internet de veículos. *Brazilian Journal of Development*, v. 9, n. 05, p. 14613–14631, 2023.

QURESHI GWANGGIL JEON, F. P. K. Anomaly detection and trust authority in artificial intelligence and cloud computing. *Computer Networks*, v. 184, p. 107647, 2021. Doi: 10.1016/j.comnet.2020.107647.

RASHA, A.-R. M. A.-M.; YUAN, T. Improving vehicular authentication in vanet using cryptography. *International Journal of Communication Networks and Information Security*, v. 10, p. 248–255, 2018.

RATHORE, H.; SAMANT, A.; JADLIWALA, M.; MOHAMED, A. Tanglecv: Decentralized technique for secure message sharing in connected vehicles. In: *Proceedings of the ACM Workshop on Automotive Cybersecurity*. New York, NY, USA: Association for Computing Machinery, 2019. p. 45–48. ISBN 9781450361804. Disponível em: <https://doi.org/10.1145/3309171.3309177>.

RAZALI, N. F.; ISA, I. S.; SULAIMAN, S. N.; KARIM, N. K. A.; OSMAN, M. K.; SOH, Z. H. C.; YUSOFF, Z. M. A comparative performance of genetic algorithm and bayesian optimization for hyperparameter tuning for mammogram classification. In: *2023 IEEE 13th International Conference on Control System, Computing and Engineering (ICCSCE)*. [S.l.: s.n.], 2023. p. 173–178.

REIDT, S. D. W. S. Efficient distribution of trust authority functions in tactical networks. *IEEE Information Assurance and Security Workshop*, p. 84–91, 2007. Doi: 10.1109/IAW.2007.381918.

REIDT, S. D. W. S. Efficient distribution of trust authority functions in tactical networks. *IEEE Military Communications Conference*, p. 1–7, 2007. Doi: 10.1109/MILCOM.2007.4454880.

RIMBA, P.; TRAN, A. B.; WEBER, I.; STAPLES, M.; PONOMAREV, A.; XU, X. Quantifying the cost of distrust: Comparing blockchain and cloud services for business process execution. *Information Systems Frontiers*, Springer, v. 22, n. 2, p. 489–507, 2020.

SAMPSON, J. R. *Adaptation in natural and artificial systems (John H. Holland)*. [S.l.]: Society for Industrial and Applied Mathematics, 1976.

SEUIL, D. D. Blockchain in the flemish government. *Informatie Vlaanderen*, 2018. Https://www.vlaanderen.be/informatievlaanderen.

SILBERSCHATZ HENRY F. KORTH, S. S. A. *DATABASE SYSTEM CONCEPTS*. [S.l.]: MGH Education, 2011. v. 7. 1266 p. ISBN: 978-0-07-802215-9.

SOMMER, R. G. C.; DRESSLER, F. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, v. 10, 2011. Doi: 10.1109/TMC.2010.133.

SUMO. Sumo documentation. *Portal Sumo*, 2023. Https://sumo.dlr.de/docs/.

SUNYAEV, A. *Distributed ledger technology*. [S.l.]: Springer in Internet Computing, 2020. v. 1. 265–299 p.

TAHIR, S. N. F.; KHALID, Z. Privacy-preserving authentication protocol based on hybrid cryptography for vanets. *International Conference on Applied and Engineering Mathematics (ICAEM)*, p. 80–85, 2019. Doi: 10.1109/ICAEM.2019.8853808.

TAN, H.; CHUNG, I. Secure authentication and key management with blockchain in vanets. *IEEE Access*, v. 8, p. 2482–2498, 2020.

TAN, I. C. H. Secure authentication and key management with blockchain in vanets. *IEEE Access*, v. 8, p. 2482–2498, 2020. Doi: 10.1109/ACCESS.2019.2962387.

TOSKALA, H. H. A.; NAKAMURA, T. *5G Technology: 3GPP New Radio, First Edition*. [S.l.]: John Wiley Sons Ltd., 2020. v. 1.

TU, S.; YU, H.; BADSHAH, A.; WAQAS, M.; HALIM, Z.; AHMAD, I. Secure internet of vehicles (iov) with decentralized consensus blockchain mechanism. *IEEE Transactions on Vehicular Technology*, v. 72, n. 9, p. 11227–11236, 2023.

VEINS.ORG. Veins documentation. *Portal Veins*, 2023. Https://veins.car2x.org/documentation/.

WANG, Y.; SU, Z.; ZHANG, K.; BENSLIMANE, A. Challenges and solutions in autonomous driving: A blockchain approach. *IEEE Network*, v. 34, n. 4, p. 218–226, 2020.

WINSLETT, M. An introduction to trust negotiation. *Springer - Trust Management, LNCS 2692*, p. 275–283, May 2003. Doi: 10.1145/2757384.2757397.

YAGA PETER MELL, N. R. D.; SCARFONE, K. *Blockchain Technology Overview*. [S.l.]: NIST National Institute of Standards and Technology, 2011. v. 8202. 66 p. Doi: 10.6028/NIST.IR.8202.

YAHIATENE, Y.; RACHEDI, A.; RIAHLA, M. A.; MENACER, D. E.; NAIT-ABDESSELAM, F. A blockchain-based framework to secure vehicular social networks. *Transactions on emerging telecommunications technologies*, Wiley Online Library, v. 30, n. 8, p. e3650, 2019.

YANG, F.; WANG, S.; LI, J.; LIU, Z.; SUN, Q. An overview of internet of vehicles. *China Communications*, v. 11, n. 10, p. 1–15, 2014.

YE, A. *The Beauty of Bayesian Optimization, Explained in Simple Terms*. 2022. Acesso em: Dezembro de 2023. Disponível em: <https://towardsdatascience.com/the-beauty-of-bayesian-optimization-explained-in-simple-terms-81f3ee13b10f>.

YERRAMILLI, R.; SWAMY, D. N. K. A comparative study of traditional authentication and authorization methods with block chain technology for egovernance services. *JournalNX*, Novateur Publication, p. 149–154, 2019.

ZHANG, H.; LAI, Y.; CHEN, Y. Authentication methods for internet of vehicles based on trusted connection architecture. *Simulation Modelling Practice and Theory*, Elsevier, v. 122, p. 102681, 2023.

ZHANG, X.; LI, R.; CUI, B. A security architecture of vanet based on blockchain and mobile edge computing. In: *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. [S.l.: s.n.], 2018. p. 258–259.

ZHANG, X.; LI, R.; CUI, B. A security architecture of vanet based on blockchain and mobile edge computing. In: *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. [S.l.: s.n.], 2018. p. 258–259.

ZHAO, D. O. S. Applying blockchain layer2 technology to mass e-commerce. *IEEE Conference on Computer Communications*, 2020.

ZHAO, Y. et al. A verifiable hidden policy cp-abe with decryption testing scheme and its application in vanet. *Trans Emerging Tel Tech - John Wiley Son*, special, p. 1–17, 2019. Doi: 10.1002/ett.3785.

ZHENG, T.; WU, J.; LI, G. Iov data sharing scheme based on the hybrid architecture of blockchain and cloud-edge computing. *Journal of Cloud Computing*, Springer, v. 12, n. 1, p. 99, 2023.

ZIBIN, Z.; SHAOAN, X.; OTHRES. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, v. 105, p. 475–491, 2020. Doi: 10.1016/j.future.2019.12.019.

ANEXO A – CÓDIGO FONTE DA BLOCKCHAIN MINERADOR - PROOF OF WORK

```
#include "BlockchainApplication.h"
#define END_CODE "\033[0m"

#include "inet/applications/tcpapp/GenericAppMsg_m.h"
#include "inet/common/ModuleAccess.h"
#include "inet/common/TimeTag_m.h"
#include "inet/common/lifecycle/ModuleOperations.h"
#include "inet/common/packet/Packet.h"
#include "inet/common/ProtocolTag_m.h"
#include "inet/common/lifecycle/NodeStatus.h"
#include "inet/common/packet/Message.h"
#include "inet/common/packet/chunk/ByteCountChunk.h"
#include "inet/common/socket/SocketTag_m.h"
#include "inet/networklayer/common/L3AddressResolver.h"
#include "inet/transportlayer/contract/tcp/TcpCommand_m.h"
#include "inet/networklayer/contract/ipv4/Ipv4Address.h"
#include "inet/networklayer/contract/IInterfaceTable.h"

#include <omnetpp.h>

using namespace omnetpp;
using namespace std;
using namespace inet;
#define MSGKIND_CONNECT    0
#define MSGKIND_SEND       1

Define_Module(BlockchainApplication);

BlockchainApplication::BlockchainApplication(){}
BlockchainApplication::~BlockchainApplication(){}


void BlockchainApplication::initialize(int stage)
{
    cSimpleModule::initialize(stage);

    if (stage == INITSTAGE_LOCAL) {
        delay = par("replyDelay");
        maxMsgDelay = 0;

        // statistics
        msgsRcvd = msgsSent = bytesRcvd = bytesSent = 0;
```

```
44          WATCH(msgsRcvd);
            WATCH(msgsSent);
46          WATCH(bytesRcvd);
            WATCH(bytesSent);
48      }
        else if (stage == INITSTAGE_APPLICATION_LAYER) {
50          cout << "stage: " << stage << endl;
            int localPort = par("localPort");
52          string localAddress;
            string peerAddrServer;
54          cModule * host = getContainingNode(this);
            string edgeNode = host->getFullName();
56
            // Criando o Socket TCP e transformando ele em servidor ...
58          socket.setOutputGate(gate("socketOut"));
            const char* charArray = localAddress.c_str();
60          socket.bind(Ipv4Address(charArray), localPort); // local address/port
            socket.listen();
62
            // Criando o Socket TCP e transformando ele em servidor ...
64          peerSocketServer.setOutputGate(gate("socketOut"));
            const char* peerAddr = peerAddrServer.c_str();
66          peerSocketServer.bind(Ipv4Address(peerAddr), localPort); // local address
                /port
            peerSocketServer.listen();
68


70          cModule *node = findContainingNode(this);
            NodeStatus *nodeStatus = node ? check_and_cast_nullable<NodeStatus *>(
                node->getSubmodule("status")) : nullptr;
72          bool isOperational = (!nodeStatus) || nodeStatus->getState() ==
                NodeStatus::UP;
            if (!isOperational)
74              throw cRuntimeError("This module doesn't support starting in node
                    DOWN state");

76          startApplication();
        }
78  }

80  void BlockchainApplication::startApplication()
    {
82      cout << "Starting Blockchain Miner Application" << endl;


84
        //Create Genesis block
86      Block block = Block(1,0,currentTransactions,0,0,0, "0000");
```

```
    block.hash = "0000";
88  chain.push_back(block);

90  mVotesAmount = 1;

92  // Inicializando os par metros do ECDH:
    ECDHKey.KeyPairGeneration();
94  ECDHIv.KeyPairGeneration();
    Server_ECC.ECC_KeyGeneration();
96
    CryptoPP::StringSink sink(ServerKey_s);
98  Server_ECC.GetPublicKey().Save(sink);

100 ServerPrivKey = Server_ECC.GetPrivateKey();

102

    std::string filename = std::to_string(mNodeId) + "-chain.txt";
104 // cout << mNodeId << " - Writting in "<<filename<<". Interval = " <<
        interval<<endl;
    outfile.open(filename, std::ios_base::app);
106 outfile << "----------------------------------------\n";
    outfile << block.toString();
108 outfile.close();

110 cMessage *msg = new cMessage("schedule the inicialization of the
        Communication");
    msg->setKind(SETPOINT);
112 scheduleAt(simTime() + 0.2, msg);
}
114
void BlockchainApplication::StartCommunication(){
116         int peerPort = 1234;
            cout << "Initialization of the peer comunication ..." << endl;
118         cout << "Connecting to: " << peerAddrClient << ":" << peerPort << endl;
            peerSocketClient.setOutputGate(gate("socketOut"));
120         const char* peerAddr = peerAddrClient.c_str();
            peerSocketClient.connect(*peerAddr ? L3AddressResolver().resolve(peerAddr
                ) : L3Address(), peerPort);
122 }

124

void BlockchainApplication::sendRequest()
126 {
    long requestLength = par("requestLength");
128 long replyLength = par("replyLength");
    if (requestLength < 1)
130     requestLength = 1;
```

```
    if (replyLength < 1)
132         replyLength = 1;

134     Packet *packet = new Packet("data");
        const auto& payload = makeShared<AuthPacket>();
136     /**/
        payload->setPacketid(CHAIN);
138     payload->setChunkLength(B(requestLength));
        payload->setExpectedReplyLength(B(replyLength));
140
        payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
142     packet->insertAtBack(payload);
        /**/
144     EV_INFO << "sending request with " << requestLength << " bytes, expected
            reply length " << replyLength << " bytes\n";
        cout << endl;
146     peerSocketClient.send(packet);
}
148
void BlockchainApplication::SetupECDH(int NodeId, const char* mkeyECDH, const
    char* mivECDH){
150
    cout << "starting ECDH setup ... " << endl;
152      cout << "Generating the ECDH keys to share a secret with Node " << NodeId <<
             endl;
    /*
154 There are two shared secrets one is used as key and the other is used as iv
        on the
    aes cryptography, aiming a security channel.
156 */

158 ECDHKey.SetPubKeyB(mkeyECDH);
    ECDHKey.SharedSecretGeneration();
160 keyECDH = ECDHKey.GetSharedSecret();
    DataBase.writeECDHkey(NodeId, keyECDH);
162 Integer a;
    a.Decode(keyECDH.BytePtr(), keyECDH.SizeInBytes());
164 cout << "Shared Secret -- key (ECDH): " << std::hex << a << endl;

166 ECDHIv.SetPubKeyB(mivECDH);
    ECDHIv.SharedSecretGeneration();
168 ivECDH = ECDHIv.GetSharedSecret();
    DataBase.writeECDHiv(NodeId, ivECDH);
170 a.Decode(ivECDH.BytePtr(), ivECDH.SizeInBytes());
    cout << "Shared Secret -- iv  (ECDH): " << std::hex << a << endl;
172 }
```

```
174 void BlockchainApplication::SetupEcc(std::string privatekeyecc)
    {
176     const char* m_privkeyecc = privatekeyecc.c_str();
        ECC.SetPrivateKey(m_privkeyecc);
178 }

180 void BlockchainApplication::SharingPublicKeyEDCH(string mkeyECDH, string mivECDH,
         int mconnId, cMessage *msg)
    {
182     const auto& DHkey = makeShared<AuthPacket>();
        Packet *mPacket = new Packet(msg->getName(), TCP_C_SEND);
184     mPacket->addTag<SocketReq>()->setSocketId(mconnId);
        DHkey->setPacketid(SHARINGECDHKEYS);
186     DHkey->setChunkLength(B(200));
        DHkey->setExpectedReplyLength(B(200));
188     // DHkey->setServerClose(false);
        DHkey->setKey(mkeyECDH.c_str());
190     DHkey->setIv(mivECDH.c_str());
        DHkey->addTag<CreationTimeTag>()->setCreationTime(simTime());
192     mPacket->insertAtBack(DHkey);
        sendPacket(mPacket);
194 }

196 void BlockchainApplication::SendAuthenticadedKeys(int NodeId, int mconnId,
        cMessage *msg)
    {
198     cout << "Set up of the keys for the node " << NodeId << endl;

200     // Key Generation
        cout << "****** Inicializando ECC --- Node "<< NodeId <<" ******" << endl;
202
        socketIds.insert({NodeId, mconnId});
204
        EllipticCurveCryptography ECCKeys;
206     ECCKeys.ECC_KeyGeneration();
        DataBase.writeECC(NodeId, ECCKeys.GetPublicKey());
208     std::string privatekeyecc = ECCKeys.GetPrivateKey();
        cout << "Private Key: " << privatekeyecc <<endl;
210
        //Encrypt Ecc private key
212     std::string EccPrivateKeyCrypt = AES.EncryptionAES(privatekeyecc, DataBase.
            readECDHkey(NodeId), DataBase.readECDHiv(NodeId));

214     const auto& ECCkey = makeShared<AuthPacket>();
        Packet *mPacket = new Packet(msg->getName(), TCP_C_SEND);
216     mPacket->addTag<SocketReq>()->setSocketId(mconnId);
        ECCkey->setPacketid(AUTHENTICATIONANS);
```

```
218     ECCkey->setChunkLength(B(200));
        ECCkey->setExpectedReplyLength(B(200));
220     // ECCkey->setServerClose(false);

222     string HexEccPrivateKeyCrypt = localhex.string_to_hex(EccPrivateKeyCrypt);
        ECCkey->setKey(HexEccPrivateKeyCrypt.c_str());
224     /********ECC pub key*********/
        // Convertendo a chave p blica do servidor para enviar
226     string HexServerKey_s = localhex.string_to_hex(ServerKey_s);
        ECCkey->setData(HexServerKey_s.c_str());
228     ECCkey->addTag<CreationTimeTag>()->setCreationTime(simTime());
        mPacket->insertAtBack(ECCkey);
230     sendPacket(mPacket);

232     cout << "Keys sended to " << NodeId << endl;
        cout << "*****************************************************"<< endl;
234 }

236 void BlockchainApplication::seeIp()
    {
238     cModule * host = getContainingNode(this);
        //L3Address addr = L3AddressResolver().addressOf(host);
240
        string mNodeId = host->getFullName();
242     cout << "Hey, I am the server: "<< mNodeId << endl ;
    }
244
    void BlockchainApplication::sendOrSchedule(cMessage *msg, simtime_t delay)
246 {
        if (delay == 0)
248         sendPacket(msg);
        else
250         scheduleAfter(delay, msg);
    }
252


254 void BlockchainApplication::sendPacket(cMessage *msg)
    {
256     cout << "Server is sending a Packet ..." << endl;
        Packet *packet = dynamic_cast<Packet *>(msg);
258
        if (packet) {
260         msgsSent++;
            bytesSent += packet->getByteLength();
262         emit(packetSentSignal, packet);

264         EV_INFO << "sending \"" << packet->getName() << "\" to TCP, " << packet->
```

```
            getByteLength() << " bytes\n";
    }
    else {
        EV_INFO << "sending \"" << msg->getName() << "\" to TCP\n";
    }

    auto& tags = check_and_cast<ITaggedObject *>(msg)->getTags();
    tags.addTagIfAbsent<DispatchProtocolReq>()->setProtocol(&Protocol::tcp);
    send(msg, "socketOut");
}

void BlockchainApplication::handleMessage(cMessage *msg)
{
    cout << "SimTime(): " << simTime() << endl;

    if (msg->isSelfMessage()) {
        switch (msg->getKind())
        {
            case AUTHSUCCESS:{
                long requestLength = par("requestLength");
                long replyLength = par("replyLength");
                if (requestLength < 1)
                    requestLength = 1;
                if (replyLength < 1)
                    replyLength = 1;
                Packet *packet = new Packet("data");
                const auto& payload = makeShared<AuthPacket>();
                payload->setPacketid(AUTHSUCCESS);
                payload->setNodeid(mNodeId);
                payload->setBlock_id(to_send_bid.c_str());
                payload->setData("no-pub-key");
                payload->setChunkLength(B(requestLength));
                payload->setExpectedReplyLength(B(replyLength));

                payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
                packet->insertAtBack(payload);
                peerSocketClient.send(packet);

                cout << mNodeId << "- is sending packet " << AUTHSUCCESS << endl;
                break;
            }
            case SETPOINT:{
                StartCommunication();
                break;
            }
        }
    }
```

```
    else if (msg->getKind() == TCP_I_PEER_CLOSED) {
312     // we'll close too, but only after there's surely no message
        // pending to be sent back in this connection
314     int connId = check_and_cast<Indication *>(msg)->getTag<SocketInd>()->
            getSocketId();
        delete msg;
316     auto request = new Request("close", TCP_C_CLOSE);
        request->addTag<SocketReq>()->setSocketId(connId);
318     sendOrSchedule(request, delay + maxMsgDelay);
    }
320 else if (msg->getKind() == TCP_I_DATA || msg->getKind() == TCP_I_URGENT_DATA)
         {
        Packet *packet = check_and_cast<Packet*>(msg);
322     int connId = packet->getTag<SocketInd>()->getSocketId();
        ChunkQueue& queue = socketQueue[connId];
324     auto chunk = packet->peekDataAt(B(0), packet->getTotalLength());
        queue.push(chunk);
326     emit(packetReceivedSignal, packet);
        cout << endl;
328     cout << "*****************************" << endl;
        cout << mNodeId <<" - Recv a Packet: " << endl;
330
        PacketPrinter printer; // turns packets into human readable strings
332     printer.printPacket(std::cout, packet); // print to standard output
        cout << "*****************************" << endl;
334     const auto& mPayload = queue.pop<AuthPacket>(b(-1));
        int Id = mPayload->getPacketid();
336     cout<< mNodeId <<" - Packet ID = " << Id << endl;
        cout << "*****************************" << endl;
338     cout<<endl;

340     switch (Id)
        {
342         case AUTHENTICATIONREQ:{
                cout << "Authetication Request Recieved from node " << mPayload->
                    getNodeid() << endl;
344             cout << "Preparing the channel for authentication." << endl;

346             SetupECDH(mPayload->getNodeid(), mPayload->getKey(), mPayload->
                    getIv());
                SharingPublicKeyEDCH(ECDHKey.GetPubKey(), ECDHIv.GetPubKey(),
                    connId, msg);
348             break;
            }
350
            case CHANNELVERIFY:{
352
```

```
                cout << "Receiving the security data verification from Node  " <<
                     mPayload->getNodeid() << endl;
354
                string mData = mPayload->getData();
356             string hexData = localhex.hex_to_string(mData);
                string AgreedMessage = AES.DecryptionAES(hexData,DataBase.
                    readECDHkey(mPayload->getNodeid()), DataBase.readECDHiv(
                    mPayload->getNodeid()));
358             cout << "Verification Message Received: " <<AgreedMessage << endl
                    ;

360             if (AgreedMessage == "Security Channel")
                {
362                 cout << "The channel verification was completed." << endl;
                    // Send the user its ECC keys
364                 BlockchainApplication::SendAuthenticadedKeys(mPayload->
                        getNodeid(), connId, msg);
                    // this->StartCommunication();
366
                    CryptoPP::ECIES < ECC_ALGORITHM >::PublicKey pubKey = ECC.
                        GetPublicKey();
368
                    std::string encodedPubKey;
370                 StringSink saveSS(encodedPubKey);
                    pubKey.Save(saveSS);
372                 keysTransactionVector.clear();
                    keysTransactionVector.push_back(encodedPubKey);
374                 Block lastBlock = chain[chain.size()-1];
                    Block block = CreateBlock(lastBlock, keysTransactionVector,
                        1, mPayload->getNodeid());
376                 if (pendingBlockchainId != "" || (mConsensusBidSize > chain.
                        size())) {
                        cout << mNodeId <<" - there is a pending blockchain. Add
                            block to pending list" << endl;
378                     pendingBlocks.push_back(block);
                    } else {
380                     MineAndBroadcast(block);
                    }
382             }else{
                    cout << "Fail!" << endl;
384             }
                break;
386         }
            case CHAIN:{
388
                int strSenderId = mPayload->getNodeid();
390             cout << mNodeId << " - Node received chain from " << strSenderId
```

```
                << endl;

392             std::string receivingBid = mPayload->getBlock_id();
                std::string blockStr = mPayload->getBlock();
394
                std::vector<Block> newChain = stringToBlocks(blockStr);
396
                receivingChainPackets = true;
398             if (mConsensusList.find(receivingBid) != mConsensusList.end()) {
                if (mConsensusList[receivingBid].GetReachedConsensusFlag() &&
400                 !mConsensusList[receivingBid].GetVeredict()) {
                    // If I'm receiving a chain that was already rejected I can
                        ignore the packets
402                 mReceivingChainMap.erase(receivingBid);
                    receivingChainPackets = (mReceivingChainMap.size() != 0);
404                 break;
                }
406             }
                // if (newChain.size() == 0) break;
408             std::vector<Block> receivingChain = mReceivingChainMap[
                    receivingBid];

410             cout << mNodeId << " - Size of chain in current packet: " <<
                    newChain.size() << endl;
                cout << mNodeId << " - Size of receiving " << receivingBid << "
                    chain: " << receivingChain.size() << endl;
412             cout << mNodeId << " - Result received chain size: " << newChain.
                    size() << endl;
                createConsensusList(receivingBid, newChain);
414
                if (!mConsensusList[receivingBid].myVote) {
416                 bool veredict = isBlockchainValid(newChain);
                    if (veredict) {
418                     cout << mNodeId << "- losing a vote ..." << endl;
                        mVotesAmount--;
420                 }
                    sendVeredict(receivingBid, veredict);
422             }
                reachedConsensus(receivingBid, newChain);
424             mReceivingChainMap.erase(receivingBid);
                receivingChainPackets = (mReceivingChainMap.size() != 0);
426             break;
            }
428
            case VEREDICT: {
430
                std::string bid = mPayload->getBlock_id();
```

```
432                 createConsensusList(bid);
                    if (mConsensusList[bid].GetReachedConsensusFlag()) {
434                 cout << mNodeId << " - ignoring this veredict since "<<bid<<" was
                         already handled, veredict was "<<mConsensusList[bid].
                        GetVeredict() <<endl;
                    break;
436                 }

438                 int id = mPayload->getNodeid();
                    std::string strId = std::to_string(id);
440                 bool vote = mPayload->getVeredict_vote();
                    int sender_id = mPayload->getVeredict_senderid();
442                 std::string strSenderId = std::to_string(sender_id);
                    Veredict veredict(sender_id, vote);
444                 cout << mNodeId << " - Node " << mNodeId  << " received a
                        veredict " << veredict.GetSenderVote() << " from " +
                        strSenderId + " of blockchain "+bid<<endl;

446                 updateConsensusList(bid, veredict.GetSenderVote());
                    if (mConsensusList[bid].chain.empty()) {
448                 cout << mNodeId << " - Did not receive the chain yet, cant try to
                         update" <<endl;
                    } else {
450                     cout << mNodeId << "- checking if reached the consensus" <<
                            endl;
                        reachedConsensus(bid, mConsensusList[bid].chain);
452                 }
                    break;
454             }

456             case AUTHSUCCESS: {
                    std::string bid = mPayload->getBlock_id();
458                 cout << mNodeId << " - bid=" << bid << endl;
                    mConsensusList[bid].AddUpdateChain();
460                 updateChain(bid);
                    break;
462             }
        }
464
        bool doClose = false;
466     if (doClose) {
            doClose = true;
468         // break;
        }
470     delete msg;

472     if (doClose) {
```

```
            auto request = new Request("close", TCP_C_CLOSE);
474         TcpCommand *cmd = new TcpCommand();
            request->addTag<SocketReq>()->setSocketId(connId);
476         request->setControlInfo(cmd);
            sendOrSchedule(request, delay + maxMsgDelay);
478     }
    }
480 else if (msg->getKind() == TCP_I_AVAILABLE)
        socket.processMessage(msg);
482 else {
        // some indication -- ignore
484     EV_WARN << "drop msg: " << msg->getName() << ", kind:" << msg->getKind()
            << "(" << cEnum::get("inet::TcpStatusInd")->getStringFor(msg->getKind
            ()) << ")\n";
        delete msg;
486 }
}
488

Block BlockchainApplication::CreateBlock(Block last_block,
490                              std::vector<std::string> currentTransactions,
                                 int type,
492                              int mainParticipant) {
    int id = last_block.id+1;
494 int proof = last_block.proof;
    std::string previous_hash = last_block.hash;
496 simtime_t t1;
    t1 = simTime();
498 double time_ = t1.dbl();
    Block block = Block(
500     id,
        time_,
502     currentTransactions,
        proof,
504     type,
        mainParticipant,
506     previous_hash
    );
508 return block;
}
510

void BlockchainApplication::finish()
512 {
    EV_INFO << getFullPath() << ": sent " << bytesSent << " bytes in " <<
        msgsSent << " packets\n";
514 EV_INFO << getFullPath() << ": received " << bytesRcvd << " bytes in " <<
        msgsRcvd << " packets\n";
```

```
516     cout << getFullPath() << ": sent " << bytesSent << " bytes in " << msgsSent
            << " packets\n" << endl;
        cout << getFullPath() << ": received " << bytesRcvd << " bytes in " <<
            msgsRcvd << " packets\n" << endl;
518 }

520

    void BlockchainApplication::refreshDisplay() const
522 {
        char buf[100];
524     sprintf(buf, "rcvd: %ld pks %ld bytes\nsent: %ld pks %ld bytes", msgsRcvd,
            bytesRcvd, msgsSent, bytesSent);
        getDisplayString().setTagArg("t", 0, buf);
526 }

528

    void BlockchainApplication::MineAndBroadcast(Block block)
530 {
        int id = mNodeId;
532     Block lastBlock = chain[chain.size()-1];
        cout <<mNodeId << " - MineAndBroadcast. MyLastBlock is " << lastBlock.id <<
            endl;
534     Block blockToMine = CreateBlock(lastBlock, block.transactions, block.type,
            block.mainParticipant);
        // N o entendi para que o eduardo colocou esse time?
536     int timeToBroadcast = blockToMine.Mine();
        // cout <<mNodeId << " - Mined in "<<timeToBroadcast << endl;
538     std::vector<Block> newChain = chain;
        newChain.push_back(blockToMine);
540     // cout <<"creating bid" << endl;
        pendingBlockchainId = "Enb"+std::to_string(id) + "_block" + std::to_string(
            mBid);
542     cout <<pendingBlockchainId << " has size = " << newChain.size() << endl;
        BroadcastBlockchain(newChain);
544     //Simulator::Schedule(timeToBroadcast, &BlockchainApplication::
            BroadcastBlockchain, this, newChain);
    }
546

    void BlockchainApplication::BroadcastBlockchain(std::vector<Block> newChain) {
548

        std::string id = std::to_string(mNodeId);
550     cout << ": Broadcasting" << endl;
        std::string thisBid = "Enb"+ id + "_block" + std::to_string(mBid);
552     // just send last two block on the network
        std::vector<Block> chainToSend;
554

        if (newChain.size() == 2) {
```

```
556         chainToSend = newChain;
    } else {
558         chainToSend.push_back(newChain[newChain.size()-2]);
            chainToSend.push_back(newChain[newChain.size()-1]);
560     }

562     cout <<"Block0 = "<< chainToSend[0].toString() << endl;
    cout <<"Block1 = "<< chainToSend[1].toString() << endl;
564     createConsensusList(thisBid, chainToSend);

566     std::string m_data = this->chainToSendToString(chainToSend);

568     long requestLength = par("requestLength");
    long replyLength = par("replyLength");
570     if (requestLength < 1)
            requestLength = 1;
572     if (replyLength < 1)
            replyLength = 1;
574
    Packet *packet = new Packet("data");
576     const auto& payload = makeShared<AuthPacket>();
    payload->setPacketid(CHAIN);
578     payload->setNodeid(mNodeId);
    payload->setBlock(m_data.c_str());
580     payload->setBlock_id(thisBid.c_str());
    payload->setChunkLength(B(requestLength));
582     payload->setExpectedReplyLength(B(replyLength));
    payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
584     packet->insertAtBack(payload);
    peerSocketClient.send(packet);
586     cout << "A chain  foi enviada ... " << endl;
}
588
void BlockchainApplication::createConsensusList(std::string bid) {
590     if (mConsensusList[bid].totalNodes == -1) {
            mConsensusList[bid] = ConsensusList(TotalNodes - 1);
592     }
}
594
void BlockchainApplication::createConsensusList(std::string bid, std::vector<
    Block> chain) {
596     if (mConsensusList[bid].totalNodes == -1) {
    mConsensusList[bid] = ConsensusList(TotalNodes - 1, chain);
598     } else {
    mConsensusList[bid] = ConsensusList(mConsensusList[bid], chain);
600     }
}
```

```
602 void BlockchainApplication::updateConsensusList(std::string bid, bool veredict) {
    if (veredict) {
604 mConsensusList[bid].AddOk();
    cout << mNodeId << " - Updated Consensus List" <<endl;
606 } else {
    mConsensusList[bid].AddNok();
608 }
}
610

612 bool BlockchainApplication::isBlockchainValid(std::vector<Block> newChain) {
    cout<<mNodeId << " - isBlockchainValid" << endl;
614 Block newestBlock = newChain[newChain.size()-1];
    int lastIndex = chain.size()-1;
616 Block lastBlock = chain[lastIndex];

618 if (newestBlock.id <= lastBlock.id) {
    cout<<mNodeId << " - small size. newestBlock.id=" << newestBlock.id << ",
        lastBlock.id="<<lastBlock.id<< endl;
620 return false;
    }
622

    if (newestBlock.id <= mConsensusBidSize) {
624 cout<<mNodeId << " - Already reaching consensus on a blockchain of this size.
         newChain="<<newestBlock.id<<" oldChain="<<chain.size()<<",
        mConsensusBidSize="<<mConsensusBidSize<< endl;
    return false;
626 }

628 if (newestBlock.previous_hash != lastBlock.hash) {
    cout<<mNodeId << " - Disconnected newest block. newestBlock.previous_hash = "
        << newestBlock.previous_hash << ", lastBlock.hash: " << lastBlock.hash<<
        endl;
630 return false;
    }
632

    Block previousBlockOfNewChain = newChain[0];
634 if (newestBlock.IsInvalid() || previousBlockOfNewChain.IsInvalid()) {
        cout<<mNodeId << "Invalid chain"<< endl;
636     return false;
    }
638

    if (previousBlockOfNewChain.hash != lastBlock.hash ||
640     previousBlockOfNewChain.previous_hash != lastBlock.previous_hash) {
        cout<<mNodeId << " - Disconnected newest block. previousBlockOfNewChain.
            hash = "<< previousBlockOfNewChain.hash << ", lastBlock.hash: " <<
            lastBlock.hash<< endl;
```

```
642         cout<<mNodeId << " - previousBlockOfNewChain.previous_hash = "<<
                previousBlockOfNewChain.previous_hash << ", lastBlock.previous_hash:
                " << lastBlock.previous_hash<< endl;
            return false;
644     }

646     if (pendingBlockchainId != "") {
        std::vector<Block> pendingChain = mConsensusList[pendingBlockchainId].chain;
648     if (pendingChain[pendingChain.size()-1].timestamp <= newestBlock.timestamp) {
            cout<<mNodeId << " -   then current pendingChain last block "<< endl;
650         return false;
        }
652     }

654     if (mVotesAmount <= 0) {
        cout << mNodeId << " - not enough votes"<< endl;
656     return false;
        }
658
        return true;
660 }

662 void BlockchainApplication::sendVeredict(std::string bid, bool veredict) {
        mConsensusList[bid].SetMyVote(veredict);
664     cout << mNodeId << ": sendVeredict " << veredict << " to " << bid << "'s
            blockchain" << endl;

666     // this->StartCommunication();

668     long requestLength = par("requestLength");
        long replyLength = par("replyLength");
670     if (requestLength < 1)
            requestLength = 1;
672     if (replyLength < 1)
            replyLength = 1;
674
        Packet *packet = new Packet("data");
676     const auto& payload = makeShared<AuthPacket>();
        payload->setPacketid(VEREDICT);
678     payload->setNodeid(mNodeId);
        payload->setBlock_id(bid.c_str());
680     payload->setVeredict_vote(veredict);
        payload->setVeredict_senderid(mNodeId);
682     payload->setChunkLength(B(requestLength));
        payload->setExpectedReplyLength(B(replyLength));
684     payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
        packet->insertAtBack(payload);
```

```
686     peerSocketClient.send(packet);

688 }

690 void BlockchainApplication::updateChain(std::string bid) {
    cout << mNodeId << " - check "<< bid << " "<< mConsensusList[bid].updateChain
        <<"/"<<mConsensusList[bid].totalNodes << endl;
692     if (mConsensusList[bid].chain.empty() && mConsensusList[bid].
        HasEveryNodeUpdated()) {
    cout << "Empty chain, cannot update it right now."<<endl;
694     return;
    } else if (!mConsensusList[bid].chain.empty() && mConsensusList[bid].
        HasEveryNodeUpdated()) {
696
        chainsDoingConsensus--;
698         std::string filename = std::to_string(mNodeId) + "-chain.txt";
        mConsensusList[bid].ok = mConsensusList[bid].totalNodes;
700
        if (chain[chain.size()-1].id >= mConsensusList[bid].chain[1].id) {
702             handleLateBlock(bid);
            return;
704         }
        Block lastBlock = mConsensusList[bid].chain[mConsensusList[bid].chain.
            size()-1];
706         cout << mNodeId << " is updating its chain to " << bid << " with
            lastBlockId "<<lastBlock.id<<endl;
        chain.push_back(lastBlock);
708         outfile.open(filename, std::ios_base::app);
        outfile << lastBlock.toString();
710         outfile.close();

712     if (pendingBlockchainId == bid) {
        if (lastBlock.type == 1) {
714             cout << "Enviar para o usu rio o authsucss" <<endl;

716             int mconnId = socketIds[lastBlock.mainParticipant];

718             const auto& AuthSucess = makeShared<AuthPacket>();
            Packet *mPacket = new Packet("", TCP_C_SEND);
720             mPacket->addTag<SocketReq>()->setSocketId(mconnId);
            AuthSucess->setPacketid(AUTHSUCCESS);
722             AuthSucess->setChunkLength(B(200));
            AuthSucess->setExpectedReplyLength(B(200));
724             std::string encodedPubKey = lastBlock.transactions[0];
            AuthSucess->setData(encodedPubKey.c_str());
726             AuthSucess->addTag<CreationTimeTag>()->setCreationTime(simTime());
            mPacket->insertAtBack(AuthSucess);
```

```
728             sendPacket(mPacket);
                    }
730             }
            }
732         pendingBlockchainId = "";

734     }
        if (pendingBlockchainId == "" && pendingBlocks.size() > 0) {
736         Block pendingBlock = pendingBlocks[0];
            pendingBlocks.pop_front();
738         // Simulator::Schedule(interval, &BlockchainApplication::MineAndBroadcast
                , this, pendingBlock);
            MineAndBroadcast(pendingBlock);
740         }
        }
742 }

744 void BlockchainApplication::handleLateBlock(std::string bid) {
        cout<< mNodeId<<" - small id "<<bid <<endl;
746     std::string filename = std::to_string(mNodeId) + "-chain.txt";

748     int indexToInsert = mConsensusList[bid].chain[1].id -1;
        Block previousBlock = chain[indexToInsert-1];
750     Block nextBlock = chain[indexToInsert];
        Block blockToInsert = mConsensusList[bid].chain[1];
752     if (previousBlock.hash == blockToInsert.previous_hash &&
        nextBlock.previous_hash == blockToInsert.hash) {
754     cout << mNodeId << " - Late message block but it fits!" <<endl;
        chain.insert(chain.begin()+indexToInsert, blockToInsert);
756     outfile.open(filename);
        outfile << "----------------------------------------\n";
758     for (int i=0; i<chain.size();i++) {
            outfile << chain[i].toString();
760     }
        outfile.close();
762     }
        if (nextBlock.id == blockToInsert.id) {
764     cout << mNodeId <<" - Every node updated a chain that I did not update"<<endl
            ;
        int carId;
766     if (nextBlock.timestamp < blockToInsert.timestamp) {
            carId = chain[indexToInsert].mainParticipant;
768         cout << mNodeId <<" - their chain is better then mine. Gonna ask "<<carId
                <<" to try again"<<endl;
            chain[indexToInsert] = blockToInsert;
770     } else {
            carId = blockToInsert.mainParticipant;
```

```
772         cout << mNodeId <<" - my chain is better then theirs"<<endl;
            }
774     }
}
776
void BlockchainApplication::reachedConsensus(std::string bid, std::vector<Block>
    newChain) {
778
    if (mConsensusList[bid].IsConsensusReached()) {
780         chainsDoingConsensus++;
            if (mConsensusList[bid].myVote && mVotesAmount == 0) {
782             // After each consensus, if this node have voted to the blockchain we
                    add one more vote to it
                mVotesAmount = 1;
784         }
            int id = mNodeId;
786         if (newChain.empty()) {
                cout << "new chain is empty" << endl;
788             if (pendingBlockchainId == bid) {
                pendingBlockchainId = "";
790             }
                return;
792         }

794         Block lastBlock = newChain[newChain.size()-1];
            cout << id <<" reached consensus on " << bid << "'s Blockchain: "<<
                mConsensusList[bid].GetVeredict() << endl;
796
            if (mConsensusList[bid].GetVeredict()) {
798             cout << "entrei aqui ..." << endl;
                for (std::map<std::string,ConsensusList>::iterator it =
                    mConsensusList.begin(); it != mConsensusList.end(); ++it) {
800             ConsensusList list = it->second;
                if (!list.IsConsensusReached() && !list.chain.empty() && list.chain
                    [1].id <= newChain[1].id) {
802                 // Anul the rest of the chains since we've found a winner chain
                    list.nok = list.totalNodes;
804                 reachedConsensus(it->first, list.chain);
                }
806             }

808             to_send_bid = bid;

810             cMessage *sendAuthSuccess = new cMessage("Scheduling the AUTHSUCCESS
                    MSG");
                sendAuthSuccess->setKind(AUTHSUCCESS);
812             scheduleAt(simTime() + 0.001, sendAuthSuccess);
```

```
814             if (mConsensusList[bid].HasEveryNodeUpdated()) {
                    updateChain(bid);
816             }

818             return;
            } else {
820             if (pendingBlockchainId == bid) {
                mFailedChain++;
822             cout <<mNodeId << " - Block not added. Should try again. lastBlock =
                    " << lastBlock.toString()<<endl;

824             if ((mConsensusBidSize > chain[chain.size()-1].id)) {
                    cout <<mNodeId << " - since I already have a block for the "<<
                        mConsensusBidSize<<" position, I'll leave the mining for
                        later."<<endl;
826                 pendingBlocks.insert(pendingBlocks.begin(), lastBlock);
                    pendingBlockchainId = "";
828             } else if (mVotesAmount < 1) {
                    cout <<mNodeId << " - cant vote, so the network must be setting
                        something, lets wait a bit"<<endl;
830                 pendingBlocks.insert(pendingBlocks.begin(), lastBlock);
                    pendingBlockchainId = "";
832             } else if (mFailedChain == 3) {
                    cout <<mNodeId << " - since I already tried 3 times, I'll wait a
                        few seconds before mining again"<<endl;
834                 pendingBlockchainId = "";
                    mFailedChain = 0;
836                 // Simulator::Schedule(interval*2, &BlockchainApplication::
                        MineAndBroadcast, this, lastBlock);
                    this->MineAndBroadcast(lastBlock);
838             } else {
                    // Simulator::Schedule(interval, &BlockchainApplication::
                        MineAndBroadcast, this, lastBlock);
840                 MineAndBroadcast(lastBlock);
                }
842             } else if (pendingBlockchainId == "" && mVotesAmount > 0 &&
                    pendingBlocks.size() > 0 && chainsDoingConsensus <= 0) {
                    cout <<mNodeId << " - I have nothing going on but some blocks are
                         pending, lets try mining"<<endl;
844                 Block pendingBlock = pendingBlocks[0];
                    pendingBlocks.pop_front();
846                 // Simulator::Schedule(interval, &BlockchainApplication::
                        MineAndBroadcast, this, pendingBlock);
                    this->MineAndBroadcast(pendingBlock);
848                 }
            }
```

```
850     }else{
            cout<<"fail - aqui " <<endl;
852     }
        return;
854 }

856 std::string BlockchainApplication::chainToSendToString(const std::vector<Block>&
        chain) {
        std::string block_to_send;
858     for (const auto& block : chain) {

860         std::string result = "";
            int m_type = 0;
862
            result += "id: "+std::to_string(block.id)+
864                 "\ntype: "+std::to_string(block.type)+
                    "\nmainParticipant: "+std::to_string(block.mainParticipant)+
866                 ",\ntimestamp: "+std::to_string(block.timestamp)+
                    ",\nprevious_hash: "+block.previous_hash+
868                 ",\nproof/nonce: "+std::to_string(block.proof)+"\n";
            m_type = std::stoi(std::to_string(block.type));
870         std::vector<std::string> transactions = block.transactions;
            if (m_type == 1) {
872             std::string encoded;
                CryptoPP::StringSource ss(transactions[0], true,
874                 new CryptoPP::HexEncoder(
                        new CryptoPP::StringSink(encoded)
876                 ) // HexEncoder
                ); // StringSource
878             result += "Transaction 0: " + encoded + "\n";
            } else {
880         for (uint32_t i = 0; i < transactions.size(); i++) {
                result += "Transaction"+std::to_string(i)+": ";
882             result += transactions[i];
                result += "\n";
884         }
            }
886         result += "Hash: "+block.hash+"\n";
            block_to_send += result + "\n";
888     }
        return block_to_send;
890 }

892 Block BlockchainApplication::stringToBlock(const std::string& str) {
        std::istringstream iss(str);
894     std::string line;
        int id, type, mainParticipant, proof;
```

```
896     uint64_t timestamp;
        std::string previous_hash, hash;
898     std::vector<std::string> transactions;

900     while (std::getline(iss, line)) {
            std::istringstream line_stream(line);
902         std::string key, value;
            std::getline(line_stream, key, ':');
904         std::getline(line_stream, value);

906         if (key == "id") {
                id = std::stoi(value);
908         } else if (key == "type") {
                type = std::stoi(value);
910         } else if (key == "mainParticipant") {
                mainParticipant = std::stoi(value);
912         } else if (key == "timestamp") {
                timestamp = std::stoull(value);
914         } else if (key == "previous_hash") {
                // Verificar se o valor do previous_hash termina com v rgula
916             if (!value.empty() && value.back() == ',') {
                    // Remover a v rgula e o espa o final
918                 value.pop_back();
                    value.erase(std::remove_if(value.begin(), value.end(), ::isspace)
                        , value.end());
920             }
                previous_hash = value;
922         } else if (key == "proof/nonce") {
                proof = std::stoi(value);
924         } else if (key.find("Transaction") != std::string::npos) {
                // Verificar se a transa  o termina com v rgula
926             if (!value.empty() && value.back() == ',') {
                    // Remover a v rgula e o espa o final
928                 value.pop_back();
                    value.erase(std::remove_if(value.begin(), value.end(), ::isspace)
                        , value.end());
930             }
                transactions.push_back(value);
932         } else if (key == "Hash") {
                // Verificar se h  espa os vazios no come o e no final
934             while (!value.empty() && std::isspace(value.front())) {
                    value.erase(0, 1);
936             }
                while (!value.empty() && std::isspace(value.back())) {
938                 value.pop_back();
                }
940             // Verificar a presen a de uma v rgula
```

```
            if (!value.empty() && value.find(',') != std::string::npos) {
942             // Remover a v rgula e o espa o final
                value.pop_back();
944             value.erase(std::remove_if(value.begin(), value.end(), ::isspace)
                    , value.end());
            }
946         hash = value;
        }
948 }
    Block initialBlock(id, timestamp, transactions, proof, type, mainParticipant,
         previous_hash);
950 initialBlock.SetHash(hash);
    return initialBlock;
952 }

954 std::vector<Block> BlockchainApplication::stringToBlocks(const std::string& str)
    {
    std::istringstream iss(str);
956 std::vector<Block> blocks;
    std::string blockStr;
958 std::stringstream blockStream;

960 while (std::getline(iss, blockStr)) {
        if (!blockStr.empty()) {
962         blockStream << blockStr << std::endl;
        } else {
964         blocks.push_back(stringToBlock(blockStream.str()));
            blockStream.str(""); // Limpar o stringstream para o pr ximo bloco
966     }
    }
968 // Adicionar o  ltimo  bloco (se houver) ap s o loop
    if (!blockStream.str().empty()) {
970     blocks.push_back(stringToBlock(blockStream.str()));
    }
972

   return blocks;
974 }
```

ANEXO B – CÓDIGO FONTE DA BLOCKCHAIN MINERADOR - PROOF OF STAKE

```
#include "BlockchainApplicationPoS.h"
#define END_CODE "\033[0m"

#include "inet/applications/tcpapp/GenericAppMsg_m.h"
#include "inet/common/ModuleAccess.h"
#include "inet/common/TimeTag_m.h"
#include "inet/common/lifecycle/ModuleOperations.h"
#include "inet/common/packet/Packet.h"
#include "inet/common/ProtocolTag_m.h"
#include "inet/common/lifecycle/NodeStatus.h"
#include "inet/common/packet/Message.h"
#include "inet/common/packet/chunk/ByteCountChunk.h"
#include "inet/common/socket/SocketTag_m.h"
#include "inet/networklayer/common/L3AddressResolver.h"
#include "inet/transportlayer/contract/tcp/TcpCommand_m.h"
#include "inet/networklayer/contract/ipv4/Ipv4Address.h"
#include "inet/networklayer/contract/IInterfaceTable.h"

#include <omnetpp.h>

using namespace omnetpp;
using namespace std;
using namespace inet;
#define MSGKIND_CONNECT    0
#define MSGKIND_SEND       1

Define_Module(BlockchainApplicationPoS);

BlockchainApplicationPoS::BlockchainApplicationPoS(){}
BlockchainApplicationPoS::~BlockchainApplicationPoS(){}


void BlockchainApplicationPoS::initialize(int stage)
{
    cSimpleModule::initialize(stage);

    if (stage == INITSTAGE_LOCAL) {
        delay = par("replyDelay");
        maxMsgDelay = 0;

        // statistics
        msgsRcvd = msgsSent = bytesRcvd = bytesSent = 0;

```

```
        WATCH(msgsRcvd);
        WATCH(msgsSent);
        WATCH(bytesRcvd);
        WATCH(bytesSent);
    }
    else if (stage == INITSTAGE_APPLICATION_LAYER) {
        cout << "stage: " << stage << endl;
        int localPort = par("localPort");
        string localAddress;
        string peerAddrServer;
        cModule * host = getContainingNode(this);
        string edgeNode = host->getFullName();
        // Criando o Socket TCP e transformando ele em servidor ...
        socket.setOutputGate(gate("socketOut"));
        socket.bind(Ipv4Address(charArray), localPort); // local address/port
        socket.listen();

        // Criando o Socket TCP e transformando ele em servidor ...
        peerSocketServer.setOutputGate(gate("socketOut"));
        peerSocketServer.bind(Ipv4Address(peerAddr), localPort); // local address
            /port
        peerSocketServer.listen();


        cModule *node = findContainingNode(this);
        NodeStatus *nodeStatus = node ? check_and_cast_nullable<NodeStatus *>(
            node->getSubmodule("status")) : nullptr;
        bool isOperational = (!nodeStatus) || nodeStatus->getState() ==
            NodeStatus::UP;
        if (!isOperational)
            throw cRuntimeError("This module doesn't support starting in node
                DOWN state");

        startApplication();
    }
}

void BlockchainApplicationPoS::startApplication()
{
    cout << "Starting Blockchain Miner Application" << endl;


    //Create Genesis block
    BlockPoS block = BlockPoS(1,0,currentTransactions,0,0,"0000",0);
    block.hash = "0000";
    blockchain.chain.push_back(block);
```

```
87      mVotesAmount = 1;

89      stake = new int[TotalNodes];
        for (int i=0; i<TotalNodes; i++) {
91          srand(i);
            stake[i] = rand() % 10;
93      }

95      cout << "Stake: ";
        for (int i=0; i<TotalNodes; i++) {
97          cout << stake[i] << " | ";
        }
99      cout << endl;


101
        // Inicializando os par metros do ECDH:
103     ECDHKey.KeyPairGeneration();
        ECDHIv.KeyPairGeneration();
105     Server_ECC.ECC_KeyGeneration();

107     CryptoPP::StringSink sink(ServerKey_s);
        Server_ECC.GetPublicKey().Save(sink);
109
        ServerPrivKey = Server_ECC.GetPrivateKey();
111


113     std::string filename = std::to_string(mNodeId) + "-chain.txt";
        // cout << mNodeId << " - Writting in "<<filename<<". Interval = " <<
            interval<<endl;
115     outfile.open(filename, std::ios_base::app);
        outfile << "----------------------------------------\n";
117     outfile << block.toString();
        outfile.close();
119
        cMessage *msg = new cMessage("schedule the inicialization of the
            Communication");
121     msg->setKind(SETPOINT);
        scheduleAt(simTime() + 0.2, msg);
123 }

125 void BlockchainApplicationPoS::StartCommunication(){
            int peerPort = 1234;
127         cout << "Initialization of the peer comunication ..." << endl;
            cout << "Connecting to: " << peerAddrClient << ":" << peerPort << endl;
129         peerSocketClient.setOutputGate(gate("socketOut"));
            const char* peerAddr = peerAddrClient.c_str();
131         peerSocketClient.connect(*peerAddr ? L3AddressResolver().resolve(peerAddr
```

```
            ) : L3Address(), peerPort);
}
133

135 void BlockchainApplicationPoS::sendRequest()
{
137     long requestLength = par("requestLength");
    long replyLength = par("replyLength");
139     if (requestLength < 1)
        requestLength = 1;
141     if (replyLength < 1)
        replyLength = 1;
143
    Packet *packet = new Packet("data");
145     const auto& payload = makeShared<AuthPacket>();
    /**/
147     payload->setPacketid(CHAIN);
    payload->setChunkLength(B(requestLength));
149     payload->setExpectedReplyLength(B(replyLength));

151     payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
    packet->insertAtBack(payload);
153     /**/
    EV_INFO << "sending request with " << requestLength << " bytes, expected
        reply length " << replyLength << " bytes\n";
155     cout << endl;
    peerSocketClient.send(packet);
157 }

159 void BlockchainApplicationPoS::SetupECDH(int NodeId, const char* mkeyECDH, const
    char* mivECDH){

161     cout << "starting ECDH setup ... " << endl;
     cout << "Generating the ECDH keys to share a secret with Node " << NodeId <<
          endl;
163     /*
    There are two shared secrets one is used as key and the other is used as iv
        on the
165     aes cryptography, aiming a security channel.
    */
167
    ECDHKey.SetPubKeyB(mkeyECDH);
169     ECDHKey.SharedSecretGeneration();
    keyECDH = ECDHKey.GetSharedSecret();
171     DataBase.writeECDHkey(NodeId, keyECDH);
    Integer a;
173     a.Decode(keyECDH.BytePtr(), keyECDH.SizeInBytes());
```

```
    cout << "Shared Secret -- key (ECDH): " << std::hex << a << endl;
175
    ECDHIv.SetPubKeyB(mivECDH);
177 ECDHIv.SharedSecretGeneration();
    ivECDH = ECDHIv.GetSharedSecret();
179 DataBase.writeECDHiv(NodeId, ivECDH);
    a.Decode(ivECDH.BytePtr(), ivECDH.SizeInBytes());
181 cout << "Shared Secret -- iv  (ECDH): " << std::hex << a << endl;
}
183
void BlockchainApplicationPoS::SetupEcc(std::string privatekeyecc)
185 {
    const char* m_privkeyecc = privatekeyecc.c_str();
187 ECC.SetPrivateKey(m_privkeyecc);
}
189
void BlockchainApplicationPoS::SharingPublicKeyEDCH(string mkeyECDH, string
    mivECDH, int mconnId, cMessage *msg)
191 {
    const auto& DHkey = makeShared<AuthPacket>();
193 Packet *mPacket = new Packet(msg->getName(), TCP_C_SEND);
    mPacket->addTag<SocketReq>()->setSocketId(mconnId);
195 DHkey->setPacketid(SHARINGECDHKEYS);
    DHkey->setChunkLength(B(200));
197 DHkey->setExpectedReplyLength(B(200));
    // DHkey->setServerClose(false);
199 DHkey->setKey(mkeyECDH.c_str());
    DHkey->setIv(mivECDH.c_str());
201 DHkey->addTag<CreationTimeTag>()->setCreationTime(simTime());
    mPacket->insertAtBack(DHkey);
203 sendPacket(mPacket);
}
205
void BlockchainApplicationPoS::SendAuthenticadedKeys(int NodeId, int mconnId,
    cMessage *msg)
207 {
    cout << "Set up of the keys for the node " << NodeId << endl;
209
    // Key Generation
211 cout << "****** Inicializando ECC --- Node "<< NodeId <<" ******" << endl;

213 socketIds.insert({NodeId, mconnId});

215 EllipticCurveCryptography ECCKeys;
    ECCKeys.ECC_KeyGeneration();
217 DataBase.writeECC(NodeId, ECCKeys.GetPublicKey());
    std::string privatekeyecc = ECCKeys.GetPrivateKey();
```

```
219     cout << "Private Key: " << privatekeyecc <<endl;

221     //Encrypt Ecc private key
        std::string EccPrivateKeyCrypt = AES.EncryptionAES(privatekeyecc, DataBase.
            readECDHkey(NodeId), DataBase.readECDHiv(NodeId));
223
        const auto& ECCkey = makeShared<AuthPacket>();
225     Packet *mPacket = new Packet(msg->getName(), TCP_C_SEND);
        mPacket->addTag<SocketReq>()->setSocketId(mconnId);
227     ECCkey->setPacketid(AUTHENTICATIONANS);
        ECCkey->setChunkLength(B(200));
229     ECCkey->setExpectedReplyLength(B(200));
        // ECCkey->setServerClose(false);
231
        string HexEccPrivateKeyCrypt = localhex.string_to_hex(EccPrivateKeyCrypt);
233     ECCkey->setKey(HexEccPrivateKeyCrypt.c_str());
        /********TA ECC pub key*********/
235     // Convertendo a chave p blica do servidor para enviar
        string HexServerKey_s = localhex.string_to_hex(ServerKey_s);
237     ECCkey->setData(HexServerKey_s.c_str());
        ECCkey->addTag<CreationTimeTag>()->setCreationTime(simTime());
239     mPacket->insertAtBack(ECCkey);
        sendPacket(mPacket);
241
        cout << "Keys sended to " << NodeId << endl;
243     cout << "*****************************************************"<< endl;
}
245
void BlockchainApplicationPoS::seeIp()
247 {
        cModule * host = getContainingNode(this);
249     //L3Address addr = L3AddressResolver().addressOf(host);

251     string mNodeId = host->getFullName();
        cout << "Hey, I am the server: "<< mNodeId << endl ;
253 }

255 void BlockchainApplicationPoS::sendOrSchedule(cMessage *msg, simtime_t delay)
{
257     if (delay == 0)
            sendPacket(msg);
259     else
            scheduleAfter(delay, msg);
261 }


263
void BlockchainApplicationPoS::sendPacket(cMessage *msg)
```

```
{
    cout << "Server is sending a Packet ..." << endl;
    Packet *packet = dynamic_cast<Packet *>(msg);

    if (packet) {
        msgsSent++;
        bytesSent += packet->getByteLength();
        emit(packetSentSignal, packet);

        EV_INFO << "sending \"" << packet->getName() << "\" to TCP, " << packet->
            getByteLength() << " bytes\n";
    }
    else {
        EV_INFO << "sending \"" << msg->getName() << "\" to TCP\n";
    }

    auto& tags = check_and_cast<ITaggedObject *>(msg)->getTags();
    tags.addTagIfAbsent<DispatchProtocolReq>()->setProtocol(&Protocol::tcp);
    send(msg, "socketOut");
}

void BlockchainApplicationPoS::handleMessage(cMessage *msg)
{
    cout << "SimTime(): " << simTime() << endl;

    if (msg->isSelfMessage()) {
        switch (msg->getKind())
        {
            case AUTHSUCCESS:{
                long requestLength = par("requestLength");
                long replyLength = par("replyLength");
                if (requestLength < 1)
                    requestLength = 1;
                if (replyLength < 1)
                    replyLength = 1;
                Packet *packet = new Packet("data");
                const auto& payload = makeShared<AuthPacket>();
                payload->setPacketid(AUTHSUCCESS);
                payload->setNodeid(mNodeId);
                payload->setBlock_id(to_send_bid.c_str());
                payload->setData("no-pub-key");
                payload->setChunkLength(B(requestLength));
                payload->setExpectedReplyLength(B(replyLength));

                payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
                packet->insertAtBack(payload);
                peerSocketClient.send(packet);
```

```
311
                cout << mNodeId << "- is sending packet " << AUTHSUCCESS << endl;
313             break;
            }
315         case SETPOINT:{
                StartCommunication();
317             break;
            }
319     }
    }
321 else if (msg->getKind() == TCP_I_PEER_CLOSED) {
        // we'll close too, but only after there's surely no message
323     // pending to be sent back in this connection
        int connId = check_and_cast<Indication *>(msg)->getTag<SocketInd>()->
            getSocketId();
325     delete msg;
        auto request = new Request("close", TCP_C_CLOSE);
327     request->addTag<SocketReq>()->setSocketId(connId);
        sendOrSchedule(request, delay + maxMsgDelay);
329 }
    else if (msg->getKind() == TCP_I_DATA || msg->getKind() == TCP_I_URGENT_DATA)
         {
331     Packet *packet = check_and_cast<Packet*>(msg);
        int connId = packet->getTag<SocketInd>()->getSocketId();
333     ChunkQueue& queue = socketQueue[connId];
        auto chunk = packet->peekDataAt(B(0), packet->getTotalLength());
335     queue.push(chunk);
        emit(packetReceivedSignal, packet);
337     cout << endl;
        cout << "*****************************" << endl;
339     cout << mNodeId <<" - Recv a Packet: " << endl;

341     PacketPrinter printer; // turns packets into human readable strings
        printer.printPacket(std::cout, packet); // print to standard output
343     cout << "*****************************" << endl;
        const auto& mPayload = queue.pop<AuthPacket>(b(-1));
345     int Id = mPayload->getPacketid();
        cout<< mNodeId <<" - Packet ID = " << Id << endl;
347     cout << "*****************************" << endl;
        cout<<endl;
349
        switch (Id)
351     {
            case AUTHENTICATIONREQ:{
353             cout << "Authetication Request Recieved from node " << mPayload->
                    getNodeid() << endl;
                cout << "Preparing the channel for authentication." << endl;
```

```
355
                SetupECDH(mPayload->getNodeid(), mPayload->getKey(), mPayload->
                    getIv());
357             SharingPublicKeyEDCH(ECDHKey.GetPubKey(), ECDHIv.GetPubKey(),
                    connId, msg);
                break;
359         }

361         case CHANNELVERIFY:{

363             cout << "Receiving the security data verification from Node  " <<
                     mPayload->getNodeid() << endl;

365             string mData = mPayload->getData();
                string hexData = localhex.hex_to_string(mData);
367             string AgreedMessage = AES.DecryptionAES(hexData,DataBase.
                    readECDHkey(mPayload->getNodeid()), DataBase.readECDHiv(
                    mPayload->getNodeid()));
                cout << "Verification Message Received: " <<AgreedMessage << endl
                    ;
369
                if (AgreedMessage == "Security Channel")
371             {
                    cout << "The channel verification was completed." << endl;
373                 // Send the user its ECC keys
                    BlockchainApplicationPoS::SendAuthenticadedKeys(mPayload->
                        getNodeid(), connId, msg);
375                 // this->StartCommunication();

377                 CryptoPP::ECIES < ECC_ALGORITHM >::PublicKey pubKey = ECC.
                        GetPublicKey();

379                 std::string encodedPubKey;
                    StringSink saveSS(encodedPubKey);
381                 pubKey.Save(saveSS);
                    keysTransactionVector.clear();
383                 keysTransactionVector.push_back(encodedPubKey);
                    BlockPoS lastBlock = blockchain.chain[blockchain.chain.size()
                        -1];
385                 BlockPoS block = CreateBlockPoS(lastBlock,
                        keysTransactionVector, 1, mPayload->getNodeid(), mPayload
                        ->getNodeid());
                    if (pendingBlockchainId != "" || (mConsensusBidSize >
                        blockchain.chain.size())) {
387                     cout << mNodeId <<" - there is a pending blockchain. Add
                            block to pending list" << endl;
                        pendingBlocks.push_back(block);
```

```
389                     } else {
                            MineAndBroadcast(block);
391                     }
                    }else{
393                     cout << "Fail!" << endl;
                    }
395                 break;
                }
397             case CHAIN:{

399                 int strSenderId = mPayload->getNodeid();
                    cout << mNodeId << " - Node received chain from " << strSenderId
                        << endl;
401
                    std::string receivingBid = mPayload->getBlock_id();
403                 std::string blockStr = mPayload->getBlock();

405
                    cout << "Cheguei na chain " << endl;
407                 std::vector<BlockPoS> newChain = stringToBlocks(blockStr);

409                 receivingChainPackets = true;
                    if (mConsensusList.find(receivingBid) != mConsensusList.end()) {
411                 if (mConsensusList[receivingBid].GetReachedConsensusFlag() &&
                        !mConsensusList[receivingBid].GetVeredict()) {
413                     // If I'm receiving a chain that was already rejected I can
                            ignore the packets
                        mReceivingChainMap.erase(receivingBid);
415                     receivingChainPackets = (mReceivingChainMap.size() != 0);
                        break;
417                 }
                    }
419                 // if (newChain.size() == 0) break;
                    std::vector<BlockPoS>receivingChain = mReceivingChainMap[
                        receivingBid];
421
                    cout << mNodeId << " - Size of chain in current packet: " <<
                        newChain.size() << endl;
423                 cout << mNodeId << " - Size of receiving " << receivingBid << "
                        chain: " << receivingChain.size() << endl;
                    cout << mNodeId << " - Result received chain size: " << newChain.
                        size() << endl;
425                 createConsensusList(receivingBid, newChain);

427                 if (!mConsensusList[receivingBid].myVote) {
                        bool veredict = isBlockchainValid(newChain);
429                     if (veredict) {
```

```
                        cout << mNodeId << "- losing a vote ..." << endl;
                        mVotesAmount--;
                    }
                    sendVeredict(receivingBid, veredict);
                }
                reachedConsensus(receivingBid, newChain);
                mReceivingChainMap.erase(receivingBid);
                receivingChainPackets = (mReceivingChainMap.size() != 0);
                break;
            }

            case VEREDICT: {

                std::string bid = mPayload->getBlock_id();
                createConsensusList(bid);
                if (mConsensusList[bid].GetReachedConsensusFlag()) {
                cout << mNodeId << " - ignoring this veredict since "<<bid<<" was
                     already handled, veredict was "<<mConsensusList[bid].
                    GetVeredict() <<endl;
                break;
                }

                int id = mPayload->getNodeid();
                std::string strId = std::to_string(id);
                bool vote = mPayload->getVeredict_vote();
                int sender_id = mPayload->getVeredict_senderid();
                std::string strSenderId = std::to_string(sender_id);
                Veredict veredict(sender_id, vote);
                cout << mNodeId << " - Node " << mNodeId  << " received a
                    veredict " << veredict.GetSenderVote() << " from " +
                    strSenderId + " of blockchain "+bid<<endl;

                updateConsensusList(bid, veredict.GetSenderVote());
                if (mConsensusList[bid].chain.empty()) {
                cout << mNodeId << " - Did not receive the chain yet, cant try to
                     update" <<endl;
                } else {
                    cout << mNodeId << "- checking if reached the consensus" <<
                        endl;
                    reachedConsensus(bid, mConsensusList[bid].chain);
                }
                break;
            }

            case AUTHSUCCESS: {
                std::string bid = mPayload->getBlock_id();
                cout << mNodeId << " - bid=" << bid << endl;
```

```
                mConsensusList[bid].AddUpdateChain();
                updateChain(bid);
                break;
            }
        }

        bool doClose = false;
        if (doClose) {
            doClose = true;
            // break;
        }
        delete msg;

        if (doClose) {
            auto request = new Request("close", TCP_C_CLOSE);
            TcpCommand *cmd = new TcpCommand();
            blockchain.chain_validators.push_back(mNodeId);
            request->addTag<SocketReq>()->setSocketId(connId);
            request->setControlInfo(cmd);
            sendOrSchedule(request, delay + maxMsgDelay);
        }
    }
    else if (msg->getKind() == TCP_I_AVAILABLE)
        socket.processMessage(msg);
    else {
        // some indication -- ignore
        EV_WARN << "drop msg: " << msg->getName() << ", kind:" << msg->getKind()
            << "(" << cEnum::get("inet::TcpStatusInd")->getStringFor(msg->getKind
            ()) << ")\n";
        delete msg;
    }
}

BlockPoS BlockchainApplicationPoS::CreateBlockPoS(BlockPoS last_block,
                                std::vector<Transaction> currentTransactions,
                                int type,
                                int mainParticipant,
                                int validator) {
    int id = last_block.id+1;
    // int proof = last_block.proof;
    simtime_t t1;
    t1 = simTime();
    double time_ = t1.dbl();
    std::string previous_hash = last_block.hash;
    BlockPoS block = BlockPoS(
        id,
        time_,
```

```
        currentTransactions,
        // proof,
        type,
        mainParticipant,
        previous_hash,
        validator
    );
    block.SetHash();
    return block;
}


bool BlockchainApplicationPoS::addTransaction(std::vector<Transaction> pool,
    Transaction transaction) {
    pool.push_back(transaction);
    if(pool.size() >= LIMIT) {
        return true;
    } else {
        return false;
    }
}

bool BlockchainApplicationPoS::transactionExists(std::vector<Transaction> pool,
    Transaction transaction) {
    for(Transaction transac_t : pool) {
    if(transac_t.content == transaction.content) {
        return true;
    }
    else {
        return false;
    }
    }
}


void BlockchainApplicationPoS::finish()
{
    EV_INFO << getFullPath() << ": sent " << bytesSent << " bytes in " <<
        msgsSent << " packets\n";
    EV_INFO << getFullPath() << ": received " << bytesRcvd << " bytes in " <<
        msgsRcvd << " packets\n";

    cout << getFullPath() << ": sent " << bytesSent << " bytes in " << msgsSent
        << " packets\n" << endl;
    cout << getFullPath() << ": received " << bytesRcvd << " bytes in " <<
        msgsRcvd << " packets\n" << endl;
```

```
557 }


559

void BlockchainApplicationPoS::refreshDisplay() const
561 {
    char buf[100];
563 sprintf(buf, "rcvd: %ld pks %ld bytes\nsent: %ld pks %ld bytes", msgsRcvd,
        bytesRcvd, msgsSent, bytesSent);
    getDisplayString().setTagArg("t", 0, buf);
565 }


567

void BlockchainApplicationPoS::MineAndBroadcast(BlockPoS block)
569 {
    int id = mNodeId;
571 BlockPoS lastBlock = blockchain.chain[blockchain.chain.size()-1];
    cout <<mNodeId << " - MineAndBroadcast. MyLastBlock is " << lastBlock.id <<
        endl;
573 BlockPoS blockToMine = CreateBlockPoS(lastBlock, block.transactions, block.
        type, block.mainParticipant, block.validator);
    // N o entendi para que o eduardo colocou esse time?
575 int timeToBroadcast = blockToMine.Mine();
    // cout <<mNodeId << " - Mined in "<<timeToBroadcast << endl;
577 Blockchain newChain = blockchain;
    newChain.chain.push_back(blockToMine);
579 // cout <<"creating bid" << endl;
    pendingBlockchainId = "Enb"+std::to_string(id) + "_block" + std::to_string(
        mBid);
581 cout <<pendingBlockchainId << " has size = " << newChain.chain.size() << endl
        ;
    BroadcastBlockchain(newChain);
583 //Simulator::Schedule(timeToBroadcast, &BlockchainApplicationPoS::
        BroadcastBlockchain, this, newChain);
}
585

void BlockchainApplicationPoS::BroadcastBlockchain(Blockchain newChain) {
587

    std::string id = std::to_string(mNodeId);
589 cout << ": Broadcasting" << endl;
    std::string thisBid = "Enb"+ id + "_block" + std::to_string(mBid);
591 // just send last two block on the network
    std::vector<BlockPoS>chainToSend;
593

    if (newChain.chain.size() == 2) {
595     chainToSend = newChain.chain;
    } else {
597     chainToSend.push_back(newChain.chain[newChain.chain.size()-2]);
```

```
        chainToSend.push_back(newChain.chain[newChain.chain.size()-1]);
599     }

601     cout <<"Block0 = "<< chainToSend[0].toString() << endl;
        cout <<"Block1 = "<< chainToSend[1].toString() << endl;
603     createConsensusList(thisBid, chainToSend);

605     std::string m_data = chainToSend[0].toString() + "\n" + chainToSend[1].
            toString();

607     long requestLength = par("requestLength");
        long replyLength = par("replyLength");
609     if (requestLength < 1)
            requestLength = 1;
611     if (replyLength < 1)
            replyLength = 1;
613
        Packet *packet = new Packet("data");
615     const auto& payload = makeShared<AuthPacket>();
        payload->setPacketid(CHAIN);
617     payload->setNodeid(mNodeId);
        payload->setBlock(m_data.c_str());
619     payload->setBlock_id(thisBid.c_str());
        payload->setChunkLength(B(requestLength));
621     payload->setExpectedReplyLength(B(replyLength));
        payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
623     packet->insertAtBack(payload);
        peerSocketClient.send(packet);
625     cout << "********** A chain  foi enviada ... ********************" << endl;
    }
627
    void BlockchainApplicationPoS::createConsensusList(std::string bid) {
629     if (mConsensusList[bid].totalNodes == -1) {
            mConsensusList[bid] = ConsensusListPoS(TotalNodes - 1);
631     }
    }
633
    void BlockchainApplicationPoS::createConsensusList(std::string bid, std::vector<
        BlockPoS>chain) {
635     if (mConsensusList[bid].totalNodes == -1) {
        mConsensusList[bid] = ConsensusListPoS(TotalNodes - 1, chain);
637     } else {
        mConsensusList[bid] = ConsensusListPoS(mConsensusList[bid], chain);
639     }
    }
641 void BlockchainApplicationPoS::updateConsensusList(std::string bid, bool veredict
        ) {
```

```
    if (veredict) {
643     mConsensusList[bid].AddOk();
    cout << mNodeId << " - Updated Consensus List" <<endl;
645     } else {
    mConsensusList[bid].AddNok();
647     }
}
649

651 bool BlockchainApplicationPoS::isBlockchainValid(std::vector<BlockPoS> newChain)
    {
    cout<<mNodeId << " - isBlockchainValid" << endl;
653
    BlockPoS newestBlock = newChain[newChain.size()-1];
655     int lastIndex = blockchain.chain.size()-1;
    BlockPoS lastBlock = blockchain.chain[lastIndex];
657     if (newestBlock.id <= lastBlock.id) {
    cout<<mNodeId << " - small size. newestBlock.id=" << newestBlock.id << ",
        lastBlock.id="<<lastBlock.id<< endl;
659     return false;
    }
661     if (newestBlock.id <= mConsensusBidSize) {
    cout<<mNodeId << " - Already reaching consensus on a blockchain of this size.
         newChain="<<newestBlock.id<<" oldChain="<<blockchain.chain.size()<<",
        mConsensusBidSize="<<mConsensusBidSize<< endl;
663     return false;
    }
665     if (newestBlock.previous_hash != lastBlock.hash) {
    cout<<mNodeId << " - Disconnected newest block. newestBlock.previous_hash = "
        << newestBlock.previous_hash << ", lastBlock.hash: " << lastBlock.hash<<
        endl;
667     return false;
    }
669     BlockPoS previousBlockOfNewChain = newChain[0];
    if (newestBlock.IsInvalid() || previousBlockOfNewChain.IsInvalid()) {
671         cout<<mNodeId << "Invalid chain"<< endl;
        return false;
673     }

675     if (previousBlockOfNewChain.hash != lastBlock.hash ||
        previousBlockOfNewChain.previous_hash != lastBlock.previous_hash) {
677         cout<<mNodeId << " - Disconnected newest block. previousBlockOfNewChain.
            hash = "<< previousBlockOfNewChain.hash << ", lastBlock.hash: " <<
            lastBlock.hash<< endl;
        cout<<mNodeId << " - previousBlockOfNewChain.previous_hash = "<<
            previousBlockOfNewChain.previous_hash << ", lastBlock.previous_hash:
            " << lastBlock.previous_hash<< endl;
```

```
679         return false;
    }
681
    if (pendingBlockchainId != "") {
683     std::vector<BlockPoS>pendingChain = mConsensusList[pendingBlockchainId].chain
        ;
    if (pendingChain[pendingChain.size()-1].timestamp <= newestBlock.timestamp) {
685         cout<<mNodeId << " -   then current pendingChain last block "<< endl;
        return false;
687     }
    }
689     if (mVotesAmount <= 0) {
    cout << mNodeId << " - not enough votes"<< endl;
691     return false;
    }
693     return true;
}
695
void BlockchainApplicationPoS::sendVeredict(std::string bid, bool veredict) {
697     mConsensusList[bid].SetMyVote(veredict);
    cout << mNodeId << ": sendVeredict " << veredict << " to " << bid << "'s
        blockchain" << endl;
699
    // this->StartCommunication();
701
    long requestLength = par("requestLength");
703     long replyLength = par("replyLength");
    if (requestLength < 1)
705         requestLength = 1;
    if (replyLength < 1)
707         replyLength = 1;

709     Packet *packet = new Packet("data");
    const auto& payload = makeShared<AuthPacket>();
711     payload->setPacketid(VEREDICT);
    payload->setNodeid(mNodeId);
713     payload->setBlock_id(bid.c_str());
    payload->setVeredict_vote(veredict);
715     payload->setVeredict_senderid(mNodeId);
    payload->setChunkLength(B(requestLength));
717     payload->setExpectedReplyLength(B(replyLength));
    payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
719     packet->insertAtBack(payload);
    peerSocketClient.send(packet);
721


723     // BroadcastPacket(packet, VEREDICT);
```

```
}

void BlockchainApplicationPoS::updateChain(std::string bid) {
    cout << mNodeId << " - check "<< bid << " "<< mConsensusList[bid].updateChain
        <<"/"<<mConsensusList[bid].totalNodes << endl;
    if (mConsensusList[bid].chain.empty() && mConsensusList[bid].
        HasEveryNodeUpdated()) {
    cout << "Empty chain, cannot update it right now."<<endl;
    return;
    } else if (!mConsensusList[bid].chain.empty() && mConsensusList[bid].
        HasEveryNodeUpdated()) {

        chainsDoingConsensus--;
        std::string filename = std::to_string(mNodeId) + "-chain.txt";
        mConsensusList[bid].ok = mConsensusList[bid].totalNodes;

        if (blockchain.chain[blockchain.chain.size()-1].id >= mConsensusList[bid
            ].chain[1].id) {
            handleLateBlock(bid);
            return;
        }
        BlockPoS lastBlock = mConsensusList[bid].chain[mConsensusList[bid].chain.
            size()-1];
        cout << mNodeId << " is updating its chain to " << bid << " with
            lastBlockId "<<lastBlock.id<<endl;
        blockchain.chain.push_back(lastBlock);
        // TODO: Fazer com que a blockchain seja escrita
        outfile.open(filename, std::ios_base::app);
        outfile << lastBlock.toString();
        outfile.close();

    if (pendingBlockchainId == bid) {
        if (lastBlock.type == 1) {
            cout << "Enviar para o usu rio o authsucss" <<endl;

            int mconnId = socketIds[lastBlock.mainParticipant];

            const auto& AuthSucess = makeShared<AuthPacket>();
            Packet *mPacket = new Packet("", TCP_C_SEND);
            mPacket->addTag<SocketReq>()->setSocketId(mconnId);
            AuthSucess->setPacketid(AUTHSUCCESS);
            AuthSucess->setChunkLength(B(200));
            AuthSucess->setExpectedReplyLength(B(200));
            // std::string encodedPubKey = lastBlock.transactions[0];
            // AuthSucess->setData(encodedPubKey.c_str());
            AuthSucess->addTag<CreationTimeTag>()->setCreationTime(simTime());
            mPacket->insertAtBack(AuthSucess);
```

```
765             sendPacket(mPacket);

767         }
            pendingBlockchainId = "";
769
        }
771     if (pendingBlockchainId == "" && pendingBlocks.size() > 0) {
            BlockPoS pendingBlock = pendingBlocks[0];
773         pendingBlocks.pop_front();
            // Simulator::Schedule(interval, &BlockchainApplicationPoS::
                MineAndBroadcast, this, pendingBlock);
775         MineAndBroadcast(pendingBlock);
        }
777     }
}
779

781 void BlockchainApplicationPoS::handleLateBlock(std::string bid) {
    cout<< mNodeId<<" - small id "<<bid <<endl;
783 std::string filename = std::to_string(mNodeId) + "-chain.txt";

785 int indexToInsert = mConsensusList[bid].chain[1].id -1;
    BlockPoS previousBlock = blockchain.chain[indexToInsert-1];
787 BlockPoS nextBlock = blockchain.chain[indexToInsert];
    BlockPoS blockToInsert = mConsensusList[bid].chain[1];
789 if (previousBlock.hash == blockToInsert.previous_hash &&
    nextBlock.previous_hash == blockToInsert.hash) {
791 cout << mNodeId << " - Late message block but it fits!" <<endl;
    blockchain.chain.insert(blockchain.chain.begin()+indexToInsert, blockToInsert
        );
793 outfile.open(filename);
    outfile << "----------------------------------------\n";
795 for (int i=0; i<blockchain.chain.size();i++) {
        outfile << blockchain.chain[i].toString();
797 }
    outfile.close();
799 }
    if (nextBlock.id == blockToInsert.id) {
801 cout << mNodeId <<" - Every node updated a chain that I did not update"<<endl
        ;
    int carId;
803 if (nextBlock.timestamp < blockToInsert.timestamp) {
        carId = blockchain.chain[indexToInsert].mainParticipant;
805     cout << mNodeId <<" - their chain is better then mine. Gonna ask "<<carId
            <<" to try again"<<endl;
        blockchain.chain[indexToInsert] = blockToInsert;
807 } else {
```

```
        carId = blockToInsert.mainParticipant;
809         cout << mNodeId <<" - my chain is better then theirs"<<endl;
    }
811
}
813
void BlockchainApplicationPoS::reachedConsensus(std::string bid, std::vector<
    BlockPoS>newChain) {
815
    if (mConsensusList[bid].IsConsensusReached()) {
817         chainsDoingConsensus++;
        if (mConsensusList[bid].myVote && mVotesAmount == 0) {
819             // After each consensus, if this node have voted to the blockchain we
                 add one more vote to it
            mVotesAmount = 1;
821         }
        int id = mNodeId;
823         if (newChain.empty()) {
            cout << "new chain is empty" << endl;
825             if (pendingBlockchainId == bid) {
            pendingBlockchainId = "";
827             }
            return;
829         }

831         BlockPoS lastBlock = newChain[newChain.size()-1];
        cout << id <<" reached consensus on " << bid << "'s Blockchain: "<<
            mConsensusList[bid].GetVeredict() << endl;
833
        if (mConsensusList[bid].GetVeredict()) {
835             cout << "entrei aqui ..." << endl;
            for (std::map<std::string,ConsensusListPoS>::iterator it =
                mConsensusList.begin(); it != mConsensusList.end(); ++it) {
837             ConsensusListPoS list = it->second;
            if (!list.IsConsensusReached() && !list.chain.empty() && list.chain
                [1].id <= newChain[1].id) {
839                 // Anul the rest of the chains since we've found a winner chain
                list.nok = list.totalNodes;
841                 reachedConsensus(it->first, list.chain);
            }
843             }

845             to_send_bid = bid;

847             cMessage *sendAuthSuccess = new cMessage("Scheduling the AUTHSUCCESS
                MSG");
            sendAuthSuccess->setKind(AUTHSUCCESS);
```

```
849             scheduleAt(simTime() + 0.001, sendAuthSuccess);

851             if (mConsensusList[bid].HasEveryNodeUpdated()) {
                    updateChain(bid);
853             }

855             return;
            } else {
857             if (pendingBlockchainId == bid) {
                mFailedChain++;
859             cout <<mNodeId << " - Block not added. Should try again. lastBlock =
                    " << lastBlock.toString()<<endl;

861             if ((mConsensusBidSize > blockchain.chain[blockchain.chain.size()-1].
                    id)) {
                    cout <<mNodeId << " - since I already have a block for the "<<
                        mConsensusBidSize<<" position, I'll leave the mining for
                        later."<<endl;
863                 pendingBlocks.insert(pendingBlocks.begin(), lastBlock);
                    pendingBlockchainId = "";
865             } else if (mVotesAmount < 1) {
                    cout <<mNodeId << " - cant vote, so the network must be setting
                        something, lets wait a bit"<<endl;
867                 pendingBlocks.insert(pendingBlocks.begin(), lastBlock);
                    pendingBlockchainId = "";
869             } else if (mFailedChain == 3) {
                    cout <<mNodeId << " - since I already tried 3 times, I'll wait a
                        few seconds before mining again"<<endl;
871                 pendingBlockchainId = "";
                    mFailedChain = 0;
873                 // Simulator::Schedule(interval*2, &BlockchainApplicationPoS::
                        MineAndBroadcast, this, lastBlock);
                    this->MineAndBroadcast(lastBlock);
875             } else {
                    // Simulator::Schedule(interval, &BlockchainApplicationPoS::
                        MineAndBroadcast, this, lastBlock);
877                 MineAndBroadcast(lastBlock);
                }
879             } else if (pendingBlockchainId == "" && mVotesAmount > 0 &&
                    pendingBlocks.size() > 0 && chainsDoingConsensus <= 0) {
                    cout <<mNodeId << " - I have nothing going on but some blocks are
                         pending, lets try mining"<<endl;
881                 BlockPoS pendingBlock = pendingBlocks[0];
                    pendingBlocks.pop_front();
883                 // Simulator::Schedule(interval, &BlockchainApplicationPoS::
                        MineAndBroadcast, this, pendingBlock);
                    this->MineAndBroadcast(pendingBlock);
```

```
                }
            }
    }else{
        cout<<"fail - aqui " <<endl;
    }
    return;
}

std::string BlockchainApplicationPoS::chainToSendToString(const std::vector<
    BlockPoS>& chain) {
    std::string block_to_send;
    for (const auto& block : chain) {

        std::string result = "";
        int m_type = 0;

        result += "id: "+std::to_string(block.id)+
                "\ntype: "+std::to_string(block.type)+
                "\nmainParticipant: "+std::to_string(block.mainParticipant)+
                ",\ntimestamp: "+std::to_string(block.timestamp)+
                ",\nprevious_hash: "+block.previous_hash+"\n";
        m_type = std::stoi(std::to_string(block.type));
        std::vector<Transaction> transactions = block.transactions;
    return block_to_send;
}



BlockPoS BlockchainApplicationPoS::stringToBlock(const std::string& str) {
    std::istringstream iss(str);
    std::string line;
    int id, type, mainParticipant, proof;
    uint64_t timestamp;
    std::string previous_hash, hash;
    std::vector<Transaction> transactions;

    while (std::getline(iss, line)) {
        std::istringstream line_stream(line);
        std::string key, value;
        std::getline(line_stream, key, ':');
        std::getline(line_stream, value);

        if (key == "id") {
            id = std::stoi(value);
        } else if (key == "type") {
            type = std::stoi(value);
        } else if (key == "mainParticipant") {
```

```
931             mainParticipant = std::stoi(value);
        } else if (key == "timestamp") {
933             timestamp = std::stoull(value);
        } else if (key == "previous_hash") {
935             // Verificar se o valor do previous_hash termina com v rgula
                if (!value.empty() && value.back() == ',') {
937                 // Remover a v rgula e o espa o final
                    value.pop_back();
939                 value.erase(std::remove_if(value.begin(), value.end(), ::isspace)
                        , value.end());
                }
941             previous_hash = value;
        } else if (key == "proof/nonce") {
943             proof = std::stoi(value);
        } else if (key.find("Transaction") != std::string::npos) {
945             // Verificar se a transa  o termina com v rgula
                if (!value.empty() && value.back() == ',') {
947                 // Remover a v rgula e o espa o final
                    value.pop_back();
949                 value.erase(std::remove_if(value.begin(), value.end(), ::isspace)
                        , value.end());
                }
951             transactions.push_back(value);
        } else if (key == "Hash") {
953             // Verificar se h  espa os vazios no come o e no final
                while (!value.empty() && std::isspace(value.front())) {
955                 value.erase(0, 1);
                }
957             while (!value.empty() && std::isspace(value.back())) {
                    value.pop_back();
959             }
                // Verificar a presen a de uma v rgula
961             if (!value.empty() && value.find(',') != std::string::npos) {
                    // Remover a v rgula e o espa o final
963                 value.pop_back();
                    value.erase(std::remove_if(value.begin(), value.end(), ::isspace)
                        , value.end());
965             }
                hash = value;
967         }
    }
969     BlockPoS initialBlock(id, timestamp, transactions, type, mainParticipant,
        previous_hash, id);
    initialBlock.SetHash(hash);
971     return initialBlock;
}
973
```

```
std::vector<BlockPoS>BlockchainApplicationPoS::stringToBlocks(const std::string&
    str) {
    std::istringstream iss(str);
    std::vector<BlockPoS> blocks;
    std::string blockStr;
    std::stringstream blockStream;

    while (std::getline(iss, blockStr)) {
        if (!blockStr.empty()) {
            blockStream << blockStr << std::endl;
        } else {
            blocks.push_back(stringToBlock(blockStream.str()));
            blockStream.str(""); // Limpar o stringstream para o pr ximo bloco
        }
    }
    // Adicionar o  ltimo  bloco (se houver) ap s o loop
    if (!blockStream.str().empty()) {
        blocks.push_back(stringToBlock(blockStream.str()));
    }

   return blocks;
}
```

ANEXO C – CÓDIGO FONTE DA AUTORIDADE DE CONFIANÇA

```
#include "TrustAuthorityApplication.h"
#define END_CODE "\033[0m"

#include "inet/applications/tcpapp/GenericAppMsg_m.h"
#include "inet/common/ModuleAccess.h"
#include "inet/common/TimeTag_m.h"
#include "inet/common/lifecycle/ModuleOperations.h"
#include "inet/common/packet/Packet.h"
#include "inet/common/ProtocolTag_m.h"
#include "inet/common/lifecycle/NodeStatus.h"
#include "inet/common/packet/Message.h"
#include "inet/common/packet/chunk/ByteCountChunk.h"
#include "inet/common/socket/SocketTag_m.h"
#include "inet/networklayer/common/L3AddressResolver.h"
#include "inet/transportlayer/contract/tcp/TcpCommand_m.h"
#include "inet/networklayer/contract/ipv4/Ipv4Address.h"
#include "inet/networklayer/contract/IInterfaceTable.h"

#include <omnetpp.h>

using namespace omnetpp;
using namespace std;
using namespace inet;
#define MSGKIND_CONNECT    0
#define MSGKIND_SEND       1

Define_Module(TrustAuthorityApplication);

TrustAuthorityApplication::TrustAuthorityApplication(){}
TrustAuthorityApplication::~TrustAuthorityApplication(){}


void TrustAuthorityApplication::initialize(int stage)
{
    cSimpleModule::initialize(stage);

    if (stage == INITSTAGE_LOCAL) {
        delay = par("replyDelay");
        maxMsgDelay = 0;

        // statistics
        msgsRcvd = msgsSent = bytesRcvd = bytesSent = 0;

```

```
        WATCH(msgsRcvd);
        WATCH(msgsSent);
        WATCH(bytesRcvd);
        WATCH(bytesSent);
    }
    else if (stage == INITSTAGE_APPLICATION_LAYER) {
        cout << "stage: " << stage << endl;
        const char *localAddress = par("localAddress");
        cout << "addr: " << localAddress << endl;
        int localPort = par("localPort");

        cout << "Server info: Addr = "<< localAddress << ":" << localPort << endl
            ;

        // Criando o Socket TCP e transformando ele em servidor ...
        socket.setOutputGate(gate("socketOut"));
        // socket.bind(*localAddress ? L3AddressResolver().resolve(localAddress)
            : L3Address(), localPort);
        socket.bind(localAddress[0] ? L3AddressResolver().resolve(localAddress) :
             L3Address(), localPort);
        socket.listen();

        cModule *node = findContainingNode(this);
        NodeStatus *nodeStatus = node ? check_and_cast_nullable<NodeStatus *>(
            node->getSubmodule("status")) : nullptr;
        bool isOperational = (!nodeStatus) || nodeStatus->getState() ==
            NodeStatus::UP;
        if (!isOperational)
            throw cRuntimeError("This module doesn't support starting in node
                DOWN state");

        startApplication();
    }
}

void TrustAuthorityApplication::startApplication()
{
    cout << "Configuring TA Application" << endl;
    mInterval = 0.05; //Time::FromDouble(0.2, Time::S);

    // Inicializando os par metros do ECDH:
    ECDHKey.KeyPairGeneration();
    ECDHIv.KeyPairGeneration();
    Server_ECC.ECC_KeyGeneration();

    CryptoPP::StringSink sink(ServerKey_s);
    Server_ECC.GetPublicKey().Save(sink);
```

```
86
    ServerPrivKey = Server_ECC.GetPrivateKey();
88
    cout<<"| TA Privite Expoent: " << Server_ECC.GetPrivate().GetPrivateExponent
        ()<<endl;
90
    cout<<"| TA Public Element:"<<endl;
92  cout<<"| X: " << Server_ECC.GetPublicKey().GetPublicElement().x<<endl;
    cout<<"| Y: " << Server_ECC.GetPublicKey().GetPublicElement().y<<endl;
94  }

96  void TrustAuthorityApplication::SetupECDH(int NodeId, const char* mkeyECDH, const
     char* mivECDH){

98      cout << "starting ECDH setup ... " << endl;
     cout << "Generating the ECDH keys to share a secret with Node " << NodeId <<
          endl;
100     /*
        There are two shared secrets one is used as key and the other is used as iv
            on the
102     aes cryptography, aiming a security channel.
        */
104
        ECDHKey.SetPubKeyB(mkeyECDH);
106     ECDHKey.SharedSecretGeneration();
        keyECDH = ECDHKey.GetSharedSecret();
108     DataBase.writeECDHkey(NodeId, keyECDH);
        Integer a;
110     a.Decode(keyECDH.BytePtr(), keyECDH.SizeInBytes());
        cout << "Shared Secret -- key (ECDH): " << std::hex << a << endl;
112
        ECDHIv.SetPubKeyB(mivECDH);
114     ECDHIv.SharedSecretGeneration();
        ivECDH = ECDHIv.GetSharedSecret();
116     DataBase.writeECDHiv(NodeId, ivECDH);
        a.Decode(ivECDH.BytePtr(), ivECDH.SizeInBytes());
118     cout << "Shared Secret -- iv  (ECDH): " << std::hex << a << endl;
    }
120
    void TrustAuthorityApplication::SetupEcc(std::string privatekeyecc)
122 {
        const char* m_privkeyecc = privatekeyecc.c_str();
124     ECC.SetPrivateKey(m_privkeyecc);
    }
126
    void TrustAuthorityApplication::SharingPublicKeyEDCH(string mkeyECDH, string
        mivECDH, int mconnId, cMessage *msg)
```

```
128 {
        const auto& DHkey = makeShared<PacketData>();
130     Packet *mPacket = new Packet(msg->getName(), TCP_C_SEND);
        mPacket->addTag<SocketReq>()->setSocketId(mconnId);
132     DHkey->setPacketid(SHARINGECDHKEYS);
        DHkey->setChunkLength(B(200));
134     DHkey->setExpectedReplyLength(B(200));
        DHkey->setServerClose(false);
136     DHkey->setKey(mkeyECDH.c_str());
        DHkey->setIv(mivECDH.c_str());
138     DHkey->addTag<CreationTimeTag>()->setCreationTime(simTime());
        mPacket->insertAtBack(DHkey);
140     sendPacket(mPacket);
    }
142
    void TrustAuthorityApplication::SendAuthenticadedKeys(int NodeId, int mconnId,
        cMessage *msg)
144 {
        cout << "Set up of the keys for the node " << NodeId << endl;
146
        // Key Generation
148     cout << "****** Inicializando ECC --- Node "<< NodeId <<" ******" << endl;

150     EllipticCurveCryptography ECCKeys;
        ECCKeys.ECC_KeyGeneration();
152     DataBase.writeECC(NodeId, ECCKeys.GetPublicKey());
        std::string privatekeyecc = ECCKeys.GetPrivateKey();
154     cout << "Private Key: " << privatekeyecc <<endl;

156     //Encrypt Ecc private key
        std::string EccPrivateKeyCrypt = AES.EncryptionAES(privatekeyecc, DataBase.
            readECDHkey(NodeId), DataBase.readECDHiv(NodeId));
158
        const auto& ECCkey = makeShared<PacketData>();
160     Packet *mPacket = new Packet(msg->getName(), TCP_C_SEND);
        mPacket->addTag<SocketReq>()->setSocketId(mconnId);
162     ECCkey->setPacketid(AUTHENTICATIONANS);
        ECCkey->setChunkLength(B(200));
164     ECCkey->setExpectedReplyLength(B(200));
        ECCkey->setServerClose(false);
166
        string HexEccPrivateKeyCrypt = localhex.string_to_hex(EccPrivateKeyCrypt);
168     ECCkey->setKey(HexEccPrivateKeyCrypt.c_str());
        /********TA ECC pub key*********/
170     // Convertendo a chave p blica do servidor para enviar
        string HexServerKey_s = localhex.string_to_hex(ServerKey_s);
172     ECCkey->setData(HexServerKey_s.c_str());
```

```
    ECCkey->addTag<CreationTimeTag>()->setCreationTime(simTime());
174     mPacket->insertAtBack(ECCkey);
    sendPacket(mPacket);
176
    cout << "Keys sended to " << NodeId << endl;
178     cout << "*****************************************************"<< endl;
}
180
void TrustAuthorityApplication::seeIp()
182 {
    cModule * host = getContainingNode(this);
184     //L3Address addr = L3AddressResolver().addressOf(host);

186     cout << "Hey, I am the server: "<< host << endl ;
}
188
void TrustAuthorityApplication::sendOrSchedule(cMessage *msg, simtime_t delay)
190 {
    if (delay == 0)
192         sendPacket(msg);
    else
194         scheduleAfter(delay, msg);
}
196


198 void TrustAuthorityApplication::sendPacket(cMessage *msg)
{
200     cout << "Server is sending a Packet ..." << endl;
    Packet *packet = dynamic_cast<Packet *>(msg);
202
    if (packet) {
204         msgsSent++;
        bytesSent += packet->getByteLength();
206         emit(packetSentSignal, packet);

208         EV_INFO << "sending \"" << packet->getName() << "\" to TCP, " << packet->
            getByteLength() << " bytes\n";
    }
210     else {
        EV_INFO << "sending \"" << msg->getName() << "\" to TCP\n";
212     }

214     auto& tags = check_and_cast<ITaggedObject *>(msg)->getTags();
    tags.addTagIfAbsent<DispatchProtocolReq>()->setProtocol(&Protocol::tcp);
216     send(msg, "socketOut");
}
218
```

```
void TrustAuthorityApplication::handleMessage(cMessage *msg)
{
    cout << "Server is handling the message ..." << endl;

    if (msg->isSelfMessage()) {
        sendPacket(msg);
    }
    else if (msg->getKind() == TCP_I_PEER_CLOSED) {
        // we'll close too, but only after there's surely no message
        // pending to be sent back in this connection
        int connId = check_and_cast<Indication *>(msg)->getTag<SocketInd>()->
            getSocketId();
        delete msg;
        auto request = new Request("close", TCP_C_CLOSE);
        request->addTag<SocketReq>()->setSocketId(connId);
        sendOrSchedule(request, delay + maxMsgDelay);
    }
    else if (msg->getKind() == TCP_I_DATA || msg->getKind() == TCP_I_URGENT_DATA)
         {
        Packet *packet = check_and_cast<Packet *>(msg);
        int connId = packet->getTag<SocketInd>()->getSocketId();
        ChunkQueue& queue = socketQueue[connId];
        auto chunk = packet->peekDataAt(B(0), packet->getTotalLength());
        queue.push(chunk);
        emit(packetReceivedSignal, packet);
        cout << endl;
        cout << "*****************************" << endl;
        cout << "Server Recv a Packet: " << endl;
        cout << "msg: " << msg << endl;

        const auto& mPayload = queue.pop<PacketData>(b(-1));
        int Id = mPayload->getPacketid();
        cout<<"Packet ID = " << Id << endl;
        cout << "*****************************" << endl;
        cout<<endl;

        switch (Id)
        {
            case AUTHENTICATIONREQ:{
                cout << "Authetication Request Recieved from node " << mPayload->
                    getNodeid() << endl;
                cout << "Preparing the channel for authentication." << endl;

                SetupECDH(mPayload->getNodeid(), mPayload->getKey(), mPayload->
                    getIv());
                SharingPublicKeyEDCH(ECDHKey.GetPubKey(), ECDHIv.GetPubKey(),
                    connId, msg);
```

```
                break;
262             }

264             case CHANNELVERIFY:{

266                 cout << "Receiving the security data verification from Node  " <<
                        mPayload->getNodeid() << endl;

268                 string mData = mPayload->getData();
                    string hexData = localhex.hex_to_string(mData);
270                 string AgreedMessage = AES.DecryptionAES(hexData,DataBase.
                        readECDHkey(mPayload->getNodeid()), DataBase.readECDHiv(
                        mPayload->getNodeid()));
                    cout << "Verification Message Received: " <<AgreedMessage << endl
                        ;
272
                    if (AgreedMessage == "Security Channel")
274                 {
                        cout << "The channel verification was completed." << endl;
276                     TrustAuthorityApplication::SendAuthenticadedKeys(mPayload->
                            getNodeid(), connId, msg);

278                     // After the car be authenticated take its interface and add
                            to auth authenticated
                        // TODO: pegar o endere o IP do usu rio para criar um
                            socket futuramente.
280                     // mAuthFace[mIndexFace] = mInterfaces.GetAddress(data.
                            GetNodeId());
                        // mIndexFace++;
282                 }else{
                        cout << "Fail!" << endl;
284                 }
                    break;
286             }
                }
288
            bool doClose = false;
290         if (doClose) {
                doClose = true;
292             // break;
            }
294         delete msg;

296         if (doClose) {
                auto request = new Request("close", TCP_C_CLOSE);
298             TcpCommand *cmd = new TcpCommand();
                request->addTag<SocketReq>()->setSocketId(connId);
```

```
            request->setControlInfo(cmd);
            sendOrSchedule(request, delay + maxMsgDelay);
        }
    }
    else if (msg->getKind() == TCP_I_AVAILABLE)
        socket.processMessage(msg);
    else {
        // some indication -- ignore
        EV_WARN << "drop msg: " << msg->getName() << ", kind:" << msg->getKind()
            << "(" << cEnum::get("inet::TcpStatusInd")->getStringFor(msg->getKind
            ()) << ")\n";
        delete msg;
    }
}

void TrustAuthorityApplication::finish()
{
    EV_INFO << getFullPath() << ": sent " << bytesSent << " bytes in " <<
        msgsSent << " packets\n";
    EV_INFO << getFullPath() << ": received " << bytesRcvd << " bytes in " <<
        msgsRcvd << " packets\n";

    cout << getFullPath() << ": sent " << bytesSent << " bytes in " << msgsSent
        << " packets\n" << endl;
    cout << getFullPath() << ": received " << bytesRcvd << " bytes in " <<
        msgsRcvd << " packets\n" << endl;
}


void TrustAuthorityApplication::refreshDisplay() const
{
    char buf[100];
    sprintf(buf, "rcvd: %ld pks %ld bytes\nsent: %ld pks %ld bytes", msgsRcvd,
        bytesRcvd, msgsSent, bytesSent);
    getDisplayString().setTagArg("t", 0, buf);
}
```

ANEXO D – MDMA - IOV

```
from socket import socket, SOCK_STREAM, AF_INET
import time
from runSimu import env
import random
import re
import json
from optuna import Trial, Study,create_study, load_study
from optuna.trial import FrozenTrial

def RunEnv(command):

    State_to_send = env(action=command['Solu  o'], level=command['Carros'],
        num_infra=command['Infras'])
    data = '{"auth_time":' + str(State_to_send[0]) + ', "link_consumption":' +
        str(State_to_send[1]) + ', "cqi":' + str(State_to_send[2])
    data = data + ', "response_time":' + str(State_to_send[3]) + ', "number_car":
        '+ str(State_to_send[4]) +', "number_infra":'+ str(State_to_send[5]) + ',
         "service_level":'+ str(State_to_send[6]) +'}'

    print(data)
    return State_to_send[0],State_to_send[1], State_to_send[3],State_to_send[4]


def objective(trial:Trial):

    solution = trial.suggest_categorical('Solu  o', ['TA'])
    num_cars = trial.suggest_categorical('Carros', [100,200,300])
    infras = trial.suggest_categorical('Infras', [2,4,6])

    command = {
    "Solu  o": solution,
    "Carros": num_cars,
    "Infras": infras
    }

    print(f'Command = {command}')

    return RunEnv(command= command)




study = create_study("sqlite:///iov_optim_TA_2.db",
                     directions=['minimize', 'maximize', 'minimize'],
```

```
                    study_name='IOV Optimization2',
                    load_if_exists=True)

study.optimize(objective, n_trials=60)

'''
Otimiza  o com n mero de carros variando nos  valores de 100 a 1000
Maximizando: o n mero de carros e  Throughput, e minimizando: Tempo de
    Autentica  o
'''
'''
study = create_study("sqlite:///iov_optim_TA.db",
                    directions=['minimize', 'maximize', 'minimize','maximize'],
                    study_name='IOV Optimization',
                    load_if_exists=True)

study.optimize(objective, n_trials=60)
'''
''''
Solu  o:"TA","PoS","PoW"
Carros: 100 a 1000
Infras:2,4,6
'''
'''
command = {
    "Solu  o": "TA",
    "Carros": 100,
    "Infras": 2
}
'''


study:Study = load_study(study_name="IOV Optimization", storage="sqlite:///
    iov_optim_variabe_startegies.db")
trials = study.get_trials()

for trial in trials:

    print(f"Params:{trial.params}, Value:{trial.values}")

print('---------------Best Values-------------------')
    for trial in study.best_trials:
        print(f"Params:{trial.params}, Value:{trial.values}")
```