UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Cristian Alves dos Santos

Uma abordagem para a integridade dos dados na preservação de evidências forenses usando identidades auto-soberanas

Florianópolis
2024

Cristian Alves dos Santos

Uma abordagem para a integridade dos dados na preservação de evidências forenses usando identidades auto-soberanas

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Ciências da Computação.
Orientador: Profª. Carla Merkle Westphall, Dra.

Florianópolis
2024

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

Santos, Cristian Alves dos
Uma abordagem para a integridade dos dados na preservação de evidências forenses usando identidades auto soberanas / Cristian Alves dos Santos ; orientadora, Carla Merkle Westphall, 2024.
128 p.

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

Inclui referências.

1. Ciência da Computação. 2. Identidade auto-soberana. 3. Preservação de evidências forenses. 4. Internet das coisas. 5. Blockchain. I. Westphall, Carla Merkle. II. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Ciência da Computação. III. Título.

Cristian Alves dos Santos

Uma abordagem para a integridade dos dados na preservação de evidências forenses usando identidades auto-soberanas

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

Prof. Caciano dos Santos Machado, Dr.
Universidade Federal do Rio Grande do Sul

Prof. Alex Sandro Roschildt Pinto, Dr.
Universidade Federal de Santa Catarina

Prof. Jean Everson Martina, Dr.
Universidade Federal de Santa Catarina

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

Documento assinado digitalmente
Marcio Bastos Castro
Data: 03/05/2024 14:51:41-0300
CPF: ***.707.720-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de
Pós-Graduação

Documento assinado digitalmente
CARLA MERKLE WESTPHALL
Data: 03/05/2024 14:40:00-0300
CPF: ***.150.339-**
Verifique as assinaturas em https://v.ufsc.br

Profª. Carla Merkle Westphall, Dra.
Orientador

Florianópolis, 2024.

Dedico este trabalho à minha amada mãe, que me amou incondicionalmente e que tanta falta faz em todos os momentos da minha vida (*in memoriam*).

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão à Professora Carla Merkle Westphall, minha orientadora, pelo constante apoio, orientação e paciência ao longo deste processo desafiador. Também desejo estender meus agradecimentos aos membros e colegas do LRG, especialmente a Leandro Loffi, cujas ideias foram importantes para o progresso deste trabalho, e aos Professores Caciano Machado, Jean Martina e Alex Sandro, por aceitarem participar da minha banca e por suas valiosas contribuições para o aprimoramento deste estudo.

Agradeço à minha esposa, Aline, por seu encorajamento, apoio e compreensão. Sem você eu não teria iniciado essa jornada tão desafiadora. Aos meus filhos, Bryan e Isadora, que são a minha maior alegria. Vocês são o meu futuro e eu me dedico a vocês para que possam construir um mundo melhor.

Expresso minha eterna gratidão à minha amada mãe, Maria Soeli, que sempre me protegeu e me amou incondicionalmente. Desde sua partida, não houve um momento em que seu exemplo de bondade, força e resiliência não tenha sido a luz a guiar meus passos. Mesmo diante de inúmeras adversidades, manteve-se sempre com um sorriso no rosto, pronta para seguir em frente. Sua calma e sabedoria eram fontes inesgotáveis de inspiração e conforto, e nunca me esqueci o quão privilegiado sou por ter tido uma super mãe como você. Mãe, eternamente te amarei!

“Quanto mais você sabe, mais você percebe que não sabe.”
“Se eu vi mais longe, foi por estar de pé sobre ombros de gigantes.”
(SÓCRATES, através de Platão, séc. IV a.C.; NEWTON, 1675)

RESUMO

Com a expansão das redes de Internet das Coisas (IoT), surgem desafios significativos relacionados ao gerenciamento seguro dos dados gerados por esses dispositivos. A integridade e autenticidade desses dados são essenciais para setores sensíveis, como a preservação de evidências forenses. Nesse contexto, este estudo propôs uma arquitetura, que abrange as camadas *Edge*, *Fog* e *Cloud computing*, baseada em protocolos e tecnologias de Identidade auto-soberana (SSI), utilizando as bibliotecas Hyperledger Indy e Aries, para possibilitar o uso de Identificadores Descentralizados (DIDs) para identificação das partes envolvidas e Credenciais Verificáveis (VCs) para assegurar a integridade dos dados emitidos em redes IoT. A implementação prática dessa solução é realizada por meio de uma aplicação na camada *Fog* que atua como ponte entre a camada *Edge*, na qual o *broker* MQTT está conectado, e a camada *Cloud Computing*, na qual os agentes atuam como interface para a conexão com outros domínios e a *blockchain*. Para avaliar a eficiência da abordagem proposta, foram conduzidos experimentos abrangentes e análises de desempenho, incluindo simulações detalhadas. Os experimentos revelaram que as VCs podem escalar para grandes conjuntos de dados, mas a utilização de DIDs ancorados em *blockchain* pode representar um desafio. A integração dessas tecnologias possibilita a certificação dos dados coletados por dispositivos IoT na origem, estabelecendo assim uma estrutura robusta para a cadeia de custódia de dados. Consequentemente, essa integração contribui substancialmente para preservar a integridade, confiabilidade, autenticidade e rastreabilidade desses dados por meio da *blockchain* em ambientes críticos.

Palavras-chave: *Internet das Coisas*. *Identidade auto-soberana*. *Identificadores Descentralizados*. *Credenciais Verificáveis*. *Blockchain*. *Preservação de Evidências*.

ABSTRACT

The proliferation of Internet of Things (IoT) networks brings forth significant challenges regarding the secure management of data generated by these devices. Data integrity and authenticity are paramount in sensitive sectors such as forensic evidence preservation. This study proposes an architecture spanning Edge, Fog, and Cloud computing layers, leveraging Self-Sovereign Identity (SSI) protocols and technologies. We utilize the Hyperledger Indy and Aries libraries to enable Decentralized Identifiers (DIDs) for stakeholder identification and Verifiable Credentials (VCs) to ensure ownership of data issued in IoT networks. A practical implementation of this solution is realized through a Fog layer application acting as a gateway between the Edge layer, where the MQTT broker resides, and the Cloud Computing layer, where agents serve as an interface for connecting to other domains and the blockchain. To assess the proposed approach's efficiency, we conducted extensive experiments and performance analyses, including detailed simulations. The experiments revealed that VCs can scale to large datasets, while the use of blockchain-anchored DIDs may pose a challenge. The integration of these technologies enables certification of data collected by IoT devices at the source, establishing a robust framework for data custody chains. Consequently, this integration contributes substantially to preserving data integrity, reliability, authenticity, and traceability through blockchain in critical environments.

Keywords: *Internet of Things*. *Self-sovereign identity*. *Decentralized Identifiers*. *Verifiable Credentials*. *Blockchain*. *Preservation of Evidence*.

LISTA DE FIGURAS

Figura 1 – Exemplo simples de um identificador descentralizado (DID). 32
Figura 2 – Exemplo simples de um documento DID. 32
Figura 3 – Visão geral da arquitetura DID. 33
Figura 4 – Modelo de credencial de papel. 35
Figura 5 – Exemplo de parte de credencial. 36
Figura 6 – Visão geral da arquitetura. 37
Figura 7 – Esquema de apresentação de VCs por meio de ZKP. 38
Figura 8 – Comparação entre tecnologias blockchain. 41
Figura 9 – Protocolos Aries no ACA-Py da perspectiva do controlador. 44
Figura 10 – Contexto para preservação de evidência forense. 61
Figura 11 – Abordagem proposta em alto nível. 62
Figura 12 – Visão geral da arquitetura proposta. 64
Figura 13 – Credencial emitida pela aplicação. 66
Figura 14 – Integração das entidades na arquitetura proposta. 70
Figura 15 – Diagrama de sequência. 72
Figura 16 – Tipos de transações na blockhain Von Network. 73
Figura 17 – Containers Docker. 76
Figura 18 – Servidor web da rede Von Network. 77
Figura 19 – Diagrama de fluxo em alto nível da aplicação. 82
Figura 20 – Fluxo de automação de sensores de temperatura no Node-RED. . . 100
Figura 21 – Tempo médio de resposta na primeira etapa da fase 1 do experimento.102
Figura 22 – Uso de memória na primeira etapa da fase 1 do experimento. 104
Figura 23 – Taxa de utilização de CPU na primeira etapa da fase 1 do experimento.104
Figura 24 – Tempo médio de resposta na segunda etapa da fase 1 do experimento.105
Figura 25 – Uso de memória na segunda etapa da fase 1 do experimento. . . . 106
Figura 26 – Taxa de utilização de CPU na segunda etapa da fase 1 do experimento.107
Figura 27 – Tempo médio de resposta em transações com a blockchain. 108
Figura 28 – Uso de memória na fase 2 do experimento. 109
Figura 29 – Taxa de utilização de CPU na fase 2 do experimento. 109
Figura 30 – Tempo médio de reposta na fase 3 do experimento. 110
Figura 31 – Uso de memória na fase 3 do experimento. 111
Figura 32 – Taxa de utilização de CPU na fase 3 do experimento. 112

LISTA DE QUADROS

Quadro 1 – Bases de dados. 48
Quadro 2 – Ferramentas utilizadas na proposta. 69

LISTA DE TABELAS

Tabela 1 – Comparação entre tecnologias propostas e recursos. 53
Tabela 2 – Comparação entre trabalhos relacionados e este estudo. 58
Tabela 3 – Resultados na primeira etapa da fase 1 do experimento. 103
Tabela 4 – Resultados na segunda etapa da fase 1 do experimento. 106
Tabela 5 – Resultados da fase 2 do experimento. 108
Tabela 6 – Resultados da fase 3 do experimento. 111

LISTA DE ABREVIATURAS E SIGLAS

AMQP	Advanced Message Queuing Protocol
CoAP	Constrained Application Protocol
DIDComm	DID Communication
DIDDoc	Decentralized Identifier Document
DIDs	Decentralized Identifiers
DLT	Distributed Ledger Technology
DNS	Domain Name System
EPC	Electronic Product Code
IIoT	Industrial Internet of Things
IMEI	International Mobile Equipment Identity
IoMT	Internet of Medical Things
IoT	Internet of Things
IoV	Internet of Vehicles
IP	Internet Protocol
JSON-LD	JSON for Linking Data
M2M	Machine to Machine
MAC	Media Access Control
MQTT	Message Queuing Telemetry Transport
OID	Object Identifier
PGP	Pretty Good Privacy
PUF	Physical Unclonable Function
REST	Representational State Transfer
RFID	Radio-Frequency Identification
SDK	Software Development Kit
SSI	Self-Sovereign Identity
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPM	Trusted Platform Module
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VCs	Verifiable Credentials
VDR	Verifiable Data Registry
XML	eXtensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
ZKP	Zero-Knowledge Proof

SUMÁRIO

1 INTRODUÇÃO 17
1.1 PROBLEMA DE PESQUISA E MOTIVAÇÃO 18
1.2 OBJETIVOS 19
1.2.1 Objetivo geral 19
1.2.2 Objetivos específicos 19
1.3 JUSTIFICATIVA 20
1.4 MÉTODO 22
1.5 ORGANIZAÇÃO DO TRABALHO 23
2 FUNDAMENTAÇÃO TEÓRICA 24
2.1 PRESERVAÇÃO DE EVIDÊNCIAS FORENSES 24
2.1.1 Cadeia de custódia de dados digitais 25
2.2 INTERNET OF THINGS 26
2.2.1 Modelos de comunicação 27
2.2.2 Protocolos de aplicação 28
2.2.3 Modelos e técnicas de autenticação 28
2.2.4 Identificação em dispositivos IoT 29
2.3 SELF-SOVEREIGN IDENTITY (SSI) 30
2.3.1 Decentralized Identifiers (DIDs) 31
2.3.1.1 DID methods 31
2.3.1.2 DID documents 32
2.3.1.3 Uso de DIDs 33
2.3.2 Verifiable Credentials (VCs) 34
2.3.2.1 Estrutura de confiança 36
2.3.2.2 Provas de conhecimento zero (ZKP) 38
2.3.3 Agentes SSI 39
2.3.4 Relação com a tecnologia blockchain 40
2.4 DEFINIÇÃO DAS FERRAMENTAS E TECNOLOGIAS 40
2.4.1 Hyperledger Indy 40
2.4.2 Hyperledger Aries 42
2.4.3 Bibliotecas e ferramentas facilitadoras 43
2.4.3.1 Hyperledger Aries Cloud Agent Python (ACA-Py) 43
2.4.3.2 Von Network 44
2.5 CONSIDERAÇÕES FINAIS 45
3 REVISÃO SISTEMÁTICA DE LITERATURA 46
3.1 PLANEJAMENTO DE CONSTRUÇÃO DO PROTOCOLO 46
3.1.1 Justificativa da necessidade 46
3.1.2 Objetivo e questões de pesquisa 47

3.1.3 Fontes de pesquisas **47**
3.1.4 String de busca **47**
3.1.5 Critérios de inclusão e exclusão **48**
3.1.6 Avaliação da qualidade **49**
3.1.7 Procedimento de seleção **50**
3.2 SELEÇÃO E EXECUÇÃO 50
3.3 SUMARIZAÇÃO E ESCRITA DOS RESULTADOS DO MAPEAMENTO 51
3.3.1 Comparação geral entre os trabalhos **51**
3.3.2 Respostas as questões de pesquisa **51**
3.3.2.1 QPE1: Quais são os domínios de aplicação da SSI para IoT que os trabalhos analisados abordaram? 51
3.3.2.2 QPE2: Quais são os padrões, protocolos e tecnologias abordados nos estudos para implementar SSI em dispositivos IoT? 53
3.3.2.3 QPE3: Quais são os desafios e obstáculos na aplicação das tecnologias SSI em dispositivos IoT, e quais estratégias são propostas para superá-los? 54
3.3.2.4 QPE4: Como os estudos abordam questões de segurança, escalabilidade, eficiência e consumo de recursos na aplicação da SSI na IoT? 54
3.3.2.5 QPE5: Quais são as metodologias e critérios de avaliação utilizados pelos estudos para analisar a eficácia e eficiência das soluções de SSI em dispositivos IoT? 55
3.3.3 Lacunas e contribuições dos trabalhos **56**
3.4 CONSIDERAÇÕES FINAIS 59
4 ABORDAGEM BASEADA EM GATEWAY PARA DISPOSITIVOS IOT RESTRITOS 60
4.1 HIPÓTESE GERAL DE PESQUISA 60
4.2 CONTEXTO DE PRESERVAÇÃO DE EVIDÊNCIAS FORENSES . . 60
4.2.1 Descrição da arquitetura proposta **61**
4.3 ESCOPO DA ARQUITETURA 63
4.3.1 Edge computing **63**
4.3.1.1 Dispositivos IoT 64
4.3.1.2 Broker MQTT 64
4.3.2 Fog computing **64**
4.3.2.1 Broker MQTT client 65
4.3.2.2 Banco de dados IoT 65
4.3.2.3 Processos do sistema 65
4.3.3 Cloud computing **67**
4.3.3.1 Aries Cloud Agent 67

4.3.3.2 Rede Von . . . 67
4.3.4 Validadores e validação . . . 67
4.4 FERRAMENTAS E TECNOLOGIAS ADOTADAS . . . 68
4.5 SEQUÊNCIA COMPLETA DE AÇÕES . . . 69
4.5.1 Mais detalhes da arquitetura . . . 71
4.6 CONSIDERAÇÕES FINAIS . . . 74
5 DESENVOLVIMENTO DA PROPOSTA . . . 75
5.1 IMPLANTAÇÃO DA REDE LOCAL . . . 75
5.1.1 Etapas gerais para implantar a blockchain Von Network e o agente Aries ACA-Py . . . 75
5.1.1.1 Rede Von . . . 75
5.1.1.2 Aries Cloud Agent Python . . . 76
5.1.2 Ambiente com imagens docker . . . 76
5.1.2.1 Iniciar uma rede Von Network local . . . 76
5.1.2.2 Etapas gerais para criar agentes usando a biblioteca ACA-Py . . . 77
5.1.3 Broker MQTT e simulador de dispositivo IoT . . . 80
5.2 DESIGN DA ARQUITETURA E IMPLEMENTAÇÃO DO GATEWAY . 81
5.2.1 Visão geral da arquitetura . . . 81
5.2.2 Implementação do cliente MQTT . . . 82
5.2.2.1 Parâmetros iniciais . . . 82
5.2.3 Processamento de mensagens . . . 84
5.2.4 Etapas gerais para gerar e registrar DIDs com Indy SDK . . . 85
5.2.4.1 Parâmetros iniciais . . . 86
5.2.4.2 Operações com pool e Wallet . . . 86
5.2.4.3 Geração de DID/Verkey . . . 86
5.2.4.4 Construção de uma transação e registro no blockchain . . . 87
5.2.5 Controlador ACA-Py . . . 87
5.2.5.1 Conexão com Agente em nuvem . . . 88
5.2.5.2 Estabelecer comunicação com Agentes através de convites . . . 88
5.2.5.2.1 Conexão através do recebimento de convite . . . 89
5.2.5.2.2 Conexão através da criação convite . . . 90
5.2.5.3 Criação de schema . . . 92
5.2.5.4 Definição do modelo de credencial . . . 93
5.2.5.5 Emissão da credencial verificável . . . 94
5.2.5.6 Apresentação da credencial . . . 95
5.3 CONSIDERAÇÕES FINAIS . . . 97
6 RESULTADOS E DISCUSSÃO . . . 98
6.1 EXPERIMENTOS . . . 98
6.1.1 Objetivos do design do experimento . . . 98

6.1.1.1 Ciclo do experimento 98
6.1.1.2 Fases do experimento 99
6.1.1.3 Adaptações para a realização do experimento 100
6.1.2 Implementação do experimento 100
6.2 RESULTADOS 101
6.2.1 Fase 1: Capacidade de processamento de dados do gateway . . 102
6.2.1.1 Etapa 1: Teste de carga com baixa demanda 102
6.2.1.2 Etapa 2: Teste de carga com extrema demanda 105
6.2.2 Fase 2: Eficiência do registro de transações na blockchain . . . 107
6.2.3 Fase 3: Tempo de emissão de credenciais verificáveis 110
6.3 ANÁLISE E DISCUSSÃO 112
6.3.1 Considerações de segurança 113
6.3.2 Aplicação de credenciais verificáveis em outros setores 114
7 CONCLUSÃO 116
7.1 LIMITAÇÕES DO TRABALHO 117
REFERÊNCIAS 118
APÊNDICE A – CÓDIGOS PARA OPERAÇÕES COM POOL E WALLET 127

1 INTRODUÇÃO

Com o rápido crescimento e evolução contínua da Internet das Coisas (do inglês *Internet of Things - IoT*), surgem novos serviços que promovem interações complexas entre seus dispositivos e sistemas inteligentes. Contudo, o aumento do número de dispositivos conectados, que geram e processam grandes volumes de dados, suscita preocupações substanciais sobre a segurança e privacidade desses registros (ALGARNI *et al.*, 2021) (GUBBI *et al.*, 2013). A intensa interconectividade no ecossistema da IoT cria uma rede heterogênea e complexa, apresentando desafios na proteção adequada desses sistemas (ALDOWAH; REHMAN; UMAR, 2019). Isso pode resultar em interações menos seguras, comprometendo a integridade das operações.

Nesse contexto, a crescente demanda por soluções inovadoras tem impulsionado a pesquisa de novas tecnologias. Entre as abordagens que têm ganhado notoriedade nos últimos anos, destacam-se os Identificadores Descentralizados (*Decentralized Identifiers - DIDs*), as Credenciais Verificáveis (*Verifiable Credentials - VCs*) e a tecnologia Blockchain. Essas tecnologias permitem a criação de um sistema de identidade digital descentralizado, em que os usuários têm controle total sobre suas próprias informações. A convergência desses conceitos é considerada essencial para a realização do paradigma de Identidade Auto-soberana (*Self-Sovereign Identity - SSI*), respondendo às necessidades crescentes de segurança nas transações e interações digitais.

Ao utilizar DIDs, é possível estabelecer identificadores globalmente únicos e, através de VCs, representar informações de forma segura que podem ser verificadas através da criptografia. Esses recursos podem ser fortalecidos pela tecnologia blockchain, garantindo a autenticidade e a integridade das transações, propiciando comunicações seguras e imutáveis. Isso é particularmente importante dada a vulnerabilidade dos dados compartilhados, sujeitos a tentativas de adulteração. Essa necessidade se destaca, principalmente quando se considera o contexto da preservação de evidências forenses, desempenhando um papel importante na cadeia de custódia de dados.

No entanto, é importante observar que dispositivos IoT frequentemente operam com recursos restritos, como capacidade de processamento reduzida, espaço de armazenamento escasso, restrições de memória e bateria de curta duração, conforme discutido em estudos anteriores (ALGARNI *et al.*, 2021). Esses desafios representam um obstáculo para a implementação de DIDs e VCs em ecossistemas IoT, uma vez que os torna incapazes de sustentar a pilha de protocolos criptográficos necessários para viabilizar a adoção dessas tecnologias em dispositivos com recursos limitados, exigindo abordagens alternativas. Uma solução prática consiste em terceirizar o processamento de forma segura para um dispositivo externo mais poderoso, reduzindo assim o custo computacional de cálculos criptográficos e mantendo a segurança dos

dados (KORTESNIEMI *et al.*, 2019).

Nesse contexto de segurança de dados e dispositivos IoT, a investigação forense digital assume um papel ainda mais relevante, atuando como elemento central em uma ampla gama de investigações criminais, aproveitando as vastas informações disponíveis e as oportunidades oferecidas pelos dados eletrônicos para a detecção e comprovação de crimes.

Diante desses desafios e oportunidades, o presente estudo propõe uma abordagem de *Gateway SSI*, projetada para dispositivos IoT com recursos limitados, com o objetivo de fortalecer a cadeia de custódia de dados no contexto da preservação de evidências forenses. A arquitetura proposta incorpora tecnologias como DIDs, que são usados para a identificação única das entidades, e VCs, que são empregadas para certificar os dados gerados por esses dispositivos. Além disso, essa arquitetura é reforçada pela tecnologia blockchain, garantindo a autenticidade e a integridade dos dados.

1.1 PROBLEMA DE PESQUISA E MOTIVAÇÃO

Novos desafios têm surgido com o crescente volume de dados, potencializados pela popularidade das tecnologias IoT. De acordo com um estudo realizado pela (STATISTA, 2020), a base total de dispositivos conectados à Internet das Coisas em todo o mundo está prevista para atingir 30,9 bilhões de unidades até 2025. A International Data Corporation (IDC, 2019) faz estimativas ainda maiores, prevendo que em 2025, serão cerca de 41,6 bilhões de dispositivos IoT conectados, gerando 79,4 *zettabytes* de dados. No entanto, essa expansão não tem ocorrido de forma segura, levantando preocupações sobre a integridade, confiabilidade e autenticidade dos dados.

Essas preocupações são particularmente relevantes no campo da investigação forense. Ao longo de todas as fases da investigação forense, as evidências digitais são suscetíveis a influências externas e contato com vários fatores. A admissibilidade legal de evidências digitais é a capacidade dessas evidências serem aceitas como prova em um tribunal de justiça. O valor probatório das evidências digitais só pode ser preservado se for possível estabelecer que os registros são precisos, ou seja, quem os criou, quando foram criados e se não houve alterações (SHAH; SALEEM; ZULQARNAIN, 2017).

Esses elementos desempenham um papel fundamental na criação de uma cadeia de custódia dos dados, conceito que, conforme (ĆOSIĆ, J.; ĆOSIĆ, Z., 2012) *apud* (STOYANOVA *et al.*, 2020), refere-se ao controle rigoroso e à auditoria do material de evidência original utilizado para fins legais. Este cuidado meticuloso é essencial para garantir a integridade e autenticidade das evidências digitais em todas as etapas da investigação forense.

De acordo com Arshad, Jantan e Abiodun (2018) durante os processos judiciais, as evidências eletrônicas são muitas vezes recebidas com extrema suspeita e incerteza, embora em algumas situações isso seja justificado. O uso de técnicas forenses cientificamente não comprovadas é amplamente criticado nos procedimentos legais atuais. Além disso, as características altamente distintas e dinâmicas dos dados eletrônicos, combinadas com a legislação e as leis de privacidade existentes, continuam a representar desafios significativos para a apresentação sistemática de evidências em um tribunal de justiça (ARSHAD; JANTAN; ABIODUN, 2018).

Nesse cenário, a heterogeneidade, interoperabilidade, ausência de padronização e os volumes massivos de dados em redes IoT trazem inúmeras possibilidades relacionadas à adulteração. Ao se explorar abordagens mais contemporâneas e potencialmente mais seguras para suprir essas necessidades, como DIDs, VCs e blockchain, identificam-se algumas limitações. Estas incluem desempenho, quantidade de energia suficiente para lidar com operações criptográficas, espaço de armazenamento não volátil para guardar códigos e chaves criptográficas, e fonte de entropia suficiente para a geração dessas chaves (KORTESNIEMI *et al.*, 2019).

Portanto, ainda existem áreas de pesquisa abertas relacionadas à aplicação de tecnologias baseadas em SSI em dispositivos IoT com recursos limitados. No contexto dessas aplicações, a preservação de evidências forenses é um aspecto importante, especialmente para o desenvolvimento de uma abordagem eficaz. Nesse cenário, a garantia da segurança, que engloba a integridade, rastreabilidade e verificação da autenticidade dos dados originados por esses dispositivos, é um elemento fundamental na administração da cadeia de custódia dessas informações.

1.2 OBJETIVOS

Nas próximas seções, serão descritos o objetivo geral e os objetivos específicos.

1.2.1 Objetivo geral

O objetivo geral deste trabalho é propor uma abordagem para a cadeia de custódia de dados, utilizando uma aplicação que funciona como gateway entre dispositivos IoT restritos e agentes Hyperledger Aries em nuvem. Esta abordagem emprega DIDs e VCs para certificar os dados gerados por esses dispositivos na origem. Adicionalmente, a tecnologia blockchain é utilizada para registro e verificação, visando garantir a autenticidade, integridade e rastreabilidade desses dados.

1.2.2 Objetivos específicos

Para alcançar o objetivo principal, são definidos os seguintes objetivos específicos:

a) Analisar soluções existentes que aplicam o conceito de SSI no contexto da IoT e explorar potenciais cenários de aplicação.

b) Criar uma arquitetura de *proxy* para redes IoT que incorpora DIDs para identificação única das entidades e VCs para certificação e apresentação de dados.

c) Empregar a tecnologia blockchain para registrar e verificar DIDs e VCs, com o objetivo de assegurar a integridade, autenticidade e rastreabilidade dos dados.

d) Implementar a arquitetura proposta em um ambiente controlado e realizar testes para avaliar seu desempenho.

e) Avaliar os resultados dos testes usando métricas de desempenho para discutir a escalabilidade e a viabilidade da arquitetura desenvolvida no contexto da gestão da cadeia de custódia de dados.

1.3 JUSTIFICATIVA

De acordo com (ĆOSIĆ, J.; ĆOSIĆ, Z., 2012), simplesmente conhecer a localização atual das provas originais não é suficiente para o tribunal. É imprescindível ter registros precisos que rastreiem o material de prova durante todo o tempo. Nesse contexto, (STOYANOVA *et al.*, 2020) argumentam que, na forense convencional, o processo deve começar quando os investigadores coletam uma peça de evidência na cena do crime e deve terminar com a apresentação do material de evidência no tribunal.

(ALEX; KISHORE, 2017) *apud* (STOYANOVA *et al.*, 2020) afirmam que o propósito da Cadeia de Custódia, um dos aspectos fundamentais em todas as investigações forenses, é fornecer informações claras e detalhadas sobre quando e como a evidência foi coletada, preservada, analisada e apresentada. Além disso, segundo (LONE, 2017), (ĆOSIĆ, J.; ĆOSIĆ, Z., 2012) e (NIK ZULKIPLI; ALENEZI; WILLS, 2017) é necessário provar que o material de evidência não foi alterado ou modificado durante todas as etapas da investigação forense.

Em uma perspectiva relacionada, (RAYES; SALAM, 2019) no estudo sobre IoT, discorrem que a salvaguarda da confidencialidade dos dados assume uma grande importância, garantindo que apenas as partes autorizadas tenham a capacidade de compreender as mensagens trocadas. Paralelamente, a integridade dos dados desempenha um papel de destaque, assegurando que as informações compartilhadas permaneçam íntegras e imunes a comprometimentos ou manipulações por parte de terceiros. Adicionalmente, o conceito de não-repúdio emerge como um fator fundamental, impedindo que uma entidade negue a realização de uma ação que tenha efetuado.

No estudo feito pela (SOVRIN, 2020), é ressaltado que os dispositivos IoT, além das funções tradicionais de identificação, autenticação e autorização, apresentam um desafio adicional: assegurar a privacidade, integridade e origem dos dados. É imprescindível que a cadeia de fornecimento de dados adote um padrão ético equiparável ao

aplicado em relação a bens físicos ou serviços. À medida que se adentra em um cenário digital cada vez mais abrangente, torna-se fundamental estabelecer uma cadeia de custódia dos dados, proporcionando uma rastreabilidade rigorosa, para promover os princípios de dados éticos.

(CURRAN; HOWARD, 2021b) argumentam que por meio da utilização de sensores de medição em dispositivos IoT, torna-se viável efetuar diversas medições, como por exemplo as emissões de gases de efeito estufa em uma unidade fabril, permitindo a transmissão segura dos dados através da emissão de credenciais verificáveis. Este processo assegura a integridade dos dados desde o momento da geração, evitando possíveis adulterações posteriores. Tais medidas estabelecem uma sólida base de confiança em relação aos dados adquiridos, desde que o próprio dispositivo seja intrinsecamente confiável. A confiabilidade do dispositivo pode ser atestada por auditores independentes ou por entidades reguladoras governamentais, as quais certificam tanto o funcionamento adequado do dispositivo quanto a legitimidade das credenciais verificáveis emitidas.

O estudo realizado por (FEDRECHESKI *et al.*, 2020) sustenta a aplicação do paradigma SSI na IoT e explora detalhadamente os conceitos subjacentes. Os autores destacam que os DIDs podem ser empregados para a identificação criptográfica, enquanto as VCs oferecem um meio para a divulgação de atributos autenticados e sensíveis à privacidade. Além disso, o estudo avalia a adequação do paradigma SSI para a IoT, demonstrando que a SSI é mais eficaz do que a utilização de tecnologias como PGP e certificados *X.509*.

Contudo, (SOVRIN, 2020) destaca que em ecossistemas IoT, abordagens que se apoiam em criptografia requerem recursos adequados nos dispositivos. No entanto, é comum que os fabricantes desenvolvam dispositivos IoT com desempenho, energia, armazenamento não volátil e fontes de entropia limitados, visando reduzir os custos de componentes e facilitar a disseminação generalizada da implantação de IoT.

(KORTESNIEMI *et al.*, 2019) complementa ao explicar que a implantação direta de DIDs em dispositivos IoT nem sempre é factível ou desejável, principalmente devido a restrições de recursos ou preocupações de segurança. Em cenários nos quais essa aplicação direta não é viável, uma abordagem baseada em *proxy* pode ser adotada. Na atualidade, a gestão de dispositivos IoT por meio de um *proxy* se destaca como uma solução comum. Um exemplo notável é o *gateway Web Things* (MOZILLA, 2023) ou a *Alexa Smart Home Skill API* (AMAZON, 2023). Isso ocorre porque, apesar de muitos dispositivos possuírem capacidades computacionais aceitáveis, sua confiabilidade para estabelecer conexões diretas com a Internet é comprometida.

Nesse contexto, é possível fundamentar a necessidade de uma arquitetura voltada para a cadeia de custódia de dados em dispositivos IoT restritos. Conforme ressaltado pelos autores, a preservação da confidencialidade, integridade, rastreabilidade

e a incorporação do conceito de não-repúdio tornam-se fatores fundamentais.

Essa abordagem permite que dispositivos IoT conectados a um agente emissor, tenham os dados assinados em sua origem antes de transmiti-los. Isso confere aos dados uma resistência à adulteração e torna sua verificação possível, o que, por sua vez, reforça a autenticidade e a imutabilidade do processo. Como resultado, a integridade das informações coletadas pode ser mantida, permitindo compartilhamento, divulgação seletiva e verificação por meio de credenciais verificáveis, um fator de extrema relevância no contexto de preservação de evidências forenses.

1.4 MÉTODO

Este estudo foi conduzido seguindo uma abordagem quantitativa, estruturada por meio de uma pesquisa exploratória que englobou uma revisão sistemática da literatura e uma fase experimental.

Inicialmente, foram examinadas soluções literárias relacionadas à criação de identidades auto-soberanas para a Internet das Coisas, incluindo o uso de blockchain, identificadores descentralizados e credenciais verificáveis. Esses estudos foram criteriosamente avaliados, levando em consideração a profundidade das soluções tecnológicas empregadas, a capacidade de garantir a integridade dos dados, a escalabilidade da implementação e os resultados das avaliações.

Posteriormente, foi concebida uma arquitetura de gateway direcionada para a certificação dos dados de redes IoT através da emissão de credenciais verificáveis. Essa arquitetura incluiu recursos para verificar a autenticidade por meio de blockchain e a divulgação seletiva de dados, abrangendo a privacidade. O desempenho do sistema foi analisado por meio de experimentos, com o objetivo de avaliar sua escalabilidade e aplicabilidade em cenários reais.

Os resultados alcançados e a análise de métricas de desempenho confirmam a eficiência da arquitetura em garantir a integridade, autenticidade dos dados e a segurança, criando um sólido fundamento para futuras melhorias. Este estudo representa um avanço, evidenciado pela publicação do artigo "*Ensuring Data Security in the Context of IoT Forensics Evidence Preservation with Blockchain and Self-Sovereign Identities*" (SANTOS; LOFFI; WESTPHALL, 2023) na 19ª Conferência Internacional sobre Segurança em Sistemas de Informação (ICISS), uma conferência de classificação *Qualis* B1. Este reconhecimento sublinha a importância e a aplicabilidade da pesquisa. Com base nos dados desses experimentos, foram elaboradas discussões e conclusões abrangentes sobre a viabilidade da arquitetura proposta.

1.5 ORGANIZAÇÃO DO TRABALHO

O restante desta dissertação de mestrado é organizada da seguinte maneira: o Capítulo 2 apresenta o referencial teórico; O Capítulo 3 realiza uma revisão sistemática da literatura; O Capítulo 4 descreve a arquitetura do modelo proposto; O Capítulo 5 detalha como o sistema foi implementado e implantado; O Capítulo 6 apresenta os experimentos realizados, seus resultados e discussão; Finalmente, o Capítulo 7 apresenta as conclusões do estudo e sugere possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais para este trabalho. Faz-se uma breve abordagem à cadeia de custódia de dados digitais, e aprofundam-se os conceitos e tecnologias por trás da Internet das Coisas. Em seguida, explora-se a identidade auto-soberana, com ênfase nos principais facilitadores, como os identificadores descentralizados e as credenciais verificáveis. Além disso, abordam-se outros conceitos intrinsecamente relacionados, como os protocolos de comunicação e as provas de conhecimento zero (ZKP). Por fim, discutem-se as principais tecnologias que suportam esses conceitos, baseadas no projeto colaborativo de código aberto Hyperledger Indy e Aries, hospedado pela The Linux Foundation (INDY, 2022). A discussão fundamenta-se nas especificações desenvolvidas pelo World Wide Web Consortium (SPORNY *et al.*, 2022a), amplamente aceitas como padrões nos tópicos abordados.

2.1 PRESERVAÇÃO DE EVIDÊNCIAS FORENSES

De acordo com o National Institute of Standards and Technology (NIST), a ciência forense, em sua essência, representa a ponte entre os domínios da ciência e do direito, dedicando-se à aplicação de princípios científicos para esclarecer questões legais. No âmbito dessa disciplina, emerge a forense digital – um campo também conhecido como forense de computadores e redes – que, embora compartilhe a mesma finalidade geral, distingue-se pela especificidade de seu foco. A forense digital abrange a ciência dedicada à coleta, exame, análise e preservação de dados digitais, com a meticulosidade de manter intocada a integridade da informação e assegurar uma cadeia de custódia ininterrupta e meticulosa (NIST, 2006).

Ainda segundo o (NIST, 2006), esses dados consistem em informações digitais formatadas de maneiras específicas e emanam de uma diversidade de fontes. Com o avanço tecnológico, as organizações veem-se inundadas por dados oriundos de sistemas, computadores, dispositivos de rede, periféricos, dispositivos eletrônicos, entre outros. Este cenário amplia exponencialmente o escopo de aplicação da forense digital, que vai desde a investigação de crimes e violações de políticas internas até a reconstrução de incidentes de segurança de computadores.

Neste contexto, a cadeia de custódia é fundamental, referindo-se ao procedimento documentado que rastreia a posse, transferência, manipulação e localização de evidências, desde a coleta até a apresentação em tribunal. Este processo é fundamental para preservar a integridade das evidências e garantir que sejam aceitas como provas admissíveis em processos legais. A cadeia de custódia é aplicável tanto a evidências físicas quanto digitais, mas a natureza intangível das evidências digitais apresenta desafios únicos que exigem procedimentos específicos para sua gestão.

2.1.1 Cadeia de custódia de dados digitais

A cadeia de custódia de dados digitais é um processo crítico usado para garantir a integridade e a segurança de dados eletrônicos desde o momento em que são coletados até o seu uso final, frequentemente em contextos legais ou de investigação. Este processo assegura que os dados não foram alterados, comprometidos ou manipulados de qualquer maneira, mantendo sua validade e confiabilidade.

A conformidade com normas internacionais, como as séries ISO/IEC 27041, 27042, e 27043, é fundamental para estabelecer procedimentos confiáveis e reconhecidos para a gestão da cadeia de custódia de dados digitais. Essas normas fornecem diretrizes sobre como coletar, analisar, armazenar e apresentar evidências digitais de maneira que preserve sua integridade e admissibilidade em processos judiciais. O fluxo da cadeia de custódia de dados digitais envolve várias etapas importantes:

- Coleta de Dados: O processo começa com a coleta de dados digitais de diversas fontes, como computadores, dispositivos móveis, servidores ou sistemas de armazenamento em nuvem. Esta etapa deve ser realizada utilizando métodos que não alterem os dados originais (ISO/IEC 27037, 2012).
- Documentação: É crucial documentar meticulosamente todas as ações e procedimentos realizados nos dados desde o momento da coleta. Isso inclui quem coletou os dados, quando e onde a coleta ocorreu, e quaisquer outras informações relevantes. A documentação deve seguir um protocolo padrão para garantir a consistência (ISO/IEC 27042, 2015).
- Preservação: Após a coleta, os dados precisam ser preservados de forma segura para prevenir qualquer alteração, perda ou dano. Isso geralmente envolve fazer cópias exatas (imagens forenses) dos dados e armazená-las em locais seguros. A integridade das cópias deve ser verificada regularmente através de *checksums* ou *hashes* (ISO/IEC 27041, 2015).
- Armazenamento e Transporte: Os dados e suas cópias devem ser armazenados e, se necessário, transportados de maneira segura. Isso inclui o uso de contêineres seguros, criptografia e outros métodos de proteção. O transporte de dados deve ser documentado, incluindo informações sobre o remetente, o destinatário e a maneira como os dados foram transportados (ISO/IEC 27043, 2015).
- Análise: A análise dos dados deve ser feita em cópias dos dados originais para preservar a integridade dos dados. Os analistas devem documentar todos os passos realizados durante a análise, incluindo as ferramentas e métodos utilizados (ISO/IEC 27042, 2015).
- Relatório: Após a análise, é gerado um relatório detalhando as descobertas. Este relatório deve incluir uma descrição de como os dados foram coletados,

preservados, analisados e quaisquer conclusões ou resultados da análise (ISO/IEC 27043, 2015).

- Apresentação: Os dados e as descobertas podem ser apresentados em contextos legais, como em tribunais, onde a cadeia de custódia documentada será essencial para estabelecer a admissibilidade dos dados como evidência (ISO/IEC 27043, 2015).

As características das evidências digitais tornam o manejo da cadeia de custódia mais complicado e complexo, qualquer violação da cadeia de custódia pode comprometer a confiabilidade dos dados e, por consequência, sua utilidade em contextos legais ou de investigação. Vários pesquisadores contribuíram com soluções para a cadeia de custódia digital de diferentes pontos de vista, abordando os problemas e desafios enfrentados nesta área (PRAYUDI, Y.; SN, 2015).

Além disso, algumas práticas são fundamentais, como é a utilização de métodos de criptografia avançados, essencial para proteger dados durante o armazenamento e o transporte. A criptografia não só ajuda a manter a confidencialidade dos dados, mas também assegura que qualquer tentativa de manipulação seja facilmente detectável (WIDATAMA; PRAYUDI, B., 2018). Em decorrência disso, a tecnologia blockchain, vem sendo cada vez mais explorada, como um meio de garantir a integridade e a imutabilidade da cadeia de custódia de dados digitais. Por meio da criação de um registro distribuído e resistente a adulterações, a blockchain pode oferecer um método transparente e seguro para registrar cada etapa da cadeia de custódia, desde a coleta até a apresentação dos dados em tribunal (BONOMI; CASINI; CICCOTELLI, 2018).

Apesar de algumas definições utilizadas nesta metodologia serem de origem mais antiga, elas continuam pertinentes e são reforçadas pelo desenvolvimento constante e futuro da Internet das Coisas. À medida que a utilização de dispositivos IoT se expande, a importância da cadeia de custódia de dados torna-se ainda mais crítica. Desse modo, a evolução da forense digital deve acompanhar o ritmo acelerado da inovação tecnológica, adaptando-se para gerir de forma eficaz a complexidade e a diversidade dos dados gerados pelo universo da IoT.

2.2 INTERNET OF THINGS

A Internet das Coisas representa a ampliação do poder da Internet, viabilizando a interconexão de diversos dispositivos físicos para a troca contínua de informações. Em essência, a IoT constitui um ecossistema de dispositivos computacionais interligados, cada qual com uma identidade única, habilitados a comunicar-se entre si. Esse entrelaçamento possibilita não apenas a interação entre pessoas e organizações, mas também a tomada de decisões estratégicas de grande relevância (ALGARNI *et al.*, 2021).

Essa rede, composta por objetos físicos, tais como sensores, atuadores e dispositivos em geral, possui a capacidade de coletar, trocar e transmitir dados por meio da Internet. Isso viabiliza interações e automação entre esses objetos, visando aprimorar a eficiência, conveniência e funcionalidade em variados contextos.

Segundo (RAYES; SALAM, 2019), a IoT, em sua forma mais simples, pode ser considerada como uma rede de elementos físicos potencializados por:

- Sensores: para coletar informações;
- Identificadores: para distinguir e rastrear dispositivos, sensores;
- *Software*: para analisar os dados;
- Conectividade à Internet: para comunicar e notificar.

Os sensores e atuadores desempenham um papel fundamental na viabilização de soluções de IoT em uma ampla gama de setores, abrangendo desde cidades inteligentes até agricultura, de cuidados de saúde a sistemas de transporte inteligentes. Sensores e atuadores são os elementos que viabilizam a capacidade desses dispositivos de computação ubíquos, dotados de conexões de rede, de coletar informações (realizando detecções e percepções) do mundo físico e, posteriormente, influenciar esse ambiente por meio de atuações controladas (MOYER, 2019).

A seguir, serão discutidas questões fundamentais para a IoT, abrangendo modelos de comunicação, protocolos de aplicação, bem como modelos de autenticação e identificação.

2.2.1 Modelos de comunicação

Em relação aos modelos de comunicação, (TSCHOFENIG *et al.*, 2015) delineia uma estrutura que abrange quatro modelos de comunicação frequentemente empregados por dispositivos IoT. Os modelos em questão compreendem:

- *Device-to-Device*: Nessa abordagem, dois ou mais dispositivos estabelecem conexão utilizando um protocolo de comunicação específico, permitindo a troca de mensagens e informações entre eles.
- *Device-to-Cloud*: Nesse modelo de comunicação, os dispositivos IoT se conectam diretamente a um serviço de nuvem na Internet, permitindo a troca de dados e o controle do fluxo de mensagens.
- *Device-to-Gateway*: Nesse formato, um software aplicativo opera em um dispositivo de *gateway* local, desempenhando o papel de intermediário entre o dispositivo em questão e o serviço de nuvem. O *gateway* oferece uma gama de serviços, incluindo segurança, conversão de dados e protocolos.
- *Back-End Data-Sharing*: Este é um padrão de comunicação utilizado quando os dados dos dispositivos precisam ser combinados com serviços de tercei-

ros para fornecer recomendações ou análises avançadas. Ele viabiliza aos usuários a exportação e análise de dados provenientes de dispositivos IoT presentes em um serviço de nuvem, permitindo a integração destes com informações de outras origens.

2.2.2 Protocolos de aplicação

A camada de aplicação IoT é responsável por fornecer serviços, garantindo uma comunicação eficaz entre os dispositivos IoT de baixo custo e recursos. Além disso, ela estabelece um conjunto de protocolos, como CoAP (*Constrained Application Protocol*), MQTT (*Message Queuing Telemetry Transport*), XMPP (*Extensible Messaging and Presence Protocol*) e AMQP (*Advanced Message Queuing Protocol*), para facilitar a troca de mensagens no nível de aplicação (SHARMA, C.; GONDHI, 2018).

O AMQP representa uma arquitetura de protocolo *publish-subscriber* de padrão aberto. Para garantir a segurança da transmissão, o protocolo utiliza o TLS (*Transport Layer Security*). No entanto, devido à sua natureza, o AMQP não é um protocolo leve e não é adequado para sensores autônomos em termos de consumo de memória, largura de banda e uso de energia (TIGHTIZ; YANG, 2020).

O CoAP é um protocolo da Internet que opera segundo o modelo de solicitação-resposta e adere aos princípios REST (*Representational State Transfer*), funcionando sobre o UDP (*User Datagram Protocol*) com o intuito de reduzir a sobrecarga e a utilização de largura de banda. Dada a natureza não confiável do UDP, o CoAP aborda a confiabilidade por meio do envio contínuo de mensagens de confirmação até que a confirmação seja recebida do outro ponto da comunicação (TIGHTIZ; YANG, 2020).

O XMPP é um dos protocolos IoT de padrão aberto, apresenta um suporte abrangente para um modelo de publicação-assinatura que opera tanto de forma assíncrona quanto síncrona. Isso possibilita a troca de mensagens usando o formato XML (*Extensible Markup Language*) (TIGHTIZ; YANG, 2020).

O MQTT é uma arquitetura de protocolo de publicação-assinatura que opera como um protocolo de padrão aberto, com a capacidade de operar sobre TCP/IP. Nessa estrutura, três elementos principais estão envolvidos: o *publisher* (emissor), o *subscriber* (receptor) e o *broker* (intermediário). *publishers* e *subscribers* realizam a troca de mensagens associadas a tópicos específicos, intermediada pelos *brokers*. Estes últimos garantem a segurança e autorização da comunicação por meio do uso confiável do protocolo TLS (TIGHTIZ; YANG, 2020).

2.2.3 Modelos e técnicas de autenticação

Com base na literatura, foram encontradas técnicas propostas para autenticação em IoT (EL-HAJJ *et al.*, 2019).

- Fator de Autenticação/Esquema Baseado em Chave Pública: Esta categoria de esquema de autenticação é baseada em uma parte apresentando informações à outra parte para se autenticar. Também conhecido como autenticação baseada em identidade e usa uma combinação de *hash*, algoritmos simétricos e assimétricos. Os dispositivos podem possuir certificados digitais únicos que são usados para verificar sua identidade.
- Esquema baseado em *token*: Este esquema é baseado no uso de *tokens* de identificação para autenticar o usuário ou dispositivo que é criado por um servidor como o protocolo *OAuth2* ou *openID*.
- Autenticação Baseada em Credenciais: Envolve o uso das credenciais (nome de usuário/senha) sempre que houver necessidade de troca de dados.
- Esquema Baseado em Hardware: Neste método, para autenticação de dispositivos IoT, utiliza-se atributos físicos do hardware, incluindo características únicas como as Unidades PUF (*Physical unclonable function*) e geradores de números aleatórios, além de componentes específicos como o TPM *(Trusted Platform Module*) para armazenar e processar chaves de autenticação.

2.2.4 Identificação em dispositivos IoT

A maioria dos dispositivos é identificada por meio de identificadores explícitos, como endereço IP *(Internet Protocol*), endereço MAC (*Media Access Control*) e outras identidades de rede. Além disso, também é possível utilizar outros tipos de identificadores, como número de série e UUID (*Universally Unique Identifier*) (CHOWDHURY, R. R. *et al.*, 2020).

Os identificadores em um padrão IoT podem ser normalmente divididos nas seguintes categorias: identificador de objeto, identificador de comunicação e identificador de aplicativo. Os identificadores de objeto representam objetos físicos ou virtuais. Códigos de barras e identificador de radiofrequência (*Radio Frequency Identification - RFID*) são alguns exemplos de identificadores que não podem ser usados para endereçamento ou comunicação.

Os identificadores de comunicação identificam exclusivamente os nós em uma rede com recursos de comunicação. O identificador de comunicação, que geralmente é feito de um endereço IP, é usado para endereçamento. Os identificadores de aplicativos identificam aplicativos da camada de serviço, objetos e entidades lógicas etc. Identificador Uniforme de Recursos (*Uniform Resource Identifier - URI*) e Localizador Uniforme de Recursos (*Uniform Resource Locator - URL*) são alguns exemplos de identificadores de aplicativos (AFTAB *et al.*, 2020).

Ainda segundo (AFTAB *et al.*, 2020), para estabelecer a comunicação entre aplicativos da IoT executados em diversas plataformas, é imperativo adotar uma meto-

dologia de identificação unificada. A questão da interoperabilidade dos identificadores se destaca como um desafio essencial, dada a coexistência de múltiplas plataformas IoT distintas. A natureza heterogênea dos dispositivos de hardware em diversas plataformas IoT acrescenta ainda mais complexidade a essa tarefa. Diversos sistemas de identificação universais estão disponíveis, incluindo o OID (*Object IDentifier*), EPC (*Electronic Product Code*), UUID e IMEI (*International Mobile Equipment Identity*), entre outros.

No entanto, não há um único sistema de identificação universalmente exclusivo, e as diferentes plataformas IoT adotam esquemas distintos. Isso resulta em dificuldades na realização da interoperabilidade entre diversas plataformas. Dessa maneira, à medida que a IoT continua a evoluir, a integração de tecnologias avançadas, como as baseadas em SSI e blockchain, oferece novas oportunidades para aprimorar a segurança, a privacidade e a interoperabilidade.

2.3 SELF-SOVEREIGN IDENTITY (SSI)

A Identidade Auto-soberana é um conceito que descreve a capacidade de um indivíduo controlar e gerenciar sua identidade digital de forma descentralizada. Essa abordagem foi proposta pela primeira vez por (ALLEN, 2016) e tem sido usada como referência por vários autores na literatura. Ele propôs 10 princípios para alcançar esse objetivo, sendo eles: existência, controle, acesso, transparência, persistência, portabilidade, interoperabilidade, consentimento, minimização e proteção. Embora ainda não exista um consenso sobre a SSI, o conceito amplamente aceito é descrito como um sistema onde o indivíduo é livre para reivindicar e gerenciar sua própria identidade sem a necessidade de uma parte confiável centralizada (BRUNNER *et al.*, 2020).

A identidade auto-soberana permite que os usuários escolham quais informações de identidade compartilhar e com quem, bem como ter acesso a suas próprias informações de identidade a qualquer momento. Em vez de depender de terceiros, como empresas ou governos, para gerenciar e armazenar suas informações de identidade, os usuários têm a capacidade de criar, armazenar e controlar suas próprias identidades digitais através de tecnologias descentralizadas, como blockchain.

Com o avanço da tecnologia blockchain, há otimismo em torno da possibilidade de implementar o conceito de SSI, que pode revolucionar a forma como interagimos uns com os outros na internet no futuro. A tecnologia blockchain é vista como a base técnica necessária para a realização desse conceito, permitindo que os indivíduos reivindiquem e gerenciem sua própria identidade, sem a necessidade de uma parte confiável centralizada (FERDOUS; CHOWDHURY, F.; ALASSAFI, 2019). A seguir são descritos alguns dos conceitos e recursos chave que podem viabilizar a ideia de identidade auto-soberana.

2.3.1 Decentralized Identifiers (DIDs)

Identificadores exclusivos são amplamente utilizados por indivíduos e organizações em diversos contextos. Eles servem como meios de comunicação, como números de telefone, endereços de e-mail e nomes de usuário em mídias sociais, além de serem utilizados como números de identificação para documentos como passaportes, carteiras de motorista e seguros de saúde. Também são utilizados como identificadores de produtos, como números de série, códigos de barras e RFIDs. Além disso, URIs são utilizados para recursos na web e cada página da web possui uma URL exclusiva (SPORNY *et al.*, 2022a).

Identificadores descentralizados são um tipo de identificador globalmente exclusivo que permite que uma entidade seja identificada de maneira verificável, persistente (desde que o controlador DID deseje) e não requer o uso de um registro centralizado. Os DIDs permitem um novo modelo de identidade digital descentralizada, muitas vezes referido como identidade auto-soberana ou identidade descentralizada (SPORNY *et al.*, 2022a).

Comparativamente, ao desejar autenticação em um site atual, os usuários informam: "Este é meu nome de usuário". O site, então, requer a senha do usuário e a valida em seu banco de dados. No contexto dos DIDs, o usuário pode informar a um site: "Este é o meu DID". O site tem a capacidade de recuperar a chave pública e solicitar a assinatura de um desafio utilizando sua chave privada. O site, posteriormente, verifica essa assinatura empregando a chave pública correspondente (JONG, 2020).

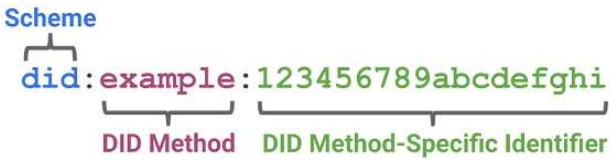
Os DIDs são fundamentais para soluções de identidade auto-soberana. Eles fornecem uma maneira escalável e segura de identificar entidades. Cada entidade possui um DID único, que pode ser resolvido por meio de um resolvedor DID, semelhante ao sistema de nomes de domínios (*Domain Name System* - DNS).

Assim como os URLs os DIDs podem ser resolvidos. Quando um DID válido é passado para um software chamado *DID Resolver*, ele funciona como um navegador que recebe uma URL, resolvendo o DID e retornando um documento. No entanto, em vez de retornar uma página da Web, um *DID Resolver* retorna um DIDDoc (*DID Document*), um documento JSON (*JavaScript Object Notation*) cujo formato é definido na especificação DID. Esse documento contém uma chave pública e um ponto de comunicação para que outras entidades possam enviar mensagens e solicitar a conexão (CURRAN; HOWARD, 2021b).

2.3.1.1 DID methods

Assim como existem tipos diferentes de URIs, todos em conformidade com o padrão URI, existem tipos diferentes de métodos DID, os quais devem estar em conformidade com o padrão DID (SPORNY *et al.*, 2022a).

Figura 1 – Exemplo simples de um identificador descentralizado (DID).



Fonte: (SPORNY *et al.*, 2022a).

Um DID é composto por três partes, conforme pode ser visto na Figura 1. O identificador do esquema URI, identificador para o método DID e identificador específico do método DID.

2.3.1.2 DID documents

De modo geral, o conceito de DID funciona como uma chave, enquanto o Documento DID representa o valor associado. Em outras palavras, um DID capacita a busca por um documento específico em um blockchain ou sistema distribuído. Este documento DID, formatado em JSON-LD (*JavaScript Object Notation for Linked Data*), serve para detalhar e descrever a entidade que é identificada pelo próprio DID. A estrutura do DID pode variar dependendo do método utilizado para criá-lo (CURRAN; HOWARD, 2021a).

Os componentes padronizados de um documento DID incluem: o próprio DID (para autodescrição); um conjunto de chaves públicas (para verificação); métodos de autenticação (para validação); *endpoints* de serviço (para interação); um carimbo de tempo (para rastreabilidade histórica); e uma assinatura (para assegurar integridade).

Figura 2 – Exemplo simples de um documento DID.

EXAMPLE 1: A simple DID document

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

Fonte: (SPORNY *et al.*, 2022a).

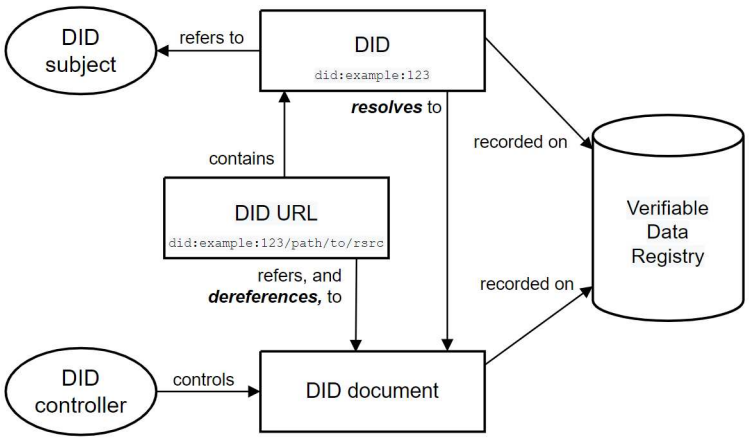
Conforme apresentado na Figura 2, um DID resolve para um documento, que é um conjunto de dados descrevendo o sujeito do DID. Este documento inclui mecanismos como chaves públicas criptográficas, que permitem ao sujeito do DID ou a um delegado autorizado autenticar-se e provar a associação com o DID. A delegação é um processo pelo qual o controle sobre algumas ou todas as capacidades de um DID podem ser transferidas para outro DID, permitindo ações em nome do sujeito original.

2.3.1.3 Uso de DIDs

Segundo (CURRAN; HOWARD, 2021a), existem dois tipos de DIDs, os privados e os públicos. Os DIDs privados são projetados para serem trocados entre duas partes específicas para criar um canal seguro, que nenhuma pessoa ou parte não autorizada pode acessar. Eles são úteis para organizações que precisam compartilhar informações confidenciais, como entre executivos de uma empresa ou advogados de uma empresa que trabalham em casos de clientes.

Já os DIDs públicos são benéficos para as partes que precisam ser publicamente identificáveis, como departamentos do governo que emitem IDs para cidadãos, participantes da cadeia de suprimentos (produtores, fabricantes, distribuidores etc.) ou prestadores de serviços de saúde. Eles são armazenados em um registro de dados público verificável (como o blockchain*)*, permitindo que qualquer entidade tenha acesso a esse DID e possa solicitar uma conexão com a entidade cujo DID é público (CURRAN; HOWARD, 2021a).

Figura 3 – Visão geral da arquitetura DID.



Fonte: (SPORNY *et al.*, 2022a).

No exemplo apresentado na Figura 3, o DID URL é utilizado para se referir a um sujeito DID e para resolver documentos DID, que contêm informações relacionadas ao DID, como chaves públicas criptográficas, serviços e interações. Os DIDs podem

ser registrados em registros de dados verificáveis, como livros contábeis distribuídos. Por fim, um controlador DID é a entidade responsável, seja uma pessoa, organização, dispositivo ou serviço, que tem a capacidade de modificar o documento DID.

Ainda segundo (CURRAN; HOWARD, 2021a), como os DIDs são descentralizados e não dependem de uma autoridade central única para emitir e gerenciar identificações, torna-se mais difícil para *hackers* e outras ameaças cibernéticas interceptar ou falsificar as identificações. Além disso, os DIDs podem utilizar a tecnologia blockchain, oferecendo uma camada adicional de segurança através da imutabilidade das informações registradas, garantindo que as informações da identidade sejam confiáveis e seguras. Os DIDs apresentam inúmeras possibilidades de uso, como melhorar a segurança de transações financeiras, facilitar a verificação de identificação em processos de votação eletrônica, proteger informações pessoais de indivíduos e organizações, além de possibilitar a autenticação segura de dispositivos IoT.

2.3.2 Verifiable Credentials (VCs)

Credenciais Verificáveis são informações digitais criptografadas emitidas por autoridades confiáveis, que servem para comprovar afirmações sobre uma pessoa. De modo geral, são um conjunto de informações digitais que podem ser usadas para provar a identidade de uma pessoa ou organização.

No contexto real, ao empregar o conceito de credenciais em papel, tais credenciais podem abranger uma variedade de documentos que atestam as habilidades ou qualificações de um indivíduo. Exemplos incluem carteira de motorista, passaporte, diploma acadêmico, dentre outros. Embora uma credencial possa ter diversos formatos, ela precisa conter algum tipo de declaração ou certificado emitido por uma fonte confiável, como uma carteira de motorista expedida pelo governo. Dessa forma, qualquer pessoa pode verificar a validade e legitimidade dessa credencial. Desse modo, esse modelo lida com reivindicações em um sistema baseado em confiança, uma vez que não existe um método infalível para provar a veracidade de uma reivindicação. Este sistema é composto por três atores: o emissor, o detentor e o verificador (CURRAN; HOWARD, 2021a).

Com o avanço da tecnologia, especialmente com a popularidade de impressoras, *scanners* e *softwares* de edição de imagem, a falsificação de documentos se tornou mais comum. Isso exige que os profissionais responsáveis por verificar a autenticidade dos documentos (verificadores) tenham habilidades especializadas para detectar documentos falsificados (CURRAN; HOWARD, 2021a).

Assim como o modelo de credenciais em papel, o modelo de credenciais verificáveis conta com três partes: um emissor, um titular e um verificador conforme evidenciado na Figura 4. A principal diferença entre os dois modelos é a forma como a verificação é realizada. Enquanto o modelo de credenciais em papel depende da

Figura 4 – Modelo de credencial de papel.

Fonte: (CURRAN; HOWARD, 2021a).

habilidade do verificador em reconhecer uma credencial alterada ou falsificada, o modelo de credencial verificável possui um registro de dados verificável, que é composto por chaves criptográficas e identificadores. Esses dados permitem a comprovação das informações.

Uma credencial verificável pode conter informações como: dados de identificação do titular, como foto, nome ou número de identificação; informações sobre a autoridade emissora, como governo municipal, estadual ou federal, agência nacional ou organismo de certificação; tipo de credencial, como passaporte, carteira de motorista ou cartão do plano de saúde; atributos ou propriedades específicas declaradas pela autoridade emissora; evidências de como a credencial foi emitida; e restrições, como data de validade ou termos de uso. As credenciais verificáveis podem conter as mesmas informações que as credenciais físicas e, além disso, incluir tecnologias de segurança, como assinaturas digitais, tornando-as mais seguras e confiáveis do que suas contrapartes físicas (SPORNY *et al.*, 2022b).

As credenciais verificáveis são emitidas por entidades confiáveis, como governos, universidades e empresas e contêm informações criptografadas que podem ser verificadas por qualquer pessoa que tenha acesso à chave pública do emissor. Isso significa que as informações contidas em uma credencial podem ser confiáveis e não podem ser alteradas ou falsificadas sem serem detectadas.

A Figura 5, demonstra uma representação fictícia de uma credencial verificável emitida por uma instituição educacional (o emissor) para um indivíduo (o titular da credencial). Os principais componentes são:

- *@context*: Estabelece o contexto no qual a credencial é interpretada, definindo termos especiais como "*issuer*" (emissor) e "*credentialSubject*" (sujeito da credencial), com significados específicos.
- *id*: Identifica a credencial, possibilitando referências precisas.
- *type*: Declara os tipos da credencial, como "*VerifiableCredential*" (Credencial Verificável) e "*UniversityDegreeCredential*" (Credencial de Grau Universitá-

Figura 5 – Exemplo de parte de credencial.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/3732",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": {
    "id": "did:example:76e12ec712ebc6f1c221ebfeb1f",
    "name": "Example University"
  },
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science and Arts"
    }
  }
}
```

Fonte: (SPORNY *et al.*, 2022a).

rio), definindo o formato e o conteúdo esperado.

- *issuer*: URL do emissor da credencial, que é a entidade responsável por emitir e garantir a confiabilidade da informação contida.
- *issuanceDate*: Indica a data de emissão da credencial.
- *credentialSubject*: Contém informações sobre o sujeito da credencial, no caso, o titular da credencial. Neste exemplo, a credencial afirma que o titular possui um diploma de bacharel em Ciências e Artes.

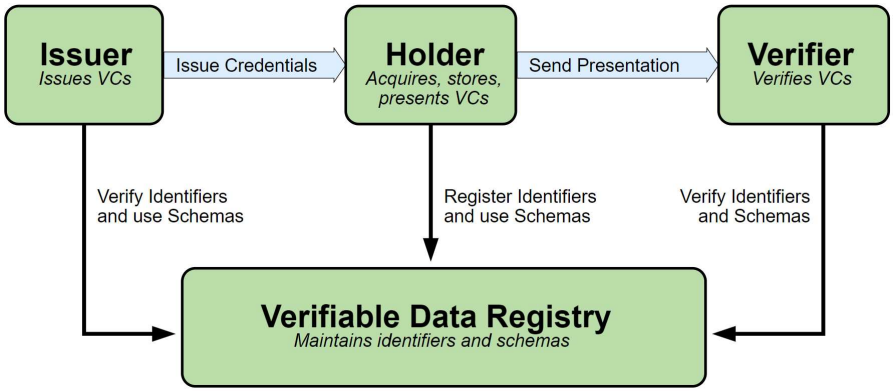
A credencial verificável proporciona uma maneira segura e confiável de compartilhar informações específicas, permitindo que as partes envolvidas verifiquem a autenticidade e a veracidade desses dados por meio de processos de verificação. Um exemplo prático dessa abordagem é o uso de uma carteira digital de identidade, onde os usuários podem armazenar e controlar suas informações pessoais de forma segura e privada.

2.3.2.1 Estrutura de confiança

No contexto da estrutura de confiança, aquele que é detentor de uma credencial verificável ocupa uma posição central em um triângulo de confiança, atuando como

intermediário entre o emissor e o verificador. O emissor deposita sua confiança no detentor da credencial, o qual, por sua vez, confia no verificador. Este, por sua vez, confia no emissor. Cada um dos papéis nesse triângulo pode ser desempenhado por um indivíduo, uma instituição ou um dispositivo IoT.

Figura 6 – Visão geral da arquitetura.



Fonte: (SPORNY *et al.*, 2022a).

Na Figura 6, são delineadas as funções centrais dos atores principais e suas interações dentro de um ecossistema de credenciais verificáveis. As funções são detalhadas a seguir:

- *Holder*: Uma função atribuível a uma entidade que detém uma ou várias credenciais verificáveis e tem a capacidade de gerar apresentações verificáveis a partir dessas credenciais. Exemplos de detentores abrangem estudantes, funcionários e clientes.
- *Issuer*: Uma função exercida por uma entidade ao emitir declarações sobre um ou mais assuntos específicos. Isso envolve a criação de credenciais verificáveis baseadas nessas declarações e sua subsequente transmissão a um detentor. Exemplificando emissores, temos corporações, organizações sem fins lucrativos, associações comerciais, governos e indivíduos.
- *Verifier*: Uma função desempenhada por uma entidade ao receber uma ou mais credenciais verificáveis, podendo ser inseridas opcionalmente em uma apresentação verificável para fins de processamento. Verificadores podem ser empregadores, pessoal de segurança, sites e outras entidades similares.
- *Verifiable Data Registry*: Uma função executada por um sistema para facilitar a criação e verificação de identificadores, chaves e outros dados pertinentes. Isso engloba esquemas de credenciais verificáveis, registros de revogação, chaves públicas do emissor e outros elementos essenciais para a utilização de credenciais verificáveis. Exemplos de registros de dados verificáveis com-

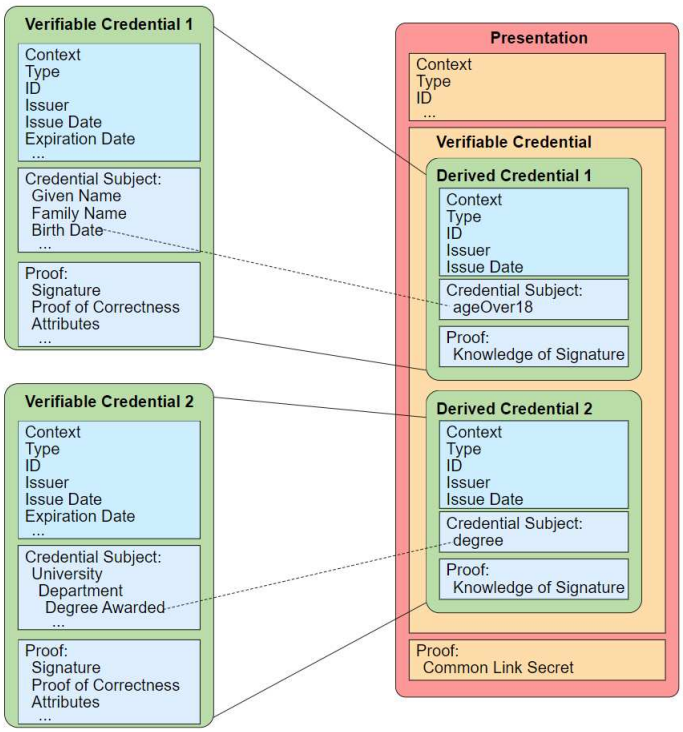
preendem bases de dados confiáveis, sistemas de dados descentralizados, bancos de dados de identificação governamental e sistemas de contabilidade distribuída.

2.3.2.2 Provas de conhecimento zero (ZKP)

Prova de Conhecimento Zero, também conhecido como *Zero-knowledge proof* (ZKP) em inglês, é uma técnica de criptografia que permite provar que se possui uma informação sem revelar o seu conteúdo, garantindo a privacidade dos dados e a segurança das transações. O modelo de credenciais verificáveis combina as implementações de prova de conhecimento zero e credenciais verificáveis com o objetivo de reduzir dados e aumentar a privacidade.

A Figura 7, demonstra a relação entre credenciais e credenciais derivadas em uma apresentação ZKP. Essa representação gráfica exemplifica como as informações parciais de cada credencial são utilizadas para derivar credenciais verificáveis durante o processo de apresentação, sem comprometer a privacidade das informações completas contidas em cada credencial.

Figura 7 – Esquema de apresentação de VCs por meio de ZKP.



Fonte: (SPORNY *et al.*, 2022a).

Com essa combinação, os provadores podem apresentar provas sem divulgar dados confidenciais e pessoais para os verificadores. Para aumentar a privacidade

e segurança, as reivindicações presentes em uma credencial verificável podem ser expostas na forma de um predicado ou divulgação seletiva de prova de conhecimento zero.

Os predicados em ZKP são utilizados sempre que um titular precisar comprovar que atende a um requisito associado a um de seus atributos sem revelar os dados relacionados a esse atributo. Um dos predicados suportados por VCs é "maior ou igual a (>=)". Um exemplo de como esse predicado pode ser usado em uma situação real seria quando uma pessoa quer comprovar que tem mais de 18 anos sem precisar revelar sua data de nascimento ou idade.

Já na prova de conhecimento zero de Divulgação Seletiva, o usuário pode criar uma apresentação para mostrar somente as informações relevantes (atributos) de uma credencial verificável para o verificador. No exemplo anterior, o indivíduo pode optar por revelar apenas a data de nascimento sem precisar revelar outros dados comumente encontrados em um documento de identificação em papel (CURRAN *et al.*, 2022) (DAVIE *et al.*, 2019).

2.3.3 Agentes SSI

É importante destacar como os identificadores descentralizados e as credenciais verificáveis são gerenciados e armazenados. Para essa finalidade, são utilizadas carteiras digitais de identidade, que são softwares específicos para armazenar DIDs e VCs. Além disso, para fazer parte do modelo de dados verificáveis e interagir com outras entidades, é necessário um software que permita gerenciar e usar os VCs mantidos nas carteiras dessas entidades. Esse software é chamado de "agente" e é usado por (ou em nome de) uma entidade para interagir com outros agentes.

Os agentes são softwares responsáveis por gerenciar e usar os DIDs e VCs armazenados nas carteiras digitais de identidade. Eles precisam de acesso à carteira para realizar operações criptográficas em nome da entidade representada. Algumas das tarefas que os agentes desempenham incluem o envio e recebimento de mensagens, cifragem e decifragem de informações, assinatura digital em nome da entidade, gerenciamento de informações na carteira e *backup*/restauração de informações. Alguns agentes também podem interagir com o blockchain, permitindo a adoção do modelo de dados verificáveis (SOVRIN, 2020) (CURRAN; HOWARD, 2021a).

Os agentes podem ser de três tipos diferentes: agentes de borda, agentes de nuvem e agentes móveis. A principal diferença entre eles é que os agentes de nuvem fornecem um ponto de extremidade público para interações com outros agentes, enquanto os agentes móveis são hospedados no dispositivo privado de uma pessoa ou entidade e os agentes de borda são executados na borda da rede em um dispositivo local (CURRAN; HOWARD, 2021a). A escolha do tipo de agente dependerá das necessidades de privacidade e recursos de computação/armazenamento e conectividade.

Para dispositivos IoT, utilizar um agente de borda é uma opção atrativa devido à limitação de recursos de computação e armazenamento, bem como conectividade. Como alternativa, é possível estabelecer uma comunicação segura com um agente de nuvem, que, por sua vez, se comunica com o livro-razão, já que os agentes de borda não possuem um ponto de extremidade público (CURRAN; HOWARD, 2021a) (TEAM ARIES, 2021).

2.3.4 Relação com a tecnologia blockchain

A blockchain apresenta características que se alinham com as propriedades desejadas de uma identidade auto-soberana. Por exemplo, a blockchain fornece essencialmente um domínio descentralizado que não é controlado por nenhuma entidade única. Os dados armazenados em qualquer blockchain estão prontamente disponíveis para qualquer entidade autorizada. Um proprietário de um dado específico tem controle total sobre ele e determina como esses dados podem ser compartilhados com outros usuários dentro do domínio blockchain, satisfazendo assim a propriedade de divulgação.

É essencial compreender o conteúdo armazenado na blockchain. De forma geral, nenhum dado privado é armazenado nela, o que proporciona maior preservação de privacidade em soluções que utilizam uma tecnologia de registro distribuído. A blockchain apenas mantém os DIDs públicos, esquemas, definições de credenciais e registros de revogação. Apesar de a criptografia utilizada atualmente ser considerada segura, não se pode garantir o mesmo no futuro. Por essa razão, é fundamental que nenhum dado privado esteja armazenado na blockchain (CURRAN; HOWARD, 2021a).

2.4 DEFINIÇÃO DAS FERRAMENTAS E TECNOLOGIAS

É importante compreender o contexto no qual estas tecnologias são aplicadas. Soluções baseadas em blockchain, a exemplo do Hyperledger Indy, emergem como alternativas promissoras, oferecendo uma infraestrutura descentralizada que enfrenta questões anteriormente discutidas. Na Subseção 2.4.1, explora-se o conceito do Hyperledger Indy e tecnologias correlatas que dele derivaram nas demais subseções, incluindo Hyperledger Aries, ACA-Py e *Von Network*.

2.4.1 Hyperledger Indy

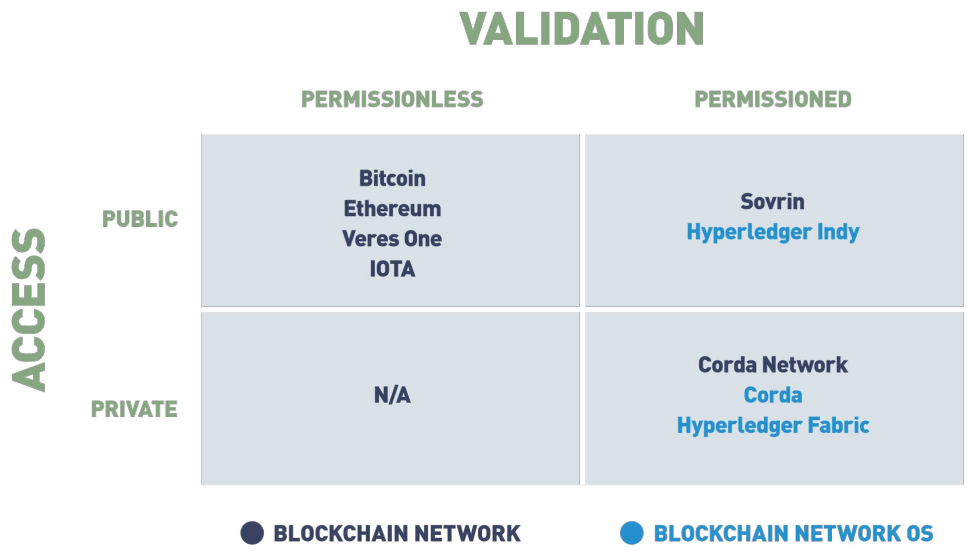
Hyperledger Indy foi a primeira estrutura blockchain "focada na identidade" juntando-se à Hyperledger em 2017. O código para Indy foi contribuído pela Sovrin Foundation. O Indy é um livro-razão distribuído desenvolvido especificamente para identidade descentralizada. O Indy inclui VCs com suporte ao ZKP, DIDs, um kit de de-

senvolvimento de software (*Indy SDK*) e uma implementação de um ledger distribuído público e autorizado (INDY, 2022).

No Hyperledger Indy, ao contrário de outras tecnologias de registros distribuídos (*Distributed Ledger Technologies* - DLTs), não é necessário o uso de incentivos. Como resultado, todas as transações que abrangem várias operações, como criação de DIDs, rotação de chaves de consulta, criação de esquema de credenciais, definição de credenciais e outras funcionalidades, podem apresentar um desempenho aprimorado. Além disso, o *SDK Indy* oferece suporte para o gerenciamento local de credenciais por meio de carteiras. Essas carteiras têm a capacidade de armazenar e gerenciar VCs e pares de chaves criptográficas (INDY, 2022).

A Figura 8 ilustra a comparação entre o blockchain Indy e outras blockchains amplamente reconhecidas.

Figura 8 – Comparação entre tecnologias blockchain.



Fonte: (CURRAN; HOWARD, 2021a).

Com o amadurecimento do Hyperledger Indy, tornou-se evidente que as capacidades criptográficas desenvolvidas nesse contexto poderiam ser aplicadas em diversos projetos dentro e até mesmo fora do ecossistema Hyperledger. Nesse contexto, no ano de 2018, foi deliberada a decisão de transferir o repositório de código do Indy relacionado à criptografia, denominado *indy-crypto*, para um projeto independente: o Hyperledger Ursa. Essa iniciativa visava a ampla utilização dessas funcionalidades criptográficas em diferentes contextos (CURRAN; HOWARD, 2021a).

O repositório *indy-sdk* era responsável pela criação de agentes que interagiam com o livro-razão Indy e entre si. Até meados de 2018, os grupos desenvolviam agentes de forma isolada, incapazes de se comunicar com agentes de outros grupos. O *Indy Agents Working Group* surgiu para possibilitar essa interoperabilidade, definindo proto-

colos padrão. Surgiu então a ideia do Hyperledger Aries, que visava permitir agentes que usassem DIDs e credenciais verificáveis de diversos ecossistemas. Aceito como projeto oficial em 2019, o Aries visa a interoperabilidade entre sistemas para construir uma identidade auto-soberana mais abrangente (INDY, 2022).

No Hyperledger Indy, o livro-razão é baseado em uma árvore Merkle e, diferentemente de outros DLTs, o uso de incentivos não é obrigatório. Como resultado, todas as transações que abrangem diversas operações, como criação de DID, rotação de chaves de consulta, criação de esquema de credenciais, definição de credenciais e outras funcionalidades, podem apresentar melhor desempenho (INDY, 2022). Além disso, o Indy SDK fornece suporte para gerenciamento de credenciais locais por meio de carteiras. Essas carteiras têm a capacidade de armazenar e gerenciar VCs e pares de chaves criptográficas.

Em termos de algoritmo de consenso, Indy emprega o PBFT (*Practical Byzantine Fault Tolerance*), o que permite um alto rendimento de transações (MASOOD; FARIDI, 2019). No entanto, é importante ressaltar os limites de segurança do PBFT em cenários de coalizão de participantes maliciosos. Embora os resultados de desempenho apresentados (LUX *et al.*, 2019) demonstrem que Indy atende aos critérios de escalabilidade global em relação à velocidade de consulta da blockchain, é essencial considerar os potenciais desafios de segurança associados ao PBFT.

Em relação à privacidade, o modelo de Gerenciamento de Identidade do Indy adota uma abordagem em que informações explicitamente públicas são armazenadas na blockchain, enquanto credenciais contendo informações privadas são mantidas fora de domínios públicos. Essa segregação é crucial para cumprir leis de proteção de dados, como o GDPR (*General Data Protection Regulation*) e LGPD (Lei Geral de Proteção de Dados Pessoais), garantindo que os requisitos de privacidade sejam atendidos durante todo o ciclo de vida das credenciais.

2.4.2 Hyperledger Aries

O Hyperledger Aries é um kit de ferramentas projetado para iniciativas e soluções focadas na criação, transmissão, armazenamento e uso de credenciais verificáveis. Em seu núcleo, estão os protocolos que permitem a conectividade e interações ponto a ponto entre agentes controlados por diferentes entidades - pessoas, organizações e coisas - usando mensagens seguras para trocar informações (CURRAN; HOWARD, 2021b). Usando o canal de mensagens padronizado, as credenciais verificáveis podem ser trocadas com base em DIDs enraizadas em diferentes registros (baseados em Indy ou outra tecnologia) usando uma variedade de implementações de credenciais verificáveis (CURRAN; HOWARD, 2021a).

Além disso, o Hyperledger Aries oferece uma interface para registro na blockchain; bibliotecas para a implementação de carteiras criptográficas destinadas ao ar-

mazenamento seguro de segredos criptográficos e outras informações; e uma implementação de VCs compatíveis com o W3C, utilizando primitivas ZKPs.

Os agentes Aries se comunicam entre si por meio de um mecanismo de mensagem chamado DIDComm (*DID Communication*). O DIDComm usa DIDs (especificamente DIDs pareados, privados - geralmente) para permitir mensagens criptografadas de ponta a ponta seguras, assíncronas entre agentes, incluindo a opção de roteamento de mensagens por meio de uma configuração de agentes mediadores. Os DIDs do agente Aries usados para DIDComm geralmente usam o método *did:peer*, que usa DIDs que não são publicados em um *ledger* distribuído, mas que são compartilhados apenas em particular entre as partes que se comunicam, geralmente apenas dois agentes (TEAM ARIES, 2021) (CURRAN; HOWARD, 2021b).

Um aplicativo que atua como um agente Aries tem a tarefa de resolver um DID. Para realizar essa tarefa, o agente consulta o *Universal* ou *DID Resolver*, conforme abordado anteriormente. Para emitir uma credencial, o emissor precisa utilizar um esquema de credencial registrado no livro-razão, que contém detalhes sobre as informações que compõem uma credencial sob esse esquema. Cada vez que um emissor cria um esquema, o livro-razão gera uma resposta com um identificador exclusivo para esse esquema. Se o emissor desejar emitir credenciais sob o esquema mencionado anteriormente, ele deve enviar uma definição de credencial para o livro-razão, mencionando o ID do esquema (já fornecido anteriormente), e outros campos, como o tipo de método de assinatura, como lidar com a revogação e assim por diante (INDY, 2022). Atualmente, existe um padrão W3C que tenta unificar as propostas existentes e encontrar uma camada comum para identificação e verificação descentralizada (BRUNNER *et al.*, 2020).

2.4.3 Bibliotecas e ferramentas facilitadoras

A seguir, são apresentadas as principais bibliotecas e ferramentas que desempenham um papel crucial na implementação das tecnologias Hyperledger Indy e Aries, facilitando significativamente o desenvolvimento e a integração dessas soluções avançadas.

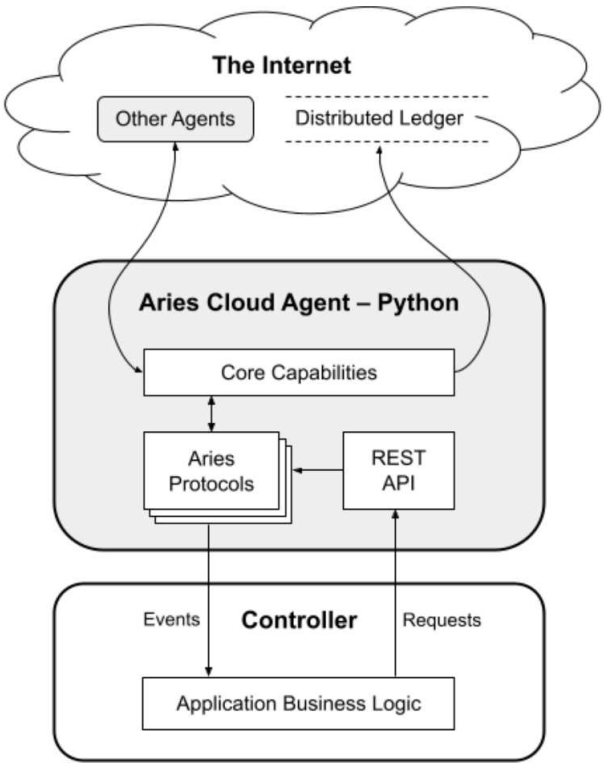
2.4.3.1 Hyperledger Aries Cloud Agent Python (ACA-Py)

O ACA-Py (*Hyperledger Aries Cloud Agent Python*) é uma biblioteca em Python projetada para a construção de ecossistemas de credenciais verificáveis e agentes. Ele opera na segunda e terceira camadas da estrutura *Trust Over IP* (IP FOUNDATION, 2023), utilizando mensagens DIDComm e protocolos Hyperledger Aries. Os protocolos Aries no ACA-Py formam a base para a construção de aplicações e serviços de identidade descentralizada. A “nuvem” no nome indica que o ACA-Py é executado em

servidores (nuvem, empresa, dispositivos IoT) e não foi projetado para ser executado em dispositivos móveis (HYPERLEDGER, 2023a) (JONG, 2020).

A Figura 9 ilustra a estrutura do ACA-Py, destacando uma instância do ACA-Py, um controlador e as interfaces entre o controlador e o ACA-Py, além dos caminhos externos para outros agentes e blockchains públicas na Internet.

Figura 9 – Protocolos Aries no ACA-Py da perspectiva do controlador.



Fonte: (HYPERLEDGER, 2023a).

Para utilizar o ACA-Py, cria-se um controlador de lógica de negócios que interage com o ACA-Py (enviando solicitações HTTP e recebendo notificações de *webhook*), e o ACA-Py lida com a funcionalidade Aries e DIDComm. Esse controlador pode ser construído em qualquer linguagem que suporte a realização e recebimento de solicitações HTTP. Os protocolos Aries suportados pelo ACA-Py incluem, de forma mais relevante, protocolos para emissão, verificação e retenção de credenciais verificáveis usando tanto o formato de credencial verificável Hyperledger AnonCreds (CURRAN *et al.*, 2022), quanto o formato de credencial verificável padrão W3C (SPORNY *et al.*, 2022b) usando JSON-LD (HYPERLEDGER, 2023a).

2.4.3.2 Von Network

A *Von Network* (*Verifiable Organizations Network*) representa uma versátil rede Indy Node para ambientes de desenvolvimento, equipada com um *Ledger Browser* que

capacita os usuários a acompanhar o status dos nós na rede, bem como a explorar, pesquisar e filtrar as transações da blockchain. É fundamental destacar que a Von Network não se destina a ser uma implementação de Indy Node para ambientes de produção. Ela foi concebida como uma rede temporária, designada exclusivamente para propósitos de desenvolvimento e teste (CURRAN; HOWARD, 2021a).

A Von Network por padrão é composta por quatro nós, desempenhando um papel fundamental como uma estrutura especializada no gerenciamento de identidade descentralizada e confiável. Ela disponibiliza recursos avançados para a publicação de DIDs públicos, a criação de esquemas e a definição de credenciais, tornando possível a verificação descentralizada de DIDs e credenciais verificáveis.

2.5 CONSIDERAÇÕES FINAIS

Neste trabalho, adotou-se um contexto voltado para a aplicação no campo da preservação de evidências forenses (Seção 2.1), especificamente com enfoque na cadeia de custódia de dados digitais provenientes de dispositivos IoT.

Em relação à abordagem da IoT, utilizou-se o modelo *Device-to-Gateway* (Subseção 2.2.1), onde o dispositivo de *gateway* é representado pelos dispositivos restritos. Foi adotado o protocolo de aplicação MQTT para a interação entre os dispositivos de recursos limitados (Subseção 2.2.2). Assume-se que o dispositivo IoT de recursos limitados esteja vinculado ao *broker* por meio de um certificado digital (Subseção 2.2.3) e propõe-se um *gateway* conectado ao *broker* atuando como agente SSI em nome dos dispositivos de maneira descentralizada.

As tecnologias e protocolos mencionados nas Subseções 2.3.1 e 2.3.2 foram adotados neste estudo. Para simplificar a implementação dessas tecnologias, utilizaram-se as bibliotecas ACA-Py e Indy SDK, juntamente com a *Von Network*, conforme apresentado na Seção 2.4 e suas respectivas subseções. Estas ferramentas foram essenciais na habilitação de funcionalidades como DIDs, VCs, ZKPs e agentes, complementadas pelo uso de blockchain para a infraestrutura de registro.

3 REVISÃO SISTEMÁTICA DE LITERATURA

Com o objetivo de coletar o máximo de informações sobre pesquisas realizadas e em andamento sobre o tema, uma revisão sistemática da literatura foi realizada, a fim de explorar a integração da identidade auto-soberana em ambientes IoT com o uso de tecnologias baseadas em DLT (*Distributed ledger technology*) adotando o conceito de identificadores decentralizados e credenciais verificáveis. O processo de revisão sistemática foi dividido em três etapas. A primeira etapa foi a de planejamento e construção do protocolo, a segunda etapa foi a de seleção e execução e, por fim, a terceira etapa sendo a sumarização e detalhamento dos resultados obtidos com a revisão. Desse modo, a Seção 3.1 aborda o processo utilizado para o mapeamento dos artigos relacionados. A Seção 3.2 apresenta a abordagem adotada para seleção e exclusão dos artigos. Na Seção 3.3 são apresentados os trabalhos que foram extraídos por meio do mapeamento.

3.1 PLANEJAMENTO DE CONSTRUÇÃO DO PROTOCOLO

3.1.1 Justificativa da necessidade

A abordagem da identidade auto-soberana oferece aos indivíduos controle total sobre seus próprios dados pessoais, permitindo-lhes compartilhar apenas as informações necessárias em transações específicas. Com o crescente número de dispositivos independentes na Internet das Coisas, surge a necessidade de adotar o paradigma de SSI para garantir a preservação da privacidade e segurança dos dados. De acordo com os padrões fornecidos pelo W3C (SPORNY *et al.*, 2022a) (SPORNY *et al.*, 2022b), usando identificadores descentralizados, é possível estabelecer uma estrutura descentralizada para a identificação única e persistente dos dispositivos IoT. Nesse contexto, as credenciais verificáveis desempenham um papel importante ao oferecerem uma maneira confiável de validar e comprovar as informações fornecidas por esses dispositivos.

No entanto, embora haja um grande potencial nessas abordagens, ainda existem lacunas significativas em relação à compreensão de como aplicar efetivamente a SSI em ambientes de IoT. É nesse ponto que a revisão sistemática da literatura desempenha um papel importante, permitindo uma análise abrangente e aprofundada das pesquisas existentes sobre o assunto. Essa abordagem é fundamental para entender o estado atual do conhecimento, identificar lacunas e desafios não resolvidos, e estabelecer uma base sólida para futuras pesquisas e desenvolvimentos. Além disso, uma revisão sistemática permite avaliar criticamente as abordagens propostas, identificar os benefícios e as limitações de cada uma delas, e fornece diretrizes práticas para implementações bem-sucedidas de SSI em dispositivos IoT. Dessa forma, a revisão

sistemática da literatura se mostra indispensável para o avanço do conhecimento na área em estudo.

3.1.2 Objetivo e questões de pesquisa

A realização da revisão sistemática tem como objetivo identificar e classificar o estado da arte dos estudos e modelos que exploram a abordagem da utilização dos princípios de SSI para dispositivos IoT. Com base nessa perspectiva, foram formuladas as seguintes Questões de Pesquisa (QPE):

- QPE1: Quais são os casos de uso práticos que demonstram a aplicação bem-sucedida de SSI em cenários de dispositivos IoT?
- QPE2: Quais são os padrões, protocolos ou tecnologias relevantes que estão sendo utilizados para implementar SSI em dispositivos IoT?
- QPE3: Quais são os desafios e obstáculos específicos relacionados à aplicação de SSI em dispositivos IoT?
- QPE4: Como as soluções de SSI para dispositivos IoT lidam com questões de escalabilidade, eficiência e consumo de recursos?
- QPE5: Quais as avaliações realizadas?

3.1.3 Fontes de pesquisas

Alguns critérios foram adotados para definição das fontes de pesquisas, ou seja, base de artigos:

- Relação com o tema: Somente estudos que se concentrem na implementação da identidade auto-soberana em dispositivos IoT, empregando padrões amplamente reconhecidos como DIDs e VCs, juntamente com tecnologias baseadas em registros distribuídos.
- Cobertura: De acordo com a quantidade de *proceedings* de conferências, *journals*, livros e área abrangida.
- Idioma: O conteúdo deve estar disponível nos idiomas inglês e português.
- Disponibilidade: O conteúdo integral deve estar disponível internamente na UFSC.

A busca foi realizada nas 5 principais base de dados relacionada a área de computação e são apresentadas no Quadro 1.

3.1.4 String de busca

Para conduzir uma revisão sistemática abrangente, uma *String* de busca foi definida com base em dois termos-chave principais: "*Internet of Things*" (IoT) e "*Self-*

Quadro 1 – Bases de dados.

Base de Dados	URL.
ACM	https://dl.acm.org/
IEEE	https://ieeexplore.ieee.org/Xplore/home.jsp
ScienceDirect	https://www.sciencedirect.com/
Scopus	https://www.scopus.com/
Springer	https://link.springer.com/

Fonte: elaborada pelo autor (2023).

Sovereign Identity" (SSI). Além disso, foram incluídos os termos relacionados *"decentralized identifiers*" (DIDs) e as tecnologias "blockchain" ou "*Distributed Ledger Technology*" (DLT). Essa estratégia de busca foi desenvolvida para recuperar estudos pertinentes que explorem a intersecção entre IoT e SSI, especialmente centrando-se nos padrões definidos pela W3C e nas tecnologias de registro distribuído, como blockchain ou DLT de forma abrangente. Assim, resultou na formulação da seguinte string de busca:

- *("Internet of Things") AND ("Self-Sovereign Identity" AND "decentralized identifiers") AND ("Blockchain" OR "Distributed ledger technology")*

3.1.5 Critérios de inclusão e exclusão

Para o processo de seleção dos trabalhos, foram estabelecidos os seguintes critérios de inclusão:

- Artigos de conferências ou periódicos.
- Publicações nos últimos 5 anos.
- Apresentam um método ou abordagem baseados nos princípios de identidade auto-soberana para redes ou dispositivos IoT, ou fazem uso de padrões ou protocolos para SSI estabelecidos pelo W3C, como DIDs e VCs.

Adicionalmente, com o objetivo de selecionar estudos de relevância, foram aplicados os seguintes critérios de exclusão:

- Estudos totalmente irrelevantes que foram recuperados devido a uma má execução da sequência de busca na base de pesquisa.
- Estudos não baseados em pesquisa (apresentados na forma de tutorial, demonstração ou baseado em opinião de especialistas).
- Estudos em formato de *whitepaper*.
- Estudos escritos em idiomas diferentes do português e inglês.
- Artigos não disponíveis internamente na UFSC.
- Estudos que adotam a forma de revisões/mapeamentos de artigos (*Surveys*).

- Estudos que não constituem um artigo completo.
- Estudos que utilizam alguma tecnologia blockchain para abordar a descentralização em IoT, mas não abordam especificamente os padrões ou protocolos relacionados a SSI.
- Estudos que abordam a temática de identidade auto-soberana, mas não discutem sua relação com IoT.
- Estudos que empregam redes IoT e SSI para um propósito específico, mas não exploram a integração entre ambas as tecnologias.
- Teses de doutorado ou dissertações de mestrado também foram excluídas porque publicações relevantes resultantes das pesquisas abordadas deveriam ter sido publicadas em periódicos ou conferências e deveriam ter sido recuperadas e incluídas neste estudo.

3.1.6 Avaliação da qualidade

Foram priorizados os artigos que abordaram aspectos técnicos significativos, ou seja, aqueles que introduzam métodos e técnicas relacionadas à identidade auto-soberana, ou que apliquem os padrões estabelecidos pelo W3C, como DIDs e VCs, em dispositivos IoT com a utilização de tecnologia blockchain. Portanto, foram definidas 5 questões para avaliar a qualidade dos trabalhos, essas questões são as seguintes:

1. Existe uma apresentação clara dos objetivos da pesquisa?
2. Os métodos ou técnicas foram apresentados de forma clara?
3. Os trabalhos foram apresentados com detalhes?
4. O artigo propõe alguma implementação (ou se restringe a explicações e sugestões de casos de uso)?
5. O artigo propõe alguma avaliação?

Os critérios utilizados foram adaptados com base no trabalho de (KITCHENHAM; CHARTERS, 2007) para avaliar a qualidade do estudo. Cada critério receberá uma pontuação na escala de 0 a 1, de acordo com a seguinte correspondência:

- Não (pontos=0)
- Parcialmente não (pontos=0.33)
- Parcialmente sim (pontos=0.67)
- Sim (pontos=1)

Com base nas pontuações obtidas para cada critério, os trabalhos serão classificados da seguinte forma:

- Índices entre 0 e 1.98: Considerados "Ruins"

- Índices entre 2 e 3.35: Considerados "Regulares"
- Índices entre 3.4 e 4.25: Considerados "Bons"
- Acima de 4.3: Considerados "Excelentes"

Essa abordagem de avaliação proporcionará uma medida global da qualidade do estudo, permitindo classificar os trabalhos em categorias que refletem seu nível de aderência aos critérios estabelecidos.

3.1.7 Procedimento de seleção

Foi empregado um processo de condução da revisão e aplicação dos critérios de exclusão/inclusão, o qual é delineado a seguir:

1. Aplicar a *String* de busca detalhada na Subseção 3.1.4 e incorporar os artigos das bases selecionadas.
2. Eliminar os artigos que não foram publicados em conferências ou periódicos.
3. Excluir os estudos que não estão acessíveis internamente na UFSC.
4. Excluir os estudos que não estão disponíveis nos idiomas português e inglês.
5. Descartar os estudos com base nos títulos e palavras-chave, seguindo os critérios de exclusão definidos na Subseção 3.1.5.
6. Eliminar os estudos com base nos resumos, levando em conta os critérios de exclusão descritos na Subseção 3.1.5.
7. Descartar os estudos com base nas introduções e conclusões, de acordo com os critérios de exclusão estabelecidos na Subseção 3.1.5.
8. Eliminar os estudos após a leitura completa, seguindo os critérios de exclusão detalhados na Subseção 3.1.5.
9. Realizar a extração dos dados de cada artigo selecionado, conforme delineado na Subseção 3.1.6.

3.2 SELEÇÃO E EXECUÇÃO

A busca foi restrita a estudos abertos à rede da UFSC, sem uma data limite específica, devido ao conceito recente de SSI para IoT. O resultado da busca em cinco bases foi de 714 artigos. Após uma análise inicial, identificaram-se 66 artigos duplicados, resultando em um total de 648 artigos. Em seguida, realizou-se a leitura do resumo (*abstract*) desses artigos, resultando na seleção de 91 candidatos para o estudo. Após uma análise mais aprofundada, constatou-se que muitos artigos não estavam disponíveis na íntegra e a maioria não estava relacionada ao estudo em questão, resultando na rejeição de 53 artigos. Por fim, foram selecionados 38 artigos

para leitura completa. Adicionalmente, com a aplicação da técnica de *snowballing*, foram incluídos mais 2 artigos ao conjunto de estudos selecionados.

3.3 SUMARIZAÇÃO E ESCRITA DOS RESULTADOS DO MAPEAMENTO

Dos 40 artigos que foram selecionados para leitura completa, após uma análise minuciosa, foi possível chegar a uma seleção final de 13 estudos que apresentaram relevância com o tema em questão, ou seja, os estudos analisados têm como objetivo propor modelos de aplicação da SSI para dispositivos IoT, adotando padrões e protocolos estabelecidos na área.

Em grande parte desses estudos, são utilizadas tecnologias baseadas em DLT como meio de garantir a segurança, eliminação de intermediários e resiliência a violações de dados. A seguir, é realizada uma comparação abrangente entre os trabalhos selecionados na Subseção 3.3.1, seguida pela abordagem das questões de pesquisa na Subseção 3.3.2 e uma discussão sobre as oportunidades de avanço na Subseção 3.3.3.

3.3.1 Comparação geral entre os trabalhos

Há uma escassez de literatura sobre o conceito de *Self-Sovereign Identity* para ecossistemas IoT. Para abordar essa lacuna, realizou-se uma comparação abrangente dos trabalhos resultantes do mapeamento sistemático, abrangendo o uso da tecnologia blockchain, identificadores descentralizados, credenciais verificáveis, mecanismos de consenso e os diferentes domínios de aplicação abordados em cada artigo.

3.3.2 Respostas as questões de pesquisa

Nesta seção, serão apresentadas as respostas às questões de pesquisa definidas na Subseção 3.1.2, seguidas de uma breve discussão individual.

3.3.2.1 QPE1: Quais são os domínios de aplicação da SSI para IoT que os trabalhos analisados abordaram?

Entre os estudos analisados, dois deles concentram-se em propor um esquema de identidade direcionado ao setor automotivo, conhecido como Internet das Coisas Veiculares IoV (*Internet of Vehicles*). (TERZI *et al.*, 2020) propõe uma estrutura para verificar a autenticidade dos valores de emissão de veículos, a fim de prevenir adulterações. Por sua vez, (THEODOULI *et al.*, 2020) apresenta uma arquitetura descentralizada que garante atualizações seguras de software em dispositivos embarcados de veículos.

Outros dois trabalhos estão voltados para um esquema de identidade destinado a dispositivos IoT no contexto industrial, conhecido como Internet das Coisas Industrial

IIoT (*Industrial Internet of Things*). (DIXIT; CREASEY; RAJARAJAN, 2022) propõe uma comunicação M2M (*Machine to Machine*) entre dispositivos, permitindo acesso a um Registro de Dados Verificáveis VDR (*Verifiable Data Registry*) por meio de DIDs. O estudo utiliza duas plataformas blockchain diferentes para fins comparativos de desempenho. Em contrapartida, (REGUEIRO *et al.*, 2022) analisa os participantes e as funções em um ambiente de identidade industrial, apresentando um protocolo que define como os dados devem ser trocados por meio de uma rede distribuída baseada em tecnologia blockchain.

(FOTOPOULOS *et al.*, 2020) apresenta uma arquitetura de identidade para dispositivos médicos IoMT (*Internet of Medical Things*) que incorpora um agente para interagir com a blockchain, permitindo o registro e a transmissão dos dados de saúde do paciente. Em um estudo conduzido por (GEBRESILASSIE *et al.*, 2020), os autores abordam de forma breve uma solução de SSI para IoT baseada em DLT pública para a indústria de aluguel de carros. (DIEGO; REGUEIRO; FERNANDEZ, 2021) propõe a utilização de SSI para o modelo de negócios *IoT-as-a-Service*, que envolve a execução de um agente interativo diretamente no dispositivo IoT para transações com a blockchain e o armazenamento de credenciais verificáveis.

(FEDRECHESKI *et al.*, 2020) introduz um mecanismo autossuficiente de identificação e comunicação para agentes IoT em redes restritas, visando minimizar a sobrecarga. Os autores apresentam uma extensão para DIDs, junto com um método mais conciso de serialização de metadados de documentos DID. Além disso, para excesso de segurança nas mensagens transmitidas, propõem o uso de um envelope de mensagem binária, mas não abordam um contexto de aplicação.

(LUECKING *et al.*, 2020) propõe um esquema de identidade auto-soberana para IoT em geral. (YIN *et al.*, 2022) apresenta o protótipo de um esquema de SSI baseado em *Bulletproofs*, um sistema de prova de declarações gerais, e blockchain Fisco Bcos. (SHARMA, P.; GODFREY; TRIVEDI, 2022) realiza uma comparação quantitativa do desempenho de protocolos de comunicação amplamente utilizados na IoT, como CoAP, MQTT e DIDComm (*Decentralized Identifier Communication*). O estudo também propõe uma arquitetura para a implementação de identidade auto-soberana em dispositivos IoT. No entanto, esses quatro últimos estudos não abordam nenhum contexto ou caso de uso específico para essa implementação.

Finalmente, (KORTESNIEMI *et al.*, 2019) propõem o uso de DIDs como identificadores para dispositivos IoT e realizam um exame preciso dos requisitos para dispositivos IoT implementarem um sistema de gerenciamento de identidade baseado em SSI. Outros autores, (BERZIN *et al.*, 2021), propõem abordagens para estabelecer a conexão entre dispositivos IoT e usuários e propor o gerenciamento de identidade desses dispositivos com DIDs e VCs.

3.3.2.2 QPE2: Quais são os padrões, protocolos e tecnologias abordados nos estudos para implementar SSI em dispositivos IoT?

Todos os estudos analisados propõem o uso dos padrões e protocolos estabelecidos pela W3C para DIDs e VCs. Em relação às tecnologias, dos estudos analisados, cinco deles fazem uso da tecnologia blockchain permissionada Hyperledger Indy (FOTOPOULOS *et al.*, 2020); (DIEGO; REGUEIRO; FERNANDEZ, 2021); (REGUEIRO *et al.*, 2022); (SHARMA, P.; GODFREY; TRIVEDI, 2022); (THEODOULI *et al.*, 2020).

Alguns estudos exploram a utilização conjunta de duas blockchains. (DIXIT; CREASEY; RAJARAJAN, 2022) têm como objetivo integrar a blockchain pública Ethereum com a blockchain Hyperledger Indy. (TERZI *et al.*, 2020) explora a integração das plataformas Hyperledger Fabric, por meio de contratos inteligentes, com o Hyperledger Indy. Outros dois estudos propõem a utilização do DLT IOTA Tangle, uma tecnologia que tem suas raízes no Ethereum, como modelo para implementação de SSI em dispositivos IoT (GEBRESILASSIE *et al.*, 2020); (LUECKING *et al.*, 2020). Além disso, um estudo explora o uso da blockchain portátil FISCO BCOS (YIN *et al.*, 2022).

Os estudos de (FEDRECHESKI *et al.*, 2021), (BERZIN *et al.*, 2021) e (KORTESNIEMI *et al.*, 2019) não mencionam especificamente o uso de blockchain ou tecnologias baseadas em livro-razão em suas propostas.

Tabela 1 – Comparação entre tecnologias propostas e recursos.

Autores	Ledger	DIDs	VCs	ZKP
(TERZI *et al.*, 2020)	Hyperledger Fabric and Indy	✓	✓	✓
(DIXIT; CREASEY; RAJARAJAN, 2022)	Ethereum and Hyperledger Indy	✓	✓	-
(GEBRESILASSIE *et al.*, 2020)	DLT IOTA Tangle	✓	✓	-
(LUECKING *et al.*, 2020)	DLT IOTA Tangle	✓	-	-
(FEDRECHESKI *et al.*, 2021)	-	✓	-	-
(YIN *et al.*, 2022)	FISCO BCOS	✓	✓	-
(THEODOULI *et al.*, 2020)	Hyperledger Indy	✓	✓	-
(FOTOPOULOS *et al.*, 2020)	Hyperledger Indy	✓	✓	-
(REGUEIRO *et al.*, 2022)	Hyperledger Indy	✓	✓	-
(SHARMA, P.; GODFREY; TRIVEDI, 2022)	Hyperledger Indy	✓	✓	-
(DIEGO; REGUEIRO; FERNANDEZ, 2021)	Hyperledger Indy	✓	✓	✓
(KORTESNIEMI *et al.*, 2019)	-	✓	✓	-
(BERZIN *et al.*, 2021)	-	✓	✓	-
Este trabalho	Hyperledger Indy	✓	✓	✓

A Tabela 1 destaca as tecnologias, padrões e protocolos propostas pelos estudos. Em cada trabalho, é descrita a tecnologia de registro distribuído utilizada, padrões como DIDs e VCs e como protocolo o suporte ZKP.

3.3.2.3 QPE3: Quais são os desafios e obstáculos na aplicação das tecnologias SSI em dispositivos IoT, e quais estratégias são propostas para superá-los?

Alguns estudos destacam que a escalabilidade é uma dificuldade significativa, especialmente para a adoção em larga escala de SSI em dispositivos IoT (YIN *et al.*, 2022), (DIEGO; REGUEIRO; FERNANDEZ, 2021), (LUECKING *et al.*, 2020), (THEODOULI *et al.*, 2020), (GEBRESILASSIE *et al.*, 2020) e (DIXIT; CREASEY; RAJARAJAN, 2022). Alguns autores abordam a necessidade de aprimorar a privacidade dos dados (YIN *et al.*, 2022), (REGUEIRO *et al.*, 2022), (LUECKING *et al.*, 2020) e (GEBRESILASSIE *et al.*, 2020).

(YIN *et al.*, 2022) destaca a dificuldade de integrar SSI com sistemas legados existentes. (GEBRESILASSIE *et al.*, 2020) e (DIXIT; CREASEY; RAJARAJAN, 2022) enfatizam a importância do desenvolvimento de padrões e protocolos comuns para garantir a interoperabilidade entre diferentes plataformas blockchain. (DIXIT; CREASEY; RAJARAJAN, 2022) menciona os custos envolvidos na implementação de SSI em dispositivos IoT como um desafio. (KORTESNIEMI *et al.*, 2019) destaca que existem riscos de segurança ao utilizar DIDs diretamente em dispositivos para se conectar diretamente à Internet.

Esses desafios e obstáculos destacam a complexidade da aplicação de SSI em dispositivos IoT e a necessidade de pesquisas e desenvolvimentos contínuos para superá-los.

3.3.2.4 QPE4: Como os estudos abordam questões de segurança, escalabilidade, eficiência e consumo de recursos na aplicação da SSI na IoT?

Em termos de escalabilidade, eficiência e uso de recursos, a abordagem proposta por (TERZI *et al.*, 2020) emprega uma arquitetura híbrida de blockchain, utilizando o Hyperledger Fabric (acesso privado) e o Hyperledger Indy (acesso público), ambos permissionados. A maioria dos estudos, incluindo (THEODOULI *et al.*, 2020), (FOTOPOULOS *et al.*, 2020), (REGUEIRO *et al.*, 2022), (SHARMA, P.; GODFREY; TRIVEDI, 2022) e (DIEGO; REGUEIRO; FERNANDEZ, 2021), também adota o Indy. Isso possibilita um processamento de transações mais ágil e eficiente quando comparado a blockchains não permissionados.

Além disso, essas soluções baseadas nessas blockchains eliminam a necessidade de incentivos, resultando em aumento de eficiência e redução do consumo de recursos. Quanto à segurança, apesar da natureza pública do Indy, nenhum dado privado é armazenado na blockchain. Isso proporciona a minimização de dados, reduzindo assim a quantidade de informações brutas que precisam ser transmitidas e processadas.

(DIXIT; CREASEY; RAJARAJAN, 2022) adota tanto o Indy quanto o Ethereum

em sua abordagem para comparação de desempenho, sendo este último um blockchain público e não permissionado. O estudo destaca os pontos fortes e fracos de ambas as tecnologias; no entanto, é evidente que o Indy se destaca devido à sua notável eficiência, uma vez que não impõe custos, resultando em desempenho superior.

Embora o trabalho de (FEDRECHESKI *et al.*, 2021) busque otimizar a eficiência e reduzir o consumo de recursos ao propor um mecanismo para diminuir o tamanho e a sobrecarga dos metadados no *diddocument*, o enfoque não abrange aspectos relacionados à blockchain. (YIN *et al.*, 2022) conduz uma análise detalhada da segurança de sua proposta e explora questões de escalabilidade, eficiência e consumo de recursos. Embora não forneça muitos detalhes sobre a blockchain, cita o mecanismo de consenso e a ausência da necessidade de incentivos.

(LUECKING *et al.*, 2020) e (GEBRESILASSIE *et al.*, 2020) apresentam uma solução baseada na tecnologia IOTA para seu blockchain distribuído. Eles argumentam que o IOTA, que não é permissionado, não possui custos de transação e é altamente escalável. No entanto, é importante notar que a IOTA é baseada em blockchain público e não é totalmente descentralizada, pois ainda depende de um componente centralizador do processo de consenso e pode sofrer com ataques de controle majoritário (34%) e ataques de gasto duplo.

(KORTESNIEMI *et al.*, 2019) e (BERZIN *et al.*, 2021) propõem abordagens baseadas em *proxy* para questões de segurança e eficiência no consumo de recursos. (BERZIN *et al.*, 2021) cita a utilização de plataforma em nuvem para tornar a proposta escalável.

3.3.2.5 QPE5: Quais são as metodologias e critérios de avaliação utilizados pelos estudos para analisar a eficácia e eficiência das soluções de SSI em dispositivos IoT?

(DIXIT; CREASEY; RAJARAJAN, 2022) realiza análises abrangentes que englobam a avaliação do desempenho do sistema, a escalabilidade e a segurança da infraestrutura proposta para a identidade digital descentralizada. Por outro lado, (GEBRESILASSIE *et al.*, 2020) propõe uma nova solução para o gerenciamento de identidade dos dispositivos IoT, mas não apresenta análises detalhadas em seu estudo. Em contraste, (LUECKING *et al.*, 2020) oferece uma revisão completa de vários *frameworks* existentes de identidade e confiança no domínio IoT, seguida pela introdução de detalhes de design e implementação do novo *framework*. No entanto, o estudo não fornece informações específicas sobre os testes ou análises realizadas. Quanto a (TERZI *et al.*, 2020), eles apresentam uma proposta para o uso de blockchain e identidades auto-soberanas na garantia de dados de emissão em redes de veículos inteligentes, mas não incluem uma avaliação específica dessa abordagem.

No estudo de (FEDRECHESKI *et al.*, 2021), são apresentadas avaliações, como

a redução do tamanho dos metadados de identidade em quase quatro vezes e a redução da sobrecarga de segurança em até cinco vezes. Em relação a (FOTOPOULOS *et al.*, 2020), os autores apresentam uma nova abordagem para autenticar dispositivos médicos na área de IoMT, no entanto, não especificam se o estudo incluiu alguma avaliação ou análise específica para validar sua proposta. No estudo de (DIEGO; REGUEIRO; FERNANDEZ, 2021), é proposto um sistema de gerenciamento de identidade baseado em SSI compatível com os padrões existentes da W3C. Além disso, o estudo inclui uma avaliação de desempenho utilizando Raspberry, considerando o tempo necessário para criação de credencial, troca de credenciais e verificação de credenciais. Por sua vez, (REGUEIRO *et al.*, 2022) não aborda os resultados em seu estudo, deixando de fornecer informações específicas sobre as análises ou conclusões alcançadas.

(YIN *et al.*, 2022) oferece uma discussão detalhada sobre a implementação de identidade distribuída através de DIDs nas plataformas blockchain Ethereum e Hyperledger Fabric, além de realizar uma avaliação de seu desempenho em relação à taxa de transferência de transações, latência e consumo de recursos. No estudo de (SHARMA, P.; GODFREY; TRIVEDI, 2022), foram realizadas comparações quantitativas entre quatro protocolos de comunicação IoT em um *framework* baseado em blockchain para identificadores descentralizados entre dispositivos IoT e servidores em uma rede sem fio. Os protocolos foram avaliados utilizando métricas como latência, taxa de transferência e sobrecarga de rede.

(KORTESNIEMI *et al.*, 2019) analisa a capacidade dos dispositivos IoT lidarem com execuções criptográficas, porém não chega a implementar nenhum protótipo, se resumindo apenas a analisar a possibilidade da aplicação de DIDs para identificação e autenticação de dispositivos restritos. (BERZIN *et al.*, 2021) se limita a analisar a latência entre um *datacenter* e um dispositivo de borda.

3.3.3 Lacunas e contribuições dos trabalhos

Primeiramente, alguns autores, como (TERZI *et al.*, 2020), introduziram um *framework* para verificar a autenticidade dos valores de emissão de veículos por meio de um sistema de autenticação e autorização descentralizado, utilizando tecnologias de Blockchain (Hyperledger Fabric e Hyperledger Indy). No entanto, há considerações a serem levadas em conta: a presença de entidades centrais, como Autoridades de Registro (RAs), levanta questionamentos sobre o grau real de descentralização e controle do ecossistema. (THEODOULI *et al.*, 2020) apresentaram um modelo para atualizações seguras de software no ecossistema da IoV por meio de uma arquitetura totalmente descentralizada usando o Hyperledger Indy. No entanto, o estudo carece de detalhes de implementação técnica e não fornece métricas sobre problemas de desempenho e escalabilidade em cenários de grande escala.

(FOTOPOULOS *et al.*, 2020) realizaram um estudo sobre mecanismos de autenticação para dispositivos médicos; no entanto, sua proposta aborda brevemente questões cruciais de segurança, eficiência e escalabilidade, e não considera a conectividade com dispositivos limitados.

Alguns autores, como (GEBRESILASSIE *et al.*, 2020), introduziram um esquema SSI baseado em IOTA para sua infraestrutura de registro distribuído e o aplicaram a um caso de uso de aluguel de carros. De forma semelhante, outros pesquisadores, como (LUECKING *et al.*, 2020), também sugeriram o IOTA como uma opção viável de DLT para a implementação do esquema SSI. Eles enfatizam sua natureza sem permissões, ausência de custos de transação e vantagens de escalabilidade. No entanto, a questão principal com essa proposta é que o IOTA depende de um coordenador central no processo de consenso, tornando-o ainda não totalmente descentralizado.

(DIXIT; CREASEY; RAJARAJAN, 2022) propuseram um quadro de identidade digital para dispositivos industriais de IoT usando uma combinação de Hyperledger Indy e Ethereum. No entanto, sua proposta não aborda de maneira abrangente preocupações com escalabilidade, visto que a plataforma Ethereum implica em custos que podem representar desafios significativos no enfrentamento do aumento da demanda e das altas taxas operacionais em ambientes com alto consumo de recursos. (FEDRECHESKI *et al.*, 2021) buscaram reduzir o *overhead* de documentos DID para redes restritas, embora evitem entrar em detalhes sobre as tecnologias de blockchain utilizadas.

(DIEGO; REGUEIRO; FERNANDEZ, 2021) introduziram um sistema de Gerenciamento de Identidade (IdM) baseado em SSI para o modelo de negócios *IoT-as-a-Service*. (REGUEIRO *et al.*, 2022) propuseram um protocolo baseado em SSI no contexto da Internet Industrial das Coisas (IIoT). Ambas as propostas utilizam o Hyperledger Indy, no entanto, elas não abordam dispositivos limitados, e apenas (DIEGO; REGUEIRO; FERNANDEZ, 2021) conduzem avaliações de desempenho. (SHARMA, C.; GONDHI, 2018) realizaram uma comparação de desempenho de protocolos de comunicação de IoT para SSI, mas não consideraram restrições de dispositivos, integridade de dados e rastreabilidade.

(KORTESNIEMI *et al.*, 2019) realiza uma análise profunda da capacidade dos dispositivos IoT de lidar com execuções criptográficas. Eles também verificam a capacidade da CPU de dispositivos restritos de lidar com essas operações, além de discutir a questão de energia e fontes de entropia, e questões de segurança e privacidade. No entanto, eles não chegam a implementar nenhum protótipo, limitando-se apenas a analisar a possibilidade da aplicação de DIDs para identificação e autenticação de dispositivos restritos, deixando credenciais de fora do estudo.

(BERZIN *et al.*, 2021), por outro lado, não fornece uma especificação concreta para a implementação de VCs para IoT. Eles se concentram principalmente na análise

de DIDs como identificadores adequados para este ambiente específico e não realizam análises aprofundadas nem consideram testes de desempenho.

Além disso, algumas arquiteturas também empregam blockchains com algoritmos de consenso, como o *Proof-of-Work*, que são computacionalmente custosos e suscetíveis a ataques conhecidos de 51%. Outro ponto a considerar é a falta de métricas específicas sobre os benefícios reais de desempenho e escalabilidade em cenários de grande escala. Por fim, embora um dos estudos tenha abordado dispositivos restritos, não foi proposto nenhum protótipo ou demonstração prática.

A Tabela 2 apresenta uma comparação entre esta proposta e diversos trabalhos relacionados no campo de SSI aplicado à IoT. Cada linha representa um estudo diferente, identificando os autores, o domínio de aplicação do trabalho e o problema abordado. Os primeiros quatro critérios servem como indicadores-chave para analisar a adequação da solução em relação a dispositivos IoT limitados e integridade de dados. Também considera escalabilidade e avaliação de desempenho.

Tabela 2 – Comparação entre trabalhos relacionados e este estudo.

Autores	**Domínio de aplicação**	**Questão abordada**	**Dispositivos restritos**	**Integridade de dados**	**Análise de Escalabilidade**	**Avaliação de Desempenho**
TERZI et al., 2020	IoT Vehicles	Data integrity	×	✓	-	×
THEODOULI et al., 2020	IoT Vehicles	Data security	×	✓	-	×
FOTOPOULOS et al., 2020	IoT Medical	Access control	×	✓	×	×
GEBRESILASSIE et al., 2020	IoT Platform	Secure transaction	×	×	✓	×
LUECKING et al., 2020	IoT Devices	Data accessing	-	×	✓	✓
DIXIT et al., 2022	IoT Industrial	Data sharing	×	×	×	✓
FEDRECHESKI et al., 2021	IoT Network	Communication	×	×	✓	✓
YIN et al., 2022	IoT Devices	Digital identity	×	×	✓	✓
DIEGO et al., 2021	IoT Services	Secure transaction	×	✓	×	✓
REGUEIRO et al., 2022	IoT Industrial	Data accessing	×	✓	×	×
SHARMA et al., 2022	IoT Network	Communication	×	×	✓	✓
Kortesniemi et al., 2019	IoT Services	Secure transaction	✓	×	×	×
Berzin et al., 2021	IoT Devices	Digital identity	×	✓	×	×
Este trabalho	IoT Data	Data integrity	✓	✓	✓	✓

É relevante notar que a maioria dos estudos analisados não esclarece os métodos específicos de conexão e identificação dos dispositivos IoT. Muitos propõem a criação de uma arquitetura de prova de conceito para aplicação de SSI nesse ecossistema. No entanto, há uma carência de pesquisas que avaliem o desempenho das tecnologias e padrões SSI para a IoT. Compreender como essa implementação pode influenciar a eficiência, latência e outros aspectos de desempenho dos dispositivos IoT e dos sistemas aos quais estão conectados é fundamental. Além disso, poucos

trabalhos abordam as particularidades dos dispositivos restritos, que não possuem capacidade suficiente para lidar com operações criptográficas.

3.4 CONSIDERAÇÕES FINAIS

Diante dessas lacunas na pesquisa existente, este trabalho tem como objetivo introduzir uma abordagem SSI voltada para dispositivos IoT. A proposta inclui o desenvolvimento de uma solução de gateway especialmente adaptada para dispositivos com recursos limitados. O foco principal é propor meios para identificar os dispositivos por meio de DIDs e garantir a confiabilidade, integridade e rastreabilidade dos dados emitidos por esses dispositivos através de VCs. Esse esforço será seguido por uma avaliação prática para medir o desempenho da solução proposta.

Acredita-se que essa abordagem contribuirá significativamente para o avanço da SSI em dispositivos IoT, oferecendo alternativas para a identificação e a cadeia de custódia de dados.

4 ABORDAGEM BASEADA EM GATEWAY PARA DISPOSITIVOS IOT RESTRITOS

Este capítulo introduz o contexto e a fundamentação da abordagem proposta, oferecendo uma visão geral do cenário que inspirou este estudo. Inicia-se com a apresentação da hipótese de pesquisa. Em seguida, aborda-se a aplicação no contexto de preservação de evidências forenses, demonstrando como a proposta deste trabalho pode contribuir para a cadeia de custódia dos dados digitais no âmbito da IoT forense. Na sequência, delineia-se o escopo da arquitetura, abordando suas camadas e funções. Procede-se com a exploração da proposta, enfatizando as tecnologias e ferramentas adotadas. Conclui-se com a discussão sobre as entidades envolvidas, esclarecendo seus papéis e interações.

4.1 HIPÓTESE GERAL DE PESQUISA

Considerando as lacunas de pesquisa identificadas e a potencial contribuição deste estudo, formula-se a seguinte hipótese: A aplicação das tecnologias de Identidade Auto-soberana, apoiadas por blockchain, em ecossistemas IoT com dispositivos de capacidade limitada, especialmente em contextos forenses, influenciará positivamente a segurança dos dispositivos e contribuirá para a integridade, autenticidade e rastreabilidade dos dados na cadeia de custódia digital.

4.2 CONTEXTO DE PRESERVAÇÃO DE EVIDÊNCIAS FORENSES

A cadeia de custódia de evidências digitais forenses é fundamental para a confiabilidade e justiça dos processos judiciais (ARSHAD; JANTAN; ABIODUN, 2018). O crescimento da IoT resulta em um número crescente de dispositivos interconectados, produzindo dados potencialmente cruciais para investigações criminais (STOYANOVA *et al.*, 2020). Entretanto, a volatilidade e complexidade dos dados IoT impõem desafios à preservação de sua autenticidade e integridade (SANDVIK; FRANKE; ÅRNES, 2020) desde a coleta até sua apresentação em juízo.

A motivação para este trabalho surge da necessidade de abordagens robustas que possam assegurar a veracidade e a imutabilidade dos dados digitais forenses. Nesse sentido, a proposta apresentada visa estabelecer um gateway de segurança da informação. Utilizando credenciais verificáveis e tecnologias descentralizadas, tal sistema não apenas certifica os dados na sua origem, mas também consolida a cadeia de custódia e mitiga o risco de repúdio. Com base nesse entendimento, descreve-se a seguir a arquitetura proposta em alto nível.

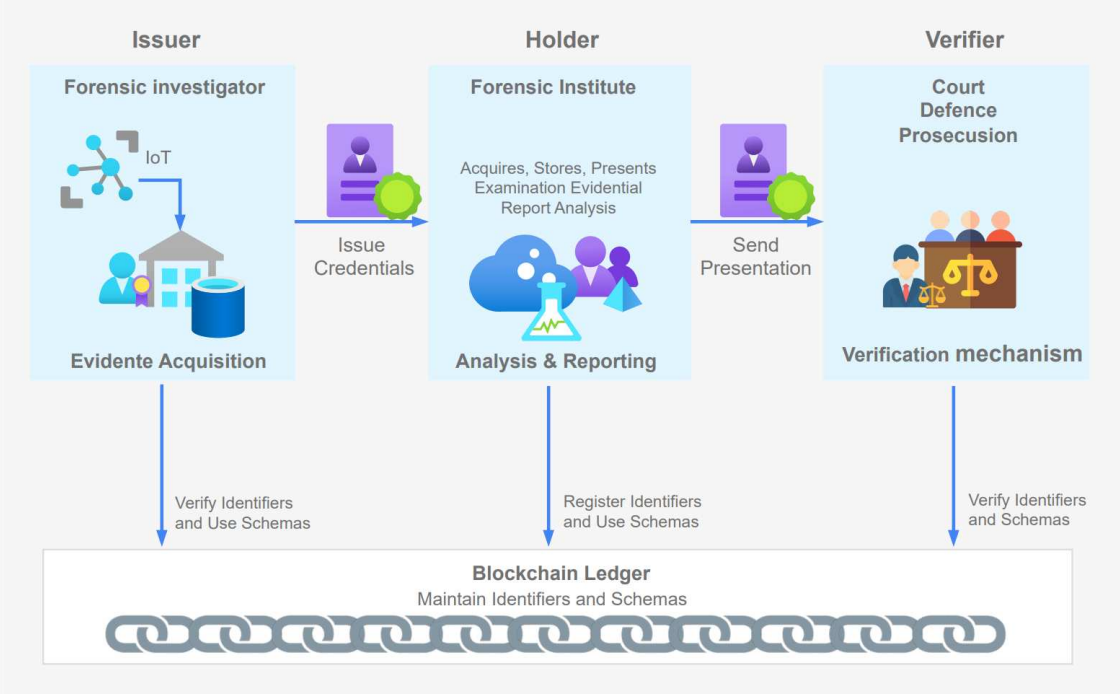
4.2.1 Descrição da arquitetura proposta

A proposta da arquitetura do Gateway visa melhorar a confiabilidade e a segurança dos dados coletados por dispositivos IoT. O processo começa com a validação e certificação desses dados, o que é fundamental para garantir sua autenticidade. Isso é feito por uma figura autorizada, conhecido como "Emissor" das credenciais, que pode ser um investigador ligado a órgãos estatais ou um perito nomeado judicialmente, conforme estabelecido pelo Código de Processo Civil (Lei nº 13.105/2015). Este Emissor adiciona uma camada inicial de confiança ao processo ao certificar os dados.

Em seguida, as credenciais geradas são emitidas para uma entidade responsável por sua gestão, como um instituto de criminalística, um laboratório criminal, ou outro órgão acreditado. Esta entidade, denominada "Detentor", assume a tarefa de analisar, armazenar e divulgar as informações certificadas. Ao fazer isso, o Detentor assegura a manutenção de um registro transparente e rastreável dos dados, fortalecendo a integridade e a confiabilidade do processo como um todo.

Na fase final, as partes interessadas, como promotores, advogados e juízes, assumem o papel de *Verifiers*, ou verificadores, onde a autenticidade e a integridade dos dados são examinadas, potencialmente utilizando protocolos ZKP para divulgação seletiva para manter a privacidade das informações sensíveis. Estas interações podem ser visualizadas na Figura 10.

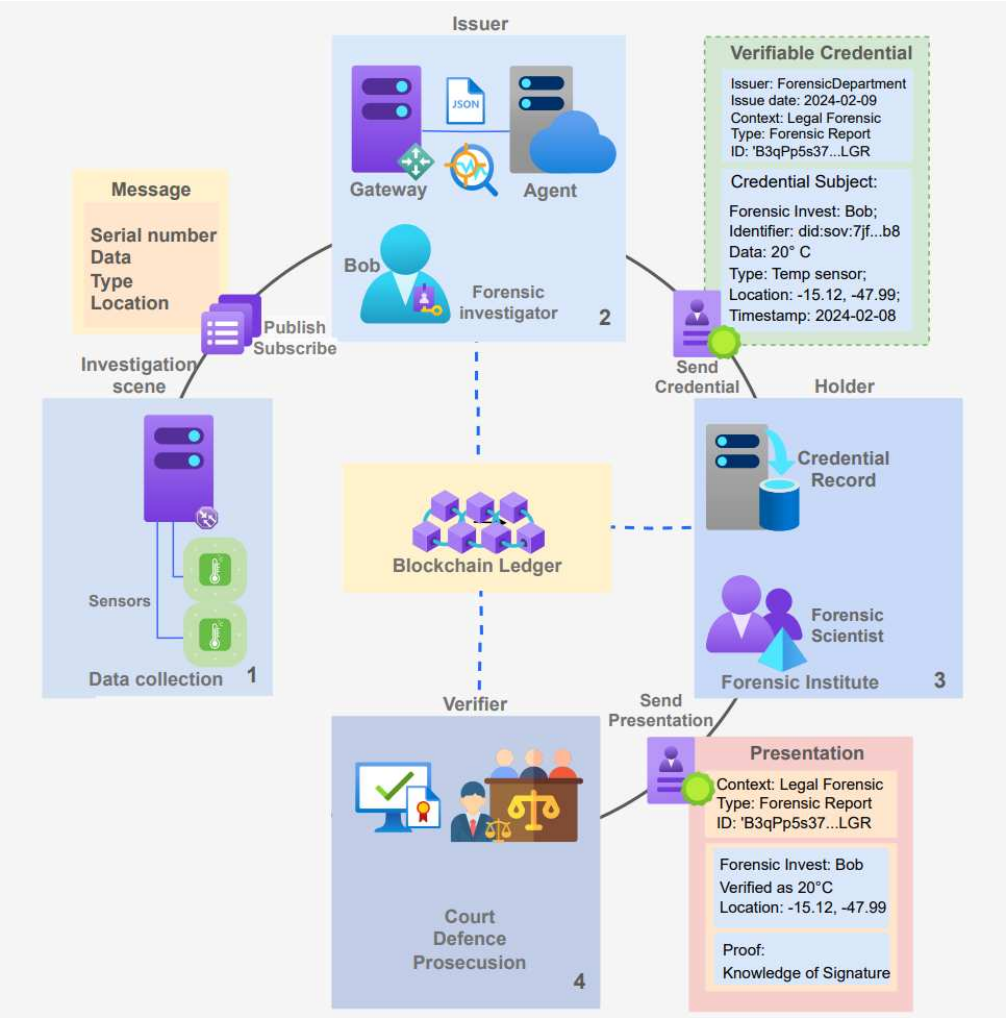
Figura 10 – Contexto para preservação de evidência forense.



Fonte: elaborada pelo autor (2023).

Este sistema fortalece a cadeia de custódia em ambientes digitais forenses ao integrar tecnologia e princípios legais. Ao proporcionar métodos robustos de preservação e autenticação de evidências digitais, aumenta-se a confiança na validade e origem das evidências apresentadas em processos judiciais. A Figura 11 ilustra o funcionamento da arquitetura proposta para a preservação de evidências forenses digitais em alto nível.

Figura 11 – Abordagem proposta em alto nível.



Fonte: elaborada pelo autor (2023).

1. Coleta de Dados na Cena da Investigação: Sensores IoT coletam dados na interação com o ambiente e publicam essas informações. Além disso, informações associadas são o número de série, tipo do dispositivo e localização na cena da investigação.
2. Emissão de Credenciais pelo Investigador Forense: Um investigador forense, identificado como Bob, utiliza o Gateway para processar os dados recebidos e os encapsula em uma credencial verificável utilizando recursos do Agente,

uma aplicação ou serviço em nuvem, com capacidades robustas e escaláveis para lidar com requisições de alta demanda de operações de emissão de credenciais verificáveis e operações com a blockchain. A credencial é emitida e sua integridade e autenticidade podem ser verificadas através da blockchain.

3. Armazenamento pelo Instituto Forense: O Instituto Forense atua como o Holder (detentor) da credencial verificável. Um cientista forense dentro do Instituto é responsável por receber, armazenar e manter o registro da credencial, que contém todos os detalhes relevantes da evidência coletada, como tipo de sensor, dados específicos coletados e carimbo de data/hora.
4. Verificação pelas Autoridades Judiciais: Finalmente, a credencial é enviada para a verificação por entidades judiciais, tribunais de justiça, promotorias e defensorias. O processo de verificação inclui a apresentação da credencial verificável que contém a prova coletada, juntamente com a confirmação de que o investigador Bob foi responsável durante esse processo. Além disso, para outras partes pode ser fornecida apresentações verificáveis, que permitem a verificação sem revelar detalhes sensíveis ou privados.

4.3 ESCOPO DA ARQUITETURA

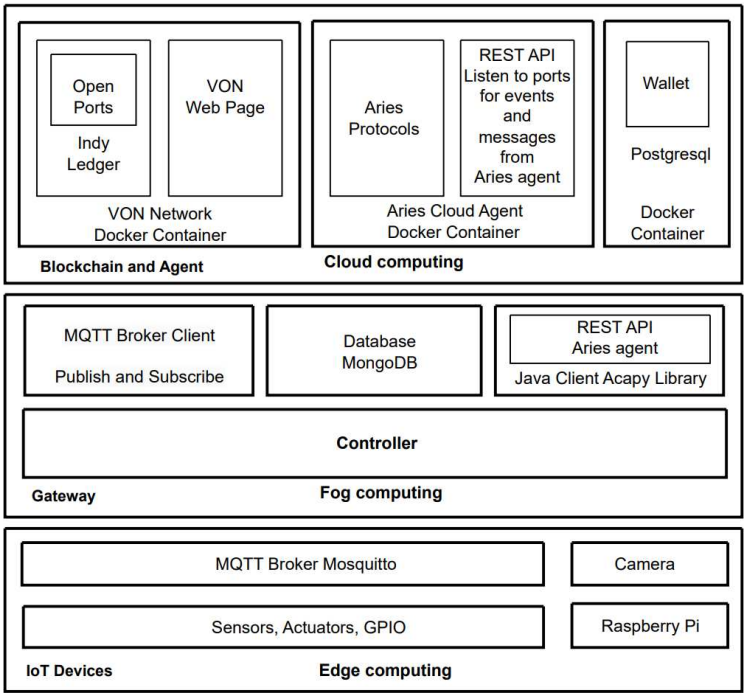
A abordagem proposta consiste em uma arquitetura dividida em três camadas interconectadas e colaborativas: *Edge*, *Fog* e *Cloud Computing*, que agregam um total de 6 entidades. Na primeira camada, encontram-se os dispositivos IoT e o *broker* MQTT. Na segunda camada, estão o banco de dados *NoSQL* e o controlador. Já na terceira camada, encontra-se o agente Aries de nuvem e a blockchain *Von Network*. A Figura 12 ilustra estas camadas juntamente com suas respectivas subentidades.

Para compreender plenamente o papel de cada entidade, as ações e interações serão detalhadas nas Subseções 4.3.1, 4.3.2 e 4.3.3, juntamente com as respectivas subseções. Isso proporcionará uma visão clara do comportamento dinâmico dentro da arquitetura proposta.

4.3.1 Edge computing

Esta camada desempenha um papel no acesso e na coleta de dados provenientes de dispositivos IoT. Ela permite que a camada *Fog* colete e transmita informações para dispositivos IoT de natureza restrita, seja por meio de um *broker* ou de uma interação direta com dispositivos de maior capacidade (não restritos). As funções das entidades envolvidas nesta camada serão abordadas a seguir.

Figura 12 – Visão geral da arquitetura proposta.



Fonte: elaborada pelo autor (2023).

4.3.1.1 Dispositivos IoT

Engloba os dispositivos físicos encarregados da coleta de dados na interação com o ambiente. Ela pode ser constituída por uma diversidade de dispositivos IoT, tais como controladores, sensores, atuadores, câmeras de segurança, dispositivos de saúde inteligentes, e outros componentes que podem ser incorporados de forma manual.

4.3.1.2 Broker MQTT

Atua como responsável por intermediar a comunicação entre dispositivos. Sua função abrange desde a recepção até o roteamento e encaminhamento eficiente de mensagens entre os dispositivos. No cerne desta operação, encontram-se os tópicos - canais de comunicação que constituem a estrutura fundamental do MQTT. Esses tópicos são essenciais, pois permitem que os dispositivos se inscrevam para receber ou publicar mensagens neles.

4.3.2 Fog computing

A camada de *Fog Computing* atua como intermediária para facilitar a comunicação segura e eficiente entre as camadas de *Edge* e *Cloud Computing*. No âmbito da proposta em questão, o foco principal recai sobre esta camada, onde o sistema

desenvolvido atua. Nesse contexto, o sistema utiliza bibliotecas para implementar um cliente MQTT, acesso a banco de dados para o armazenamento dos dados publicados no *broker*, e integra funcionalidades de SSI provenientes das bibliotecas Indy SDK e ACA-Py. A seguir, serão exploradas em detalhes as funcionalidades que compõem esta camada.

4.3.2.1 Broker MQTT client

O cliente MQTT desempenha um papel essencial ao gerenciar inscrições e publicações nos tópicos especificados, simplificando a transmissão de diversas formas de informações, tais como fluxos de dados, telemetria, texto, áudio, vídeo e eventos. Sua principal responsabilidade recai na coleta de dados e na identificação dos dispositivos. Essa identificação engloba a atribuição de um identificador único, registro da data e hora, informações georreferenciadas e a categorização do tipo de dispositivo envolvido.

4.3.2.2 Banco de dados IoT

Um banco de dados pode ser empregado para armazenar uma variedade de informações, incluindo DIDs, atributos e dados coletados de dispositivos IoT. Esta organização de dados facilita a recuperação para processamentos futuros, como a emissão de credenciais verificáveis sob demanda. Para garantir a integridade e a disponibilidade desses dados, cada coleta pode ser associada a um *hash* único e registrado. Isso proporciona uma camada adicional de segurança, assegurando que os dados permaneçam íntegros e acessíveis quando necessário.

4.3.2.3 Processos do sistema

No cerne desta pesquisa encontra-se o sistema desenvolvido. Por meio das operações implementadas, o sistema tem como principal objetivo processar as informações originadas na camada *edge* (ponto de coleta) e encaminhá-las para a camada de nuvem, onde operam o agente e a blockchain.

O processo se inicia quando a aplicação estabelece uma conexão segura com o *broker*, monitorando ativamente os tópicos para interceptar mensagens em trânsito. As informações capturadas abrangem detalhes sobre o dispositivo que originou a transação, juntamente com a mensagem, dados ou arquivos associados.

Para a identificação dos dispositivos, uma função específica é acionada com o objetivo de gerar um novo DID e registrá-lo na blockchain. Este processo envolve a vinculação do DID a um identificador único do dispositivo, que pode ser o número de série, UUID ou endereço MAC. Esta funcionalidade tira proveito das capacidades da biblioteca *Indy SDK*, que é utilizada para implementar os DIDs e facilitar a interação com

a blockchain. Adicionalmente, existe a possibilidade de gerar uma identificação única para os dados coletados. Isso é realizado através da geração de um DID utilizando o mesmo método e registrando-o na blockchain. Este recurso aprimora significativamente o processo de identificação e rastreamento das evidências coletadas, garantindo uma maior confiabilidade e segurança dos dados.

Na Figura 13, é apresentada uma pequena parte da estrutura JSON que compõe a credencial emitida com base nos dados de telemetria coletados de dispositivos IoT. Esses dados incluem identificação do operador responsável ou perito, o DID previamente atribuído ao dispositivo, o DID associado a cada transmissão, carimbos de data/hora, localização do dispositivo e as leituras de sensores transmitidas.

Figura 13 – Credencial emitida pela aplicação.


```
"credential_proposal": {
  "@type": "https://didcomm.org/issue-credential/1.0/credential-preview",
  "attributes": [
    {
      "name": "Operator",
      "value": "LRG UFSC"
    },
    {
      "name": "device_did",
      "value": "did:sov:7jfceWZRA7jYdb866iFHRR"
    },
    {
      "name": "collection_did",
      "value": "did:sov:L5DknuZMuzchXyXLvmfAuX"
    },
    {
      "name": "data",
      "value": "50.5F"
    },
    {
      "name": "device_type",
      "value": "TemperatureSensor"
    },
    {
      "name": "location",
      "value": "living_room"
    },
    {
      "name": "timestamp",
      "value": "2023-10-02 00:31:03.669"
    }
  ]
},
"schema_id": "B3qPp5s37rQNYpWSUs9bXC:2:lrg:1.0",
"cred_def_id": "B3qPp5s37rQNYpWSUs9bXC:3:CL:82542:lgr"
```


Fonte: elaborada pelo autor (2023).

Após a recepção, os dados são processados e formatados de acordo com requisitos predefinidos em esquemas e definições de credenciais. Arquivos grandes ou de *streaming*, que não podem ser inseridos diretamente na credencial, podem ser acessados externamente, desde que permaneçam íntegros. Para verificar a integridade, é gerado um *hash* do arquivo que será inserido na credencial. Além disso, a referência ao local de indexação pode ser incluída na credencial. Entre os demais atributos que compõem a credencial, estão o DID do dispositivo, carimbo de data e hora, informações georreferenciadas e tipo de dispositivo.

Um controlador presente na aplicação atua como responsável por definir as

regras de negócio e instanciar recursos do agente Aries em nuvem. Este agente lida com a criação de esquemas e modelos de credenciais, que são previamente registrados na blockchain. Adicionalmente, o controlador desempenha um papel essencial na criação de conexões com outros agentes, gerenciando tanto o envio quanto o recebimento de convites de conexão. Por fim, o controlador é acionado para realizar a comunicação com o outro agente, gerando e emitindo a credencial verificável que inclui a identificação do dispositivo, atributos e dados coletados que formam a evidência.

4.3.3 Cloud computing

Nesta camada, encontra-se o agente Aries, que desempenha um papel fundamental ao atuar como interface para o Gateway da arquitetura. Além disso, essa camada inclui um banco de dados dedicado ao armazenamento de chaves criptográficas, registros de conexões e credenciais. E, de igual importância, a blockchain opera para garantir a imutabilidade e a verificação das transações, assegurando a integridade e a rastreabilidade das evidências. As entidades que fazem parte desta camada serão apresentadas a seguir.

4.3.3.1 Aries Cloud Agent

O agente Aries desempenha um papel crucial ao processar solicitações originadas no controlador presente na camada *Fog*. Sua estrutura é equipada com o procedimento necessário para iniciar conexões, responder a solicitações, transmitir mensagens, gerenciar o armazenamento de forma segura, interagir com outros agentes, emitir credenciais e criar apresentações verificáveis. Além disso, essa camada possui recursos de processamento para lidar com criptografia, oferecendo suporte para armazenamento, processamento e fontes de entropia necessárias à geração de chaves criptográficas. Adicionalmente, é responsável por realizar comunicações e transações na blockchain.

4.3.3.2 Rede Von

A blockchain *Von Network* é utilizada para o registro distribuído verificável. Ela oferece recursos avançados para a publicação de DIDs públicos, a criação de esquemas e a definição de credenciais, possibilitando a verificação descentralizada de DIDs e credenciais verificáveis.

4.3.4 Validadores e validação

Com o uso de credenciais verificáveis, os agentes integrados em dispositivos IoT ou associados a eles asseguram que os dados emitidos - sejam leituras de sensores, transmissões de texto, áudios, vídeos ou eventos - sejam assinados em sua origem.

Isso torna os dados imunes a adulterações e permite a verificação de sua procedência por meio da blockchain, garantindo a integridade dos dados após sua geração.

Esta metodologia estabelece uma base sólida de confiança nos dados gerados, contanto que o dispositivo seja considerado confiável. A construção desta confiança pode ser validada pela atuação de auditores independentes ou entidades reguladoras governamentais, que certificam o funcionamento do dispositivo e, consequentemente, a legitimidade das credenciais verificáveis emitidas.

Vale ressaltar que a estrutura proposta não define normas e critérios precisos para a validação das credenciais e para a certificação dos dispositivos, uma vez que isso exigiria conhecimento especializado em áreas como perícia forense, agências de fiscalização, órgãos regulatórios ou sistema judicial. A abordagem adotada parte do pressuposto de que esta estratégia é viável e é demonstrada de forma simulada, emitindo dados fictícios provenientes de sensores e credenciais digitais que podem ser verificadas por meio da tecnologia blockchain.

4.4 FERRAMENTAS E TECNOLOGIAS ADOTADAS

As tecnologias selecionadas foram alinhadas com as características discutidas no Capítulo 2. Na avaliação das tecnologias disponíveis que suportam DIDs e VCs e blockchain, considerou-se: a maturidade e cobertura dos recursos necessários para implementar esta proposta; a existência de técnicas que garantam privacidade, integridade e rastreabilidade dos dados; e a escalabilidade do blockchain permissionado. O Quadro 2 oferece uma breve descrição de cada ferramenta e tecnologia empregada neste contexto.

Os componentes da blockchain Indy (*Von Network*), o agente ACA-Py, o *broker* Mosquitto, além dos bancos de dados PostgreSQL e MongoDB, são implementados em contêineres, utilizando Docker e Docker Compose para facilitar a implantação e a gestão.

A rede *Von Network* desempenha um papel chave na facilitação da implantação da blockchain Indy, enquanto a gestão de carteiras digitais, essenciais para o armazenamento seguro de credenciais e chaves criptográficas, adota o padrão indy-wallet, utilizando PostgreSQL como solução de armazenamento.

Para o gerenciamento eficaz das mensagens entre dispositivos IoT, o *broker* MQTT Mosquitto é utilizado, permitindo a inscrição e a publicação de tópicos de forma eficiente e segura. A comunicação com o *broker* MQTT é estabelecida através da biblioteca MQTT Paho, garantindo conexões confiáveis.

No Gateway proposto, a biblioteca MQTT Paho é essencial para estabelecer conexões confiáveis com o *broker* MQTT, assegurando a eficiência na comunicação entre dispositivos IoT. A segurança dessas conexões é reforçada pelo uso da biblioteca Bouncy Castle, que facilita operações seguras com certificados digitais. Além disso, a

Quadro 2 – Ferramentas utilizadas na proposta.

Ferramentas	**Descrição**
Hyperledger Indy	Base para a blockchain *Von Network* e biblioteca ACA-Py.
Hyperledger Aries	Emprega o ACA-Py para comunicação *off-ledger* entre agentes.
Indy SDK	Cria e registra DIDs Sovrin na blockchain.
ACA-Py	Base para criar agentes Hyperleger Aries.
ACA-Py Client Library	Facilita a comunicação com o agente ACA-Py via APIs.
Von-network	Blockchain Indy para registro de dados verificáveis.
Mosquitto	Facilita a comunicação entre dispositivos em uma rede IoT ao gerenciar a troca de mensagens entre publicadores e assinantes.
Java Paho MQTT Client	Implementa a comunicação com o broker MQTT Mosquitto.
Bouncy Castle	Biblioteca para leitura de certificados x509 na comunicação entre Gateway e *broker* Mosquitto.
PostgreSQL	Armazena credenciais e chaves criptográficas.
MongoDB	Armazena dados de dispositivos IoT.
Docker Community	Facilita a implantação das aplicações e sistemas em contêineres.

Fonte: Elaborado pelo autor (2023).

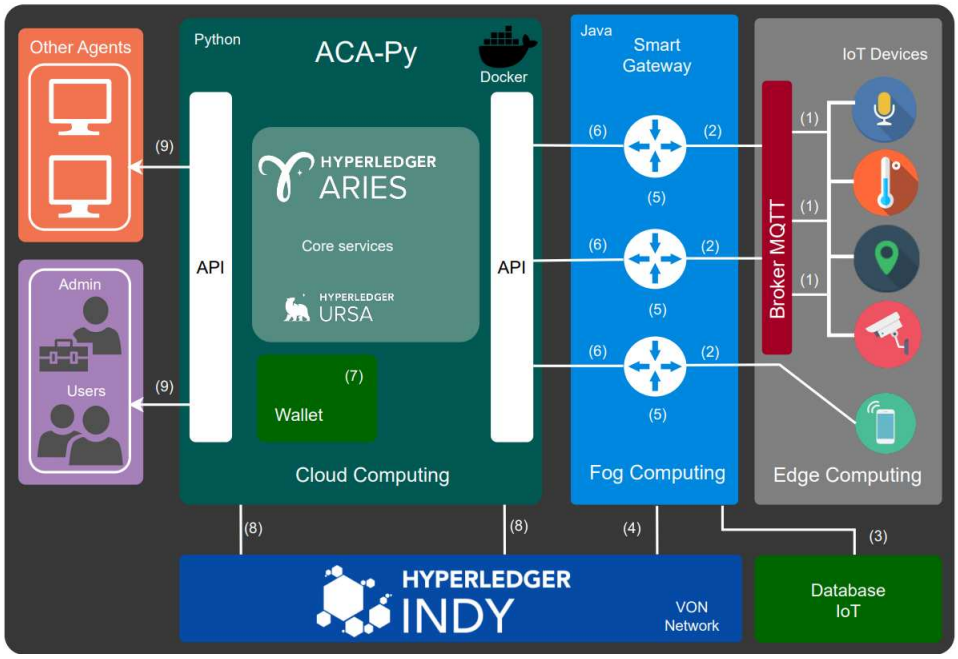
implementação do controlador, que aproveita as funcionalidades do agente ACA-Py, é realizada através da biblioteca ACA-Py Java Client. Para a criação e registro de DIDs na blockchain Indy, a biblioteca Indy SDK é utilizada, empregando o método Sovrin para garantir a integridade e a segurança dos identificadores digitais.

4.5 SEQUÊNCIA COMPLETA DE AÇÕES

Esta seção detalha as ações realizadas pelas entidades, destacando a função do Gateway na identificação de dispositivos e na geração de credenciais a partir dos dados emitidos. A Figura 14 esquematiza esta sequência de ações, demonstrando como a arquitetura proposta permite que dispositivos IoT restritos interajam de maneira segura e confiável com o ecossistema de SSI oferecido pelo agente de nuvem Aries. A sequência ilustra o funcionamento da arquitetura dentro da cadeia de custódia de dados.

1. Estabelecimento de Conexões Seguras: Dispositivos utilizam certificados para estabelecer conexões seguras e enviar dados via MQTT para um tópico designado.
2. Assinatura e Recebimento de Dados pelo Gateway: Utilizando um certificado próprio, o Gateway conecta-se ao *broker*, assina ao tópico relevante e recebe os dados transmitidos.

Figura 14 – Integração das entidades na arquitetura proposta.



Fonte: elaborada pelo autor (2023).

3. Armazenamento de Dados: Os dados podem ser armazenados em um banco de dados para facilitar consultas futuras.
4. Identificação e Registro na Blockchain: O Gateway identifica o dispositivo, registrando o DID e o número de série na blockchain. Um registro adicional pode ser feito para cada conjunto de dados coletados, vinculado ao DID do dispositivo.
5. Criação e Incorporação de Pacotes de Dados: Informações como *timestamp*, tipo de dispositivo, localização, e os dados coletados são encapsulados em um pacote. Para arquivos que não podem ser diretamente incluídos, um *hash* é gerado para assegurar a integridade. O *hash* e o link para acesso ao arquivo são registrados na credencial.
6. Geração de Credenciais pelo Gateway: Utilizando APIs, o Gateway acessa recursos do agente na nuvem para gerar as credenciais.
7. Implementação da Carteira Indy: Uma carteira Indy é implementada usando PostgreSQL para armazenar elementos críticos como chaves criptográficas, DIDs, VCs e outras informações sensíveis necessárias para a interação com a rede.
8. Interação com a Blockchain: O agente gerencia transações na blockchain, incluindo a criação de DIDs, atributos e definições de esquemas e mode-

los de credenciais, possibilitando a verificação e validação das credenciais emitidas.

9. Entrega e Armazenamento de Credenciais: Credenciais são entregues e armazenadas em carteiras digitais, disponíveis para consulta e verificação por entidades autorizadas, utilizando a blockchain para confirmar a autenticidade.

Esta estrutura possibilita que os titulares armazenem essas credenciais e as apresentem aos verificadores ou criem provas de conhecimento zero ao elaborar apresentações verificáveis. As partes interessadas, por sua vez, podem verificar a autenticidade dessas credenciais, para isso, elas consultam a blockchain para acessar a chave pública do emissor, que corresponde à chave privada usada para assinar a credencial. Isso permite a verificação não apenas da autenticidade, mas também da integridade das informações, através da análise do *hash* de assinatura de cada atributo da credencial. Adicionalmente, o rastreamento do DID associado à coleta e ao dispositivo na blockchain oferece uma camada extra de segurança, assegurando a custódia dos dados.

4.5.1 Mais detalhes da arquitetura

Os dispositivos de IoT, exemplificados por um sensor de temperatura, estabelecem conexões seguras utilizando certificados x509. As mensagens enviadas ao *broker* são transmitidas através da função de publicação (*publish*) em um tópico específico.

O Gateway implementa uma biblioteca cliente MQTT para assinar o tópico e outra biblioteca para gerenciar certificados x509. Todas as mensagens que circulam neste tópico são recebidas e processadas pelo Gateway, estabelecendo-se assim a correlação entre a mensagem e o dispositivo de origem.

A arquitetura prevê duas situações de uso para o registro de DIDs na blockchain, além dos agentes envolvidos e objetos associados às credenciais verificáveis. Esses DIDs são criados através do Indy SDK, uma vez que o ACA-Py implementa recursos destinados à manipulação de objetos relacionados a credenciais, protocolos e comunicação com outros agentes.

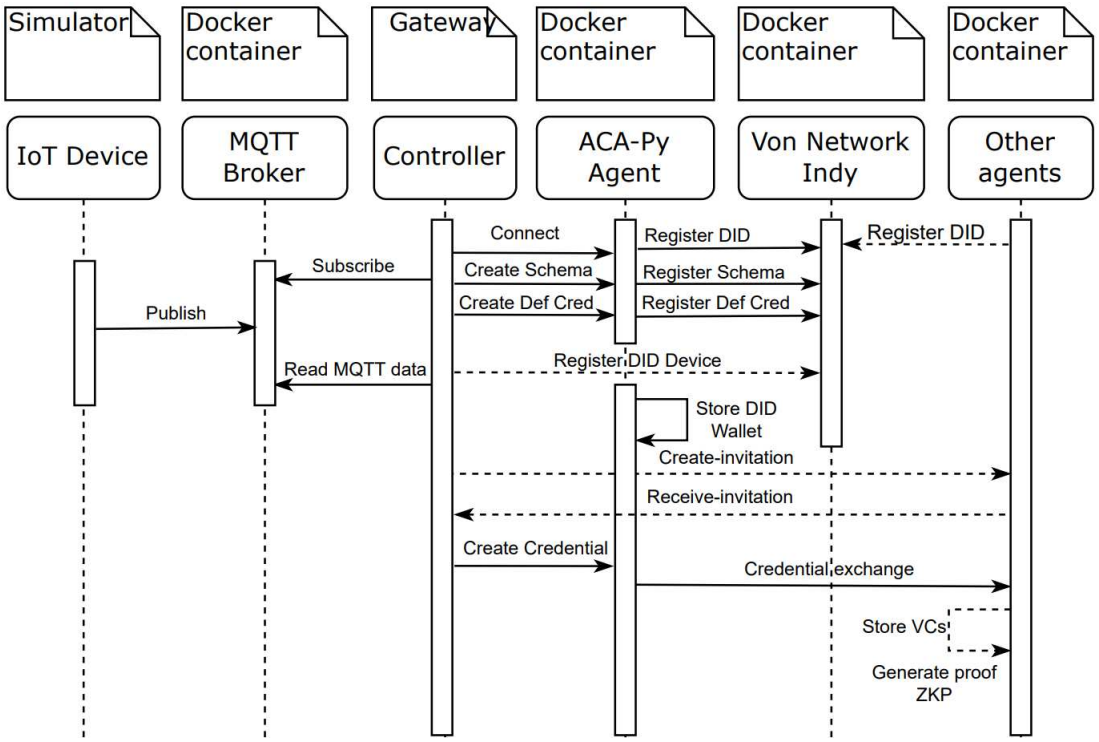
Na primeira situação, uma identificação do dispositivo pode ser registrada na blockchain através de um DID do tipo Sovrin ("did:sov"). Isso serve para manter um registro dos dispositivos registrados na arquitetura. Esse registro vincula na blockchain um identificador único do dispositivo, como um número de série, MAC ou qualquer outro identificador. Esse registro é realizado apenas uma vez, evitando assim interações contínuas com a blockchain.

Na segunda situação, embora as credenciais já sirvam para garantir a certificação dos dados, DIDs podem ser registrados na blockchain para registrar uma publicação ou um lote de publicações MQTT, com o objetivo de manter um identificador com

um *hash* associado, que pode ser entendido como um identificador para evidências. No entanto, esse processo exigirá interações contínuas com a blockchain, o que pode causar algum atraso no sistema.

O diagrama de sequência na Figura 15 ilustra a interação entre os atores e os componentes-chave na arquitetura proposta, que estabelece um sistema intermediário para facilitar a comunicação segura entre dispositivos IoT restritos e o agente de nuvem Aries, oferecendo recursos de SSI.

Figura 15 – Diagrama de sequência.



Fonte: elaborada pelo autor (2023).

Uma biblioteca cliente disponibiliza métodos que se comunicam com o agente Aries por meio de APIs. Desta forma, o Gateway atua como controlador do agente na nuvem, seguindo o padrão proposto pela comunidade Hyperledger. Para criar um agente, é necessário um DID público registrado na blockchain. Esse DID é criado com base em uma semente que foi previamente registrada na blockchain *Von Network*. Com esse DID para identificar o agente na blockchain, é possível realizar transações para criar esquemas, definir credenciais e gerenciar o envio e recebimento de conexões.

Os esquemas e definições de credenciais devem ser criados e registrados na blockchain antes da criação e emissão da credencial. A credencial é criada de acordo com um modelo que é a definição de credencial, que por sua vez, foi criada utilizando um esquema. Esse esquema inclui atributos que visam identificar a origem da evidên-

cia, como o DID do dispositivo, operador, tipo de dispositivo, dados coletados, DID da coleta, localização do dispositivo e carimbo de data e hora.

As Figuras 16 (a-d) ilustram as transações realizadas na blockchain da *Von Network*, detalhando cada etapa do processo de registro. A figura 16a apresenta o registro de NYM (Nome de Identificação), que é necessário para o registro de DIDs. A Figura 16b detalha o registro de ATTR (Atributos), mostrando os atributos associados a um DID específico. A Figura 16c revela o registro de Schema, correspondente ao esquema da credencial, enquanto a Figura 16d descreve o registro de CRED_DEF (Definição de Credencial), estabelecido a partir do esquema para a efetiva emissão da credencial.

(a) Transação NYM.

(b) Transação ATTRIB.

(c) Transação SCHEMA.

(d) Transação CRED_DEF.

Figura 16 – Tipos de transações na blockhain Von Network.

Através das Figuras 16c e 16d, é possível observar os atributos especificados para a composição da credencial, demonstrando o detalhamento e a estruturação necessários para sua validação e uso na blockchain.

Os dados relacionados a credenciais e chaves privadas são mantidos de maneira segura em uma carteira digital armazenada em um banco de dados PostgreSQL

por meio da *indy-wallet*.

A conexão com outros agentes pode ser estabelecida usando o conceito de protocolo "*Out-of-Band*". Esse protocolo é útil quando se deseja interagir com outro agente e não se dispõe de uma conexão DIDComm para usar na interação. Assim, um convite pode ser gerado com base na chave pública da outra parte, possibilitando o início da comunicação.

Após a configuração inicial da conexão, ocorre a troca de credenciais. O agente tem a capacidade de armazenar a credencial vinculada à emissão dos dispositivos IoT e, além disso, pode criar apresentações seletivas de dados, com o objetivo de garantir privacidade e reforçar a segurança. Outros domínios interessados podem consultar a blockchain para verificar a autenticidade das credenciais.

4.6 CONSIDERAÇÕES FINAIS

Na integração das tecnologias empregadas na proposta, é possível estabelecer comunicações seguras com outros domínios utilizando os protocolos de comunicação Aries. Além disso, a rastreabilidade dos dados emitidos pelos dispositivos de IoT é garantida através do uso de DIDs, VCs e blockchain. Esta abordagem assegura que os dados coletados pelos dispositivos de IoT preservem sua integridade e estejam resistentes a adulterações.

As credenciais desempenham um papel crucial no registro e certificação das informações, proporcionando um histórico confiável e transparente. É factível utilizar VCs para autenticar atributos e informações emitidas pelos dispositivos de IoT, com o intuito de prevenir o repúdio, ou seja, a negação subsequente das interações ou alegações feitas. Este recurso é de particular importância neste contexto de preservação de evidências, assegurando que as mesmas não possam ser contestadas posteriormente. No próximo capítulo, apresenta-se as etapas da configuração da arquitetura e detalhes de como o Gateway foi implementado.

5 DESENVOLVIMENTO DA PROPOSTA

Este capítulo detalha o desenvolvimento da proposta, organizado em quatro seções principais. Na Seção 5.1, apresentam-se as etapas e configurações necessárias para a implantação de uma rede de produção do Hyperledger Indy por meio da *Von Network* e agentes Aries utilizando o ACA-Py em um ambiente local. Na Seção 5.2, concentra-se a atenção no design da arquitetura, com especial foco no Gateway. Explica-se a implementação do controlador e das estruturas de dados essenciais, empregando a biblioteca Indy SDK e o ACA-Py. Adicionalmente, aborda-se a criação de convites de comunicação, esquemas, definições de credenciais e as credenciais verificáveis.

5.1 IMPLANTAÇÃO DA REDE LOCAL

Para o desenvolvimento e validação da proposta, foi necessária a implementação de uma rede Hyperledger Indy, utilizando a infraestrutura da rede *Von Network*. Os Agentes Aries foram empregados como interfaces para o Gateway, facilitando a interação com a rede. Esta seção fornece uma descrição das etapas gerais na criação da rede, visando oferecer um entendimento abrangente do desenvolvimento realizado.

5.1.1 Etapas gerais para implantar a blockchain Von Network e o agente Aries ACA-Py

A *Von Network* oferece duas opções aos desenvolvedores: uma configurável para operação em ambiente local e outra que disponibiliza uma instância de teste pública. Ambas as opções envolvem a participação de quatro *peers*, os quais desempenham um papel crucial na validação das transações. Optou-se pela configuração local para maior controle e acesso, atendendo às necessidades específicas desta proposta.

5.1.1.1 Rede Von

As etapas para a criação de uma rede *Von Network* incluem:

1. Criar imagens Docker para a rede *Von Network*.
2. Iniciar a rede *Von Network* local.
3. Obter o arquivo de gênesis da rede por meio do *endpoint/genesis*.
4. Autenticar um novo DID.

Em relação aos agentes, criaram-se dois perfis distintos: um como emissor e outro como detentor das credenciais. Procedimentos para a gestão de carteiras e conexões com a blockchain foram estabelecidos.

5.1.1.2 Aries Cloud Agent Python

As etapas para criar os agentes incluem:

1. Criar imagens Docker para o ACA-Py.
2. Configurar o arquivo Docker Compose com os parâmetros necessários.
3. Iniciar uma instância do banco de dados para persistência das carteiras.
4. Iniciar os agentes, incluindo emissor, detentor e, opcionalmente, verificador.

5.1.2 Ambiente com imagens docker

Utilizaram-se contêineres para implementar todas as funcionalidades do Hyperledger Indy e Aries, visando simplificar e otimizar o processo. Detalha-se o procedimento para a implementação da rede *Von Network* como blockchain Indy e do ACA-Py como agentes Aries nas subseções subsequentes.

5.1.2.1 Iniciar uma rede Von Network local

Para construir uma blockchain Indy Node de testes através da *Von Network*, inicialmente, clona-se o repositório *von-network* e, em seguida, acessa-se o diretório correspondente por meio dos comandos 5.1.

Listing 5.1 – Clonar repositório

```
1         $git clone https://github.com/bcgov/von-network
2         $cd von-network
```

Após a clonagem do repositório, serão construídas imagens docker para a rede *Von Network* a partir do script correspondente e, em seguida, a rede Indy será iniciada, conforme os comandos 5.2.

Listing 5.2 – Iniciar a rede blockchain Indy

```
1         $./manage build
2         $./manage start --logs
```

Isso inicia quatro nós Indy e um von-webserver, conforme pode ser observado na Figura 17.

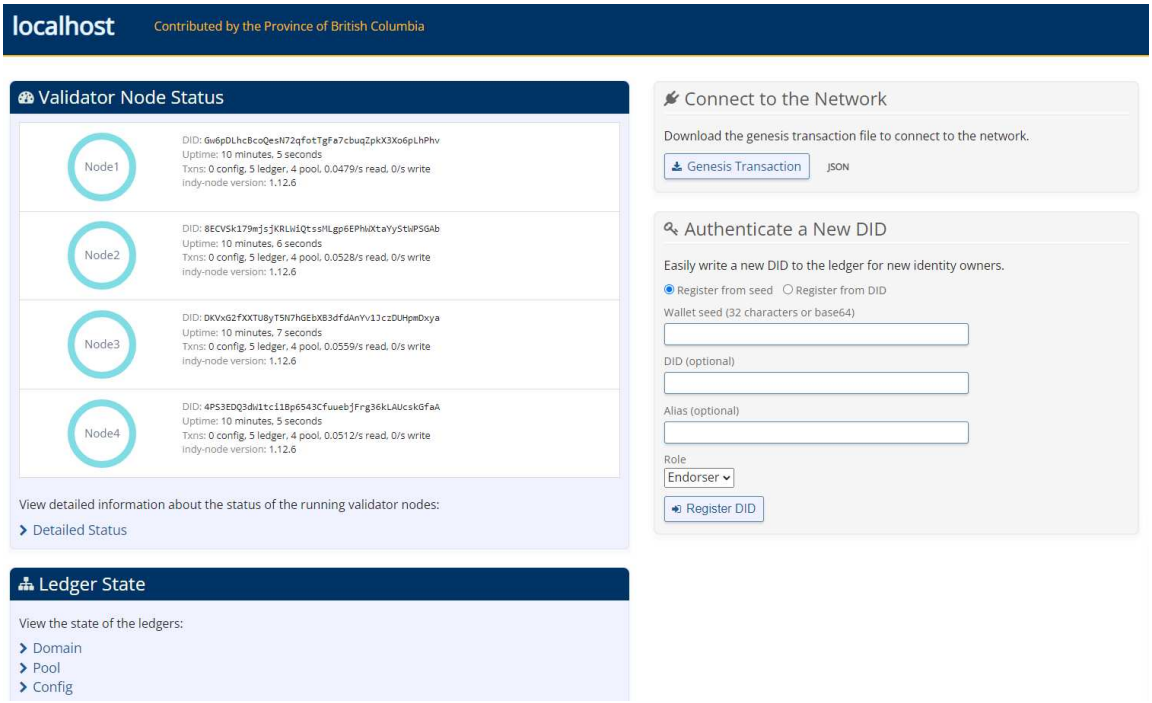
Figura 17 – Containers Docker.

```
$ docker ps
CONTAINER ID   IMAGE              COMMAND                  PORTS                                   NAMES
6aafa542817f   von-network-base   "./scripts/start_nod…"   0.0.0.0:9703-9704->9703-9704/tcp        von-node2-1
9cbce215338b   von-network-base   "./scripts/start_nod…"   0.0.0.0:9705-9706->9705-9706/tcp        von-node3-1
4df8470fe04a   von-network-base   "./scripts/start_nod…"   0.0.0.0:9701-9702->9701-9702/tcp        von-node1-1
d044945cf19d   von-network-base   "bash -c 'sleep 10 &…"   0.0.0.0:9000->8000/tcp                  von-webserver-1
74d38a006006   von-network-base   "./scripts/start_nod…"   0.0.0.0:9707-9708->9707-9708/tcp        von-node4-1
```

Fonte: elaborada pelo autor (2023).

O *von-webserver* dispõe de uma interface web em *localhost:9000* que permite visualizar o status dos nós da rede, acessar o arquivo gênesis da rede, criar um DID e navegar pelas transações no blockchain, conforme ilustrado na Figura 18.

Figura 18 – Servidor web da rede Von Network.



Fonte: elaborada pelo autor (2023).

Em relação as transações, é possível consultar três livros que compõem uma rede *Von Network*. O razão do domínio, onde residem os DIDs, o esquema, etc. O livro-razão do Pool, onde o conjunto de nós da rede é rastreado. O livro-razão de configuração, onde as alterações na configuração da rede são rastreadas.

Na página do *von-webserver*, conforme mostrado na Figura 18, há duas partes importantes: baixar a transação gênesis para ingressar na rede e registrar um novo DID. Assim como acontece com a maioria dos blockchains, para ingressar é preciso obter a transação gênesis. Na rede de produção Sovrin é preciso pagar para adicionar um DID ao blockchain, mas localmente não.

5.1.2.2 Etapas gerais para criar agentes usando a biblioteca ACA-Py

O ACA-Py depende da biblioteca *libindy*. É possível instalar o *libindy* ou executar um contêiner docker que o contenha. O *libindy* e suas dependências são definidos em um arquivo *Dockerfile*, que é usado para criar uma imagem Docker. Essa imagem é então utilizada no processo de criação do contêiner ao executar o *docker-compose*.

O código 5.3 apresenta um exemplo de como configurar as definições de um agente em um arquivo *docker-compose*.

Listing 5.3 – Exemplo docker-compose.yml

```
version: "3"
services:
vcr-agent:
build:
context: ../../
dockerfile: docker/Dockerfile.run
ports:
        - 8010:8010
        - 8001:8001
networks:
        - wallet-net
entrypoint: /bin/bash
command: [
        "-c",
        "sleep 5; \
        ACAPy start \
        --label 'IoT_agent' \
        --admin '0.0.0.0' 8010 \
        --admin-insecure-mode \
        --endpoint 'http://host.docker.internal:8001' \
        --outbound-transport http \
        --inbound-transport http '0.0.0.0' 8001 \
        --genesis-url 'http://test.bcovrin.vonx.io/genesis' \
        --seed 'Sagittarius_a0000000000000000000' \
        --auto-provision \
        --auto-accept-invites \
        --auto-accept-requests \
        --auto-ping-connection \
        --auto-respond-messages \
        --auto-respond-credential-proposal \
        --auto-respond-credential-offer \
        --auto-respond-credential-request \
        --wallet-type 'askar' \
        --wallet-name 'acapy_agent_wallet' \
        --wallet-key 'key' \
        --wallet-storage-type 'postgres_storage' \
        --wallet-storage-config '{\"url\":\"wallet-db:5432\"}' \
        --wallet-storage-creds '{\"account\":\"DB_USER\",\"password\":\"
            db_pass\",\"admin_account\":\"postgres\",\"admin_password\":\"
            mysecretpassword\"}' \
        --log-level 'info' ",
        ]
        networks:
```

```
42          wallet-net:
43          external:
44          name: docker-test_wallet-net
```

Os arquivos mencionados permitem a configuração de uma instância do ACA-Py utilizando um banco de dados PostgreSQL. No arquivo *docker-compose.yml*, especificamente no código 5.3, há uma série de parâmetros fundamentais para a inicialização do agente:

- ***label*:** Este é o nome atribuído a esta instância do ACA-Py, que será utilizado nas mensagens.
- ***admin*:** Define o endereço IP (0.0.0.0) e a porta (8010) para a interface de administração do aplicativo/controlador. Além disso, essa porta permite o acesso à documentação OpenAPI/Swagger para esta instância do ACA-Py.
- ***admin-insecure-mode*:** Essa opção permite o uso do servidor web administrativo sem a necessidade de uma chave de API. Importante ressaltar que essa opção não deve ser habilitada em ambientes de produção, devido a possíveis vulnerabilidades.
- ***inbound-transport*:** Define o protocolo, endereço IP (0.0.0.0) e porta (8001) nos quais outras instâncias do ACA-Py podem se conectar a esta instância.
- ***outbound-transport*:** Especifica o protocolo que esta instância do ACA-Py usará para se comunicar com outras instâncias.
- ***endpoint*:** Define o URL no qual esta instância do ACA-Py estará disponível para acesso por outras instâncias do ACA-Py. É importante notar que o protocolo, endereço IP e porta devem coincidir com os definidos em *–inbound-transport*.
- ***genesis-url*:** Aponta para o servidor da rede *Von Network* onde o arquivo *genesis* pode ser encontrado. Outros argumentos de linha de comando podem ser utilizados para apontar para um arquivo no disco, se necessário.
- ***seed*:** Define a semente (por exemplo, *Sagittarius_a0000000000000000000*) que será usada para gerar um DID público registrado no razão. A criação de DIDs públicos é possível através do servidor da rede *Von Network*, que permite o registro de DIDs a partir de uma semente. Como alternativa, o registro de um DID pode ser realizado via *localhost:9000/register*, utilizando a mesma semente na configuração.
- ***auto-provision*:** Parâmetro para automatizar o provisionamento da carteira, caso ela não exista.
- ***auto-accept-invites* e *auto-accept-requests*:** Quando habilitados, esses parâmetros automatizam a aceitação de convites e solicitações de conexão,

resultando na criação automática de uma conexão após a aceitação da solicitação.

- ***auto-ping-connection*:** Após o estabelecimento da conexão, esta opção envia automaticamente uma mensagem de ping para marcar a conexão como 'ativa'.
- ***auto-respond-messages*:** Ao ativar essa opção, o agente responderá automaticamente a mensagens recebidas.
- ***auto-respond-credential-proposal*, *auto-respond-credential-offer* e *auto-respond-credential-request*:** Esses parâmetros automatizam a aceitação de propostas, ofertas e solicitações de credenciais.
- ***wallet-type askar*:** Esta configuração instrui a biblioteca Indy (*libindy*) a criar uma carteira para a comunicação com um razão Indy.
- ***wallet-name acapy_agent_wallet*:** O nome da carteira criada pela *libindy*. Se nenhum nome for fornecido, o padrão será utilizado.
- ***wallet-key secret*:** Essa chave é necessária para acessar a carteira e, quando usada em conjunto com *–auto-provision*, é usada para provisionar a carteira.
- ***wallet-storage-type, wallet-storage-config* e *wallet-storage-creds*:** Esses parâmetros configuram detalhes relacionados à carteira, como o tipo de armazenamento (neste caso, '*postgres_storage*'), as configurações do contêiner onde a carteira está sendo executada e as informações de autenticação para o banco de dados da carteira.
- ***log-level*:** Este parâmetro controla o nível de detalhamento dos *logs* durante a execução do agente ACA-Py.

Após a configuração adequada do *docker-compose.yml*, ao executar o comando 5.4, o agente é inicializado.

Listing 5.4 – Iniciar o agente 1

```
1          $docker-compose -f docker-compose-agent.yml up
```

Esses contêineres possuem a flexibilidade de serem implantados tanto na mesma máquina física quanto distribuídos em várias máquinas em diferentes redes. Os testes iniciais foram conduzidos com a implantação de todos os contêineres na mesma máquina física.

5.1.3 Broker MQTT e simulador de dispositivo IoT

O *broker* foi implementado com a imagem do Mosquitto. Certificados autoassinados foram gerados para assegurar comunicações seguras entre as partes. Esses

certificados são configurados no *broker* Mosquitto e no simulador dos sensores de temperatura. O simulador envia mensagens que contêm não só a leitura de telemetria, mas também informações adicionais, como a localização do dispositivo e um identificador, como o número de série.

5.2 DESIGN DA ARQUITETURA E IMPLEMENTAÇÃO DO GATEWAY

Nesta seção, serão introduzidos os componentes que compõem o Gateway, começando pela integração da biblioteca MQTT client para estabelecer uma conexão segura com o *broker*, incluindo a gestão de certificados. Em seguida, será aprofundada a implementação dos recursos da biblioteca Indy SDK para criar e registrar DIDs na blockchain. A utilização dessa biblioteca é imprescindível, visto que a ACA-Py se especializa na administração de credenciais verificáveis e na comunicação com outros agentes. Por último, será apresentado o papel do controlador na arquitetura, que desempenha uma função central instanciando recursos do agente na nuvem e gerenciando a comunicação com outros domínios representados pelo agente "*Holder*", esquemas, definições de modelos de credencial e emissão de credencial.

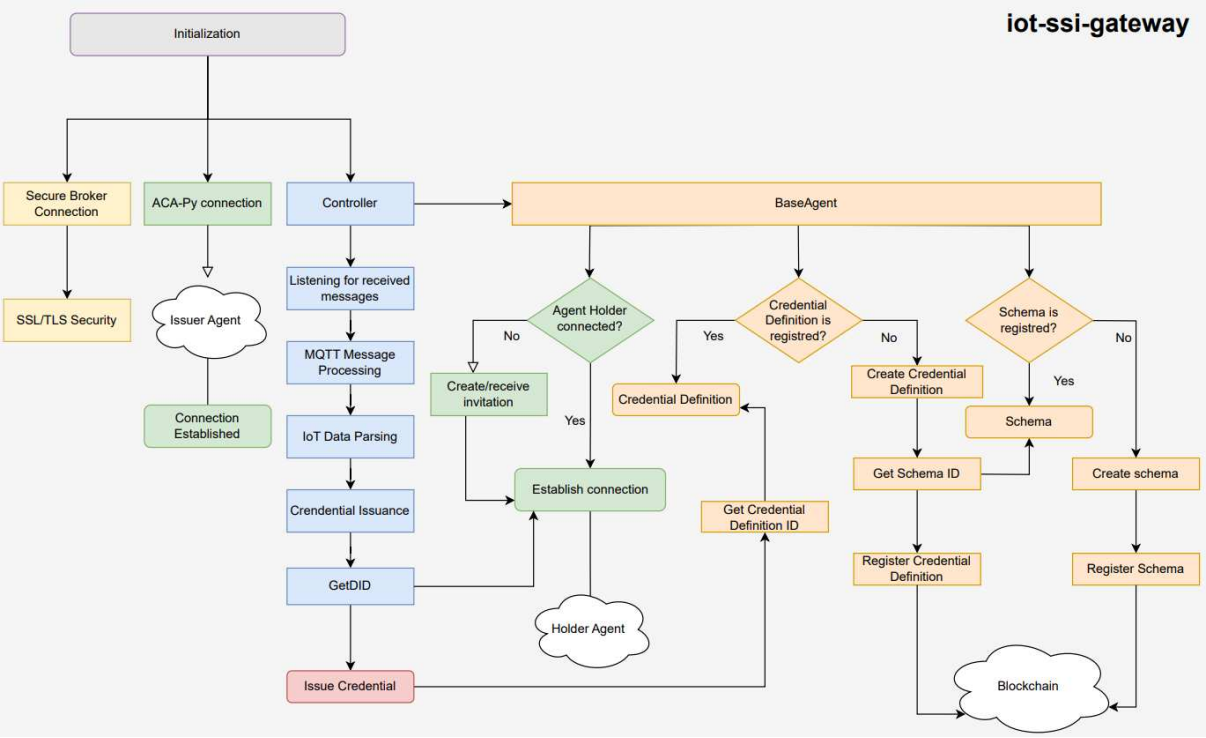
5.2.1 Visão geral da arquitetura

A Figura 19 representa um diagrama de fluxo para o Gateway proposto. O processo inicia com a "Inicialização" e estabelece uma "Conexão segura com o *broker*", utilizando SSL/TLS por meio da biblioteca Bouncy Castle, com certificados digitais autoassinados. O "Agente Emissor" então estabelece a conexão através de uma API segura com o agente ACA-Py.

A através da assinatura no tópico, a aplicação entra no modo de escuta, aguardando as mensagens recebidas, logo que recebida, processa mensagens MQTT, extraindo as informações necessárias no formato Json e então chama o processo para de emitir credenciais.

No lado direito, o "Agente Base", são os demais processos do sistema, verifica se o "Agente Detentor" está conectado e se a "Definição de Credencial" está registrada. Se não estiver, ele cria a Definição de credencial com o "ID do schema" e registra na blockchain. Se o esquema não estiver registrado, o sistema a cria e registra na blockchain. Tanto o a "Definição de credencial" quanto o "schema" devem ser registrados na blockchain. O diagrama apresenta apenas os principais processos e fluxos, o registro de DIDs não é mostrado nesse diagrama, uma vez que faz parte de um estudo adicional da proposta.

Figura 19 – Diagrama de fluxo em alto nível da aplicação.



Fonte: elaborada pelo autor (2023).

5.2.2 Implementação do cliente MQTT

Nos segmentos de código a seguir, é demonstrada a implementação do cliente MQTT para estabelecer a conexão segura com o *broker* e realizar a assinatura no tópico correspondente para receber dados dos dispositivos IoT. O segmento de código que lida com a manipulação de certificados não é apresentado, pois não é relevante para o propósito desta seção.

5.2.2.1 Parâmetros iniciais

O código Java apresentado na listagem 5.5 é destinado à conexão com o *broker* MQTT. Na fase inicial, são estabelecidas variáveis para a configuração, essenciais para a comunicação segura através do protocolo MQTT. A variável `broker` desempenha um papel central ao especificar o endereço do *broker* MQTT. Utilizando o protocolo SSL, ela aponta para o *localhost* na porta 8883.

O tópico MQTT é configurado como “mqtt-iot”, delimitando o canal específico de comunicação no qual o cliente se inscreverá para receber mensagens. O `clientid` é definido como “*iot-ssi-gateway*”, atribuindo uma identificação ao Gateway no contexto do sistema, para distingui-lo de outros clientes conectados ao *broker*.

Os caminhos dos arquivos de certificados e chaves, nas variáveis "`caFilePath`",

"clientCrtFilePath" e "clientKeyFilePath", apontam para os certificados necessários para a construção da conexão segura. O arquivo "ca.crt" contém a autoridade de certificação, enquanto "client.crt" e "client.key" armazenam o certificado e a chave privada do cliente, respectivamente.

A variável "password" tem o papel de armazenar a senha associada ao cliente MQTT. Por fim, o valor de qos (*Quality of Service*) é definido como 0, indicando que a entrega da mensagem não requer confirmação.

```
1  String broker = "ssl://localhost:8883";
2  String topic = "mqtt-iot";
3  String clientid = "iot-ssi-gateway";
4  String caFilePath = "C:\\Mosquitto\\ca.crt";
5  String clientCrtFilePath = "C:\\Mosquitto\\client.crt";
6  String clientKeyFilePath = "C:\\Mosquitto\\client.key";
7  String password = "*******";
8  int qos = 0;
```

Listing 5.5 – Parâmetros de conexão

O trecho de código na listagem 5.6 é responsável pela manipulação de mensagens MQTT recebidas. Inicialmente, um cliente MQTT é instanciado com o endereço do *broker* e o ID do cliente. As opções de conexão, incluindo tempo limite e intervalo de *keep-alive*, são configuradas. Um *callback* é definido para gerenciar eventos MQTT, como perda de conexão no método "connectionLost", recebimento de mensagens "messageArrived" e conclusão da entrega "deliveryComplete". Notavelmente, quando uma mensagem é recebida, o método "onMessageReceived" de um objeto "listener", referenciado no código 5.7, é acionado para processar a mensagem.

```
1  public void processMqttData() throws Exception {
2    MqttClient client = new MqttClient(broker, clientid, new
        MemoryPersistence());
3    MqttConnectOptions options = new MqttConnectOptions();
4    options.setConnectionTimeout(60);
5    options.setKeepAliveInterval(60);
6    options.setSocketFactory(SslUtil.getSocketFactory(caFilePath,
        clientCrtFilePath, clientKeyFilePath, password));
7
8    client.setCallback(new MqttCallback() {
9      public void connectionLost(Throwable cause) {
10       System.out.println("connectionLost: " + cause.getMessage());
11     }
12     public void messageArrived(String topic, MqttMessage message) {
13       listener.onMessageReceived(topic, message);
14     }
```

```
15      public void deliveryComplete(IMqttDeliveryToken token) {
16        System.out.println("deliveryComplete---------" + token.
              isComplete());
17      }
18    });
19    client.connect(options);
20    client.subscribe(topic, qos);
21  }
```

Listing 5.6 – Manipulacao de mensagens recebidas

5.2.3 Processamento de mensagens

O código representado na listagem 5.7 é encarregado de extrair as mensagens MQTT recebidas no sistema. A função é acionada sempre que uma nova mensagem é recebida no tópico, recebendo dois parâmetros: o tópico da mensagem `"topic"` e o objeto `"MqttMessage"`, que contém os dados da mensagem recebida. No início do processo, projetado para receber mensagens MQTT, geralmente em formato JSON, são extraídas informações específicas do `"payload"`. Cada parte dessas informações é então atribuída a variáveis específicas, que incluem o ID do dispositivo, o tipo, a localização e a temperatura. Esses dados extraídos são utilizados para construir um objeto `"IoTDevice"` por meio do método `"assembleIoTData"`.

```
1  public void onMessageReceived(String topic, MqttMessage message) {
2    String messageString = new String(message.getPayload());
3    /* Parse the JSON payload */
4    JSONObject parsedPayload = new JSONObject(new String(message.
         getPayload()));
5    /* Extract individual values */
6    deviceId = parsedPayload.getString("id");
7    deviceType = parsedPayload.getString("type");
8    location = parsedPayload.getString("location");
9    temperature = parsedPayload.getInt("temperature");
10   IoTDevice device = assembleIoTData();
11 }
```

Listing 5.7 – Processamento de mensagem

O código na listagem 5.8 demonstra como a aplicação processa as mensagens MQTT recebidas. Inicialmente, um *timestamp* é gerado, marcando o momento exato em que a mensagem é processada. Os dados extraídos são organizados de maneira estruturada em um objeto `"device"`, uma classe que encapsula os atributos de um dispositivo IoT. Este objeto é preenchido com os valores extraídos da mensagem MQTT. O operador responsável pela coleta dos dados é definido como `"LRG UFSC"`. O

objeto `"device"`, contendo essas informações, é retornado para utilização posterior na arquitetura.

Uma etapa adicional na abordagem do sistema é a parte do código onde uma instância da classe `"DIDController"` é utilizada para obter e atribuir um DID ao dispositivo. Uma *string* combinada é formada pela concatenação dos dados do dispositivo e do *timestamp*. A partir dessa *string*, um *hash* é gerado. Este *hash* é utilizado para garantir a integridade dos dados e será empregado no processo de registro na blockchain para identificar essa leitura de dados, fortalecendo a cadeia de custódia.

```
1  public IoTDataSchema assembleIoTData() throws Exception {
2    IoTDataSchema device = new IoTDataSchema();
3    DIDController controller = new DIDController();
4    Timestamp ts = new Timestamp(System.currentTimeMillis());
5
6    device.setDevice_id(deviceId);
7    device.setDevice_type(deviceType);
8    device.setLocation(location);
9    device.setData(String.valueOf(temperature));
10   device.setTimestamp(ts);
11   device.setOperator("LRG UFSC");
12   device.setDID(controller.GetDidDevice(device.getDID()));
13   String combinedString = deviceId + deviceType + location +
        temperature + ts;
14   String hash_data = HashGenerator.generateHash(combinedString);
15   String did_evidence = controller.createDIDfromEvidence(hash_data)
        ;
16   device.setEmission_did(did_evidence);
17
18   return device;
19 }
```

Listing 5.8 – Processamento de mensagem

Por fim, o controlador é acionado para chamar a função da emissão da credencial. Este procedimento aceita como parâmetros o objeto gerado a partir da mensagem recebida no tópico, juntamente com os atributos relacionados ao dispositivo.

5.2.4 Etapas gerais para gerar e registrar DIDs com Indy SDK

Nesta etapa, demonstra-se o processo que envolve a criação e manipulação de DIDs do tipo *Sovrin* e pares de chaves criptográficas em um ambiente que utiliza a biblioteca *Indy SDK*. Foi empregado o *wrapper* em Java do *Indy SDK* para acessar os recursos da biblioteca. Foram definidos os parâmetros para criação e acesso da *wallet*

e *pool*. Este ambiente realiza a interação com o *ledger* da rede *Von Network*, e utiliza uma carteira para armazenar chaves e outras informações sensíveis.

5.2.4.1 Parâmetros iniciais

O código descrito na listagem A.1 configura uma *wallet* e um *pool* para o sistema. Variáveis são inicializadas, onde "`walletName`" representa o nome da carteira, "`poolName`" o nome do *pool*, "`stewardSeed`" a semente do *steward* e "`poolConfig`" é uma *string* JSON com a configuração do *pool*, contendo a chave "`genesis_txn`" que contém informações iniciais sobre o estado do *ledger* e outras configurações necessárias para iniciar o *pool*.

5.2.4.2 Operações com pool e Wallet

O trecho de código A.2 define as funções para criar, deletar, abrir e fechar conexões com o *pool* local. A função `"openPool"` é usada para abrir o *"pool ledger"* com o nome do *pool* especificado, a fim de conectar os nós da rede *Von Network*. Isso retorna um identificador `"handle"`, que será usado em operações subsequentes. A manipulação do *pool* é essencial para interações com o *ledger*. Operações como criar transações, submeter transações assinadas e realizar consultas no *ledger* geralmente exigem uma conexão ativa com o *pool*.

O código descrito na listagem A.3, define um nome para a carteira `"walletName"` e utiliza a função `"openWallet"` para abrir a carteira, caso ela já exista. Esta operação retorna um identificador `"handle"` para a carteira, que será usado em operações subsequentes. A carteira é fundamental para assinar transações antes de serem submetidas ao *ledger*. No exemplo fornecido, a função "`signAndSubmitRequest`" utiliza a carteira para obter a chave privada necessária para assinar a transação antes de enviá-la ao *ledger*.

5.2.4.3 Geração de DID/Verkey

O código 5.9 demonstra a geração do DID e *Verkey*. Inicialmente, uma semente é utilizada para obter a DID do *Steward* a partir do *ledger*, que já contém essa informação. O DID do *Steward* é essencial, pois é usado para construir transações que serão posteriormente submetidas ao *ledger*. Em seguida, um novo par de *DID/Verkey* é gerado para um "*Trust Anchor*" (entidade responsável por construir transações que serão posteriormente submetidas ao ledger) e armazenado na carteira.

```
1  String did_json = "{\"seed\": \"" + stewardSeed + "\"}";
2  DidResults.CreateAndStoreMyDidResult stewardResult = Did.
      createAndStoreMyDid(walletHandle, did_json).get();
3  String defaultStewardDid = stewardResult.getDid();
```

```
4
5   DidResults.CreateAndStoreMyDidResult trustAnchorResult = Did.
        createAndStoreMyDid(walletHandle, "{}").get();
6   String trustAnchorDID = trustAnchorResult.getDid();
7   String trustAnchorVerkey = trustAnchorResult.getVerkey();
```

Listing 5.9 – Criação de DIDs de Steward e Trust Anchor em Java

5.2.4.4 Construção de uma transação e registro no blockchain

O próximo passo é a construção de uma transação do tipo NYM utilizando o DID do *Steward*. Essa transação tem como objetivo adicionar o DID e o *Verkey* do *Trust Anchor* ao *ledger*, atribuindo a ele a função de "*Trust Anchor*". A transação é assinada com a chave privada correspondente à DID do *Steward*. Esse processo é descrito no código 5.10.

```
1   String nymRequest = buildNymRequest(defaultStewardDid,
        trustAnchorDID, trustAnchorVerkey, "TRUST_ANCHOR").get();
2   String nymResponseJson = signAndSubmitRequest(pool, walletHandle,
         defaultStewardDid, nymRequest).get();
```

Listing 5.10 – Geracao de dids

Após a transação *NYM* ser bem-sucedida e os dados do *Trust Anchor* serem registrados no *ledger*, o código gera um novo par de *DID/Verkey* para representar um cliente. Essas novas identidades do cliente são utilizadas para criar uma solicitação do tipo *GET_NYM*, que é uma consulta ao *ledger* para confirmar o *Verkey* do *Trust Anchor*. Vale ressaltar que, para simplificar, o código utiliza uma única carteira. No entanto, em um cenário real, seriam utilizadas três carteiras diferentes, e os DIDs seriam trocadas por meio de algum canal de comunicação seguro (INDY, 2022).

5.2.5 Controlador ACA-Py

A seguir, serão destacados os principais atributos do controlador do agente em nuvem, acompanhados do código correspondente. Os fluxos e interações entre os agentes foram configurados em conformidade com a documentação do ACA-Py (HYPERLEDGER, 2023a). É importante destacar que o agente ACA-Py deve ter obtido previamente o bloco gênesis para ingressar na rede e registrado a semente para adquirir um DID público. Este DID já foi registrado antecipadamente no ledger, conforme evidenciado na Subseção 5.1.2.1.

Embora a implementação abranja vários outros recursos, serão apresentados os mais relevantes, que incluem:

- A instanciação do agente emissor.

- A conexão com o agente detentor das credenciais.
- A criação de esquemas.
- As definições de credenciais.
- A emissão de credenciais.

Os três últimos recursos são estruturados de acordo com os atributos associados aos dispositivos IoT, considerando a necessidade da cadeia de custódia de dados. O controlador compreende funções projetadas para executar as operações esboçadas no diagrama de sequência, conforme ilustrado na Figura 15. O controlador interage com o agente de nuvem por meio de chamadas API e, para facilitar essa integração, foi empregada a biblioteca cliente ACA-Py em Java. Todos os recursos implementados nos códigos nas subseções subsequentes utilizam essa biblioteca.

5.2.5.1 Conexão com Agente em nuvem

Para criar uma instância do Agente em nuvem, que atua como emissor na arquitetura proposta, a conexão é configurada utilizando o código 5.11.

```
1  public void createConnection() throws IOException {
2    ariesclient = AriesClient
3    .builder()
4    .url("http://localhost:8010")
5    .apiKey("secret")
6    .bearerToken("123.456.789")
7    .build();
8    String did = ariesclient.walletDidPublic().get().getDid();
9    System.out.print("Public DID information: " + did);
10 }
```

Listing 5.11 – Instancia do agente em nuvem

Neste segmento de código, a conexão é estabelecida. O *container* que hospeda o agente emissor está em execução no endereço local na porta 8010, fornecendo o *endpoint* de conexão, a chave de API (*apiKey*) e o *token* de autenticação (*bearerToken*). Após a conexão ser estabelecida com sucesso, o DID público, registrado na blockchain, é obtido e exibido.

5.2.5.2 Estabelecer comunicação com Agentes através de convites

Antes de explorar a implementação da conexão entre agentes, é essencial entender as formas de conexão atualmente suportadas pelo Hyperledger Aries, que ocorrem por meio de convites públicos e não públicos.

Um convite público contém um DID para que outros agentes se conectem. Para um convite público, o *ledger* precisa saber em qual *endpoint* um agente pode ser al-

cançado. Isso implica que uma consulta ao *ledger* é necessária pelo agente convidado. Por outro lado, um convite não público não usa um DID público; em vez disso, contém uma URL de serviço de terminal, permitindo que o agente convidado se conecte diretamente ao autor do convite (CURRAN; HOWARD, 2021b).

Nos testes realizados, foram utilizados convites não públicos, seguindo a documentação oficial do ACA-Py. Os agentes se comunicam usando mensagens DIDComm e se conectam trocando DIDs e DIDDocs em pares com base no *did:peer* (HYPERLEDGER, 2023a).

A seguir, será apresentado o processo de estabelecimento de uma conexão por meio do envio e recebimento de convites, onde uma das partes tem a capacidade de criar o convite, enquanto a outra tem a capacidade de aceitá-lo.

5.2.5.2.1 Conexão através do recebimento de convite

A primeira forma de estabelecer uma conexão ocorre através do recebimento de um convite, no qual outro agente cria um convite contendo informações e parâmetros para a conexão, como *endpoint*, chave pública, versão do protocolo para conexão, entre outros.

Inicialmente, conforme o código na lista 5.12, o método `"receiveInvitation"` recebe dois parâmetros: um convite representado por uma *string* `"invitation"` e uma chave pública associada ao convite `"key"`. Os parâmetros do convite, como tipo, identificador, rótulo, *endpoint* de serviço, chaves de destinatário, entre outros, são definidos e organizados para construir um objeto `"ReceiveInvitationRequest"`. Este objeto é criado usando o padrão `"Builder"`, facilitando a construção de objetos complexos.

Adicionalmente, um filtro `"ConnectionReceiveInvitationFilter"` é definido para controlar o comportamento da conexão ao receber o convite. O filtro inclui um alias, uma indicação de aceitação automática da conexão `"autoAccept"`, e um identificador de mediação. O método `"connectionsReceiveInvitation"` é então chamado, utilizando o objeto `"ReceiveInvitationRequest"` e o filtro definido anteriormente. Este método processa o convite e retorna um objeto `"Optional<ConnectionRecord>"`, encapsulando a possível conexão estabelecida. A estrutura `"Optional"` é utilizada para lidar com a possibilidade de ausência de resposta ou de um resultado nulo.

```
1 public void receiveInvitation(String invitation, String key) {
2   String invitationType = "https://didcomm.org/connections/1.0/
      invitation";
3   String invitationId = invitation;
4   String invitationLabel = "alice.agent";
5   String serviceEndpoint = "http://host.docker.internal:8030";
6   String[] keys = {key};
7   List<String> recipientKeys = Arrays.asList(keys);
```

```
8   List<String> routingKeys = Arrays.asList("");
9   String did = "";
10  String imageUrl = "";
11  String alias = "FABER_INVITE";
12  Boolean autoAccept = true;
13  String mediationId = "";
14
15  ReceiveInvitationRequest invite = ReceiveInvitationRequest
16    .builder()
17    .type(invitationType)
18    .id(invitationId)
19    .recipientKeys(recipientKeys)
20    .label(invitationLabel)
21    .serviceEndpoint(serviceEndpoint)
22    .build();
23
24  ConnectionReceiveInvitationFilter filter =
       ConnectionReceiveInvitationFilter
25    .builder()
26    .alias(alias)
27    .autoAccept(autoAccept)
28    .mediationId(mediationId)
29    .build();
30    Optional<ConnectionRecord> connectionRecord = ariesClient.
         connectionsReceiveInvitation(invite, filter);
31    ConnectionRecord connRecordStatus = connectionRecord.get();
32  }
```

Listing 5.12 – Conexão através do recebimento de convite

Por fim, é extraído o objeto `"ConnectionRecord"` resultante, representando a conexão bem-sucedida. Esta conexão pode ser posteriormente utilizada para interações descentralizadas no contexto do Hyperledger Aries.

5.2.5.2.2 Conexão através da criação convite

A segunda maneira de estabelecer uma conexão é enviando um convite que contém detalhes e parâmetros para a conexão, como *endpoint*, chave pública, versão do protocolo, entre outros.

Conforme mostrado no código da listagem 5.13, o método `"createInvitation"` inicialmente define parâmetros internos essenciais para a criação do convite. O rótulo associado à conexão, denominado `"label"`, é estabelecido como "*Alice connection*".

Além disso, é especificada uma lista, `"recipientKeys"`, contendo a chave pública do destinatário.

A construção do objeto `"CreateInvitationRequest"` é realizada utilizando o padrão `builder`. Esse objeto encapsula as informações necessárias para criar o convite. O rótulo da conexão `"myLabel"` e a lista de chaves públicas dos destinatários `"recipientKeys"` são configurados. Em seguida, o método invoca a função `"connectionsCreateInvitation"`, que é responsável por criar o convite de conexão. O objeto `"invitationRequest"` é passado como parâmetro. O resultado dessa operação é encapsulado em um objeto `"Optional<CreateInvitationResponse>"`.

O resultado é armazenado na variável `"connectionInvitation"`. Este objeto opcional contém a resposta da criação do convite.

```
1  public void createInvitation() throws IOException {
2    String label = "Alice connection";
3    List<String> recipientKeys = Arrays.asList(
4    "EYymRn6QZtsWkgN8KjHnsaG4Jum4zAUN4x4NXv2fJYgg");
5
6    CreateInvitationRequest invitationRequest =
        CreateInvitationRequest
7    .builder()
8    .myLabel(label)
9    .recipientKeys(recipientKeys)
10   .build();
11   final Optional<CreateInvitationResponse> connInvite = ariesclient
        .connectionsCreateInvitation(invitationRequest);
12 }
```

Listing 5.13 – Conexão através da criação de convite

O código da listagem 5.14 estende a funcionalidade anterior, extraindo informações específicas do convite de conexão e exibindo-as em formato JSON no console. Isso proporciona uma maneira clara de visualizar e compartilhar os detalhes essenciais do convite gerado.

```
1  Object jsonObject = new Object() {
2    String inviteType=connInvite.get().getInvitation().getAtType();
3    String connId=connInvite.get().getConnectionId();
4    String inviteLabel=connInvite.get().getInvitation().getLabel();
5    String serviceEndpoint=connInvite.get().getInvitation().
        getServiceEndpoint();
6    String recipientKey = connInvite.get().getInvitation().
        getRecipientKeys().get(0);
7    String inviteUrl = connInvite.get().getInvitationUrl();
8  };
```

```
9     System.out.println("Invitation: " + jsonObject);
10 }
```

Listing 5.14 – Formatar convite em Json

5.2.5.3 Criação de schema

O trecho de código na listagem 5.15 e 5.16 ilustra o processo de criação de um esquema. Parâmetros como o nome, a versão e o DID associado ao esquema são definidos. O método inicializa, então, variáveis locais, sendo `"attributes"` uma lista de atributos do esquema obtida através da função `"attributesAddSchema"`, descrita na listagem do código 5.16. A função `"builder"` é chamada para criar um objeto `"SchemaSendRequest"`, fundamental na definição do esquema. Este objeto é construído com o nome, a versão e os atributos do esquema.

A etapa seguinte envolve a invocação do método `"schemas"`, linha 14, responsável por enviar a definição do esquema para a blockchain. O objeto `"schemaRequest"` é utilizado como parâmetro, fornecendo as informações necessárias. A resposta dessa operação é encapsulada em um objeto `"Optional<SchemaSendResponse>"`.

O resultado é então armazenado em `"schemaResponse"`. Por fim, o método retorna o ID do esquema convertido para uma *string* por meio de `"schemaResponse.get"`. Esse ID será posteriormente usado na criação da definição do modelo de credencial.

```
1  public String schemas() throws IOException {
2    String schemaName = name;
3    String schemaVersion = version;
4    String schemaDid = did;
5    List<String> attributes = attributesAddSchema();
6    SchemaSendRequest schemaRequest = SchemaSendRequest
7    .builder()
8    .schemaName(schemaName)
9    .schemaVersion(schemaVersion)
10   .attributes(attributes)
11   .build();
12
13   Optional<SchemaSendResponse> schemaResponse = ariesclient.schemas
        (schemaRequest);
14   return schemaResponse.get().getSchemaId().toString(); }
```

Listing 5.15 – Definir Schema

A função delineada na listagem código 5.16 tem como propósito a criação e retorno de uma lista de strings que representam os atributos associados a um esquema. Cada atributo é adicionado à lista por meio do método `"add"` da classe `"ArrayList"`.

Os atributos incluídos nessa lista desempenham papéis específicos no contexto da gestão de dados de dispositivos IoT, como o identificador que será empregado para distinguir cada dispositivo; o DID utilizado para identificar cada dispositivo; o tipo de dispositivo associado; o *hash* de um dado que não pode ser diretamente incorporado à credencial, como mencionado na Subseção 4.3.2.3; os dados propriamente ditos, que podem incluir informações como leituras de sensores e são passíveis de serem inseridos na credencial; a inclusão opcional do DID de cada coleta, registrado na blockchain; a localização do dispositivo e o indicando o momento em que os dados foram emitidos.

```
1  private static List<String> attributesAddSchema() {
2    List<String> attributes = new ArrayList<>();
3    attributes.add("id");
4    attributes.add("device_did");
5    attributes.add("device_type");
6    attributes.add("hash_data")
7    attributes.add("data");
8    attributes.add("collection_did");
9    attributes.add("location");
10   attributes.add("timestamp");
11   return attributes;
12 }
```

Listing 5.16 – Adiciona atributos no Schema

No processo de emissão da credencial, esses campos são empregados para atribuir os valores correspondentes. Em situações em que a credencial contenha atributos distintos do *schema*, um erro será gerado. Se a credencial apresentar outros campos, será necessário criar um novo *schema* e uma definição de credencial com base nesse esquema. Esse procedimento assegura a integridade e consistência das credenciais emitidas em conformidade com os esquemas previamente estabelecidos.

5.2.5.4 Definição do modelo de credencial

O código 5.17 tem a função de criar uma definição de credencial vinculada a um esquema específico. Ele recebe como entrada o identificador do esquema ao qual a credencial será associada. Com esse ID em mãos, inicia-se a criação do objeto.

No processo de construção, o código define o `"schemaId"`, que é o identificador do esquema associado à credencial, e adiciona uma "tag" à credencial, neste caso, denominada `"evidence"`. Com o objeto de requisição `"request"` devidamente preparado, o método `"credentialDefinitionsCreate"` é acionado. Esse passo é responsável por criar a definição da credencial e registrá-la na blockchain.

A chamada ao método mencionado utiliza o objeto `"request"` como parâmetro, fornecendo todas as informações necessárias. Como resultado, obtemos um objeto, encapsulando a resposta da operação.

Finalmente, para acessar o resultado da criação da definição de credencial, recorremos à obtenção do ID associado a ela. Isso é alcançado através da chamada do método `"getCredentialDefinitionId"`. Esse ID torna-se um requisito essencial para o processo de emissão da credencial.

```
1 public void defineCredential(String schemaId) throws IOException {
2   CredentialDefinition.CredentialDefinitionRequest request =
       CredentialDefinition.CredentialDefinitionRequest
3   .builder()
4   .schemaId(schemaId)
5   .tag("evidence")
6   .build();
7   Optional<CredentialDefinition.CredentialDefinitionResponse>
       credentialDefinition = ariesclient.credentialDefinitionsCreate
       (request);
8   String definitionId = credentialDefinition.get().
       getCredentialDefinitionId();
9 }
```

Listing 5.17 – Cria definição de credencial

5.2.5.5 Emissão da credencial verificável

A função do código na listagem 5.18 necessita de três parâmetros: um objeto `"IoTDevice"` que contém informações sobre um dispositivo IoT, uma *string* `"connId"` representando a identificação da conexão com o agente representado por Alice, e uma string `"credDefId"` que especifica o identificador da definição de credencial a ser utilizada que foi criada anteriormente.

No início da função, um mapa chamado é criado para armazenar os atributos que irão compor a credencial. Os valores desses atributos são extraídos do objeto `"device"`, que contém informações específicas do dispositivo IoT incluindo o dado coletado, tais como operador, DID que identifica o dispositivo, DID que identifica a coleta, tipo de dispositivo, dado coletado, localização e carimbo de data/hora.

Após o preenchimento do mapa, uma lista de objetos `"CredentialAttributes"` é criada. Essa lista representa os atributos que irão compor a credencial. Em seguida, uma instância de `"CredentialPreview"` é criada utilizando a lista de atributos criada anteriormente `"credPrev"`. Posteriormente, são inicializadas duas *strings*, com os parâmetros recebidos `"connectionId"` e `"credentialdefinitionId"`. Um booleano `"autoIssue"` é definido como `"true"` para que a credencial seja emitida.

```
1  public void issueCredential(IotDevice device, String connId, String
       credDefId)throws IOException{
2    Map<String, String> attributeValues = new HashMap<>();
3    attributeValues.put("id", device.getDevice_id());
4    attributeValues.put("Operator", device.getOperator());
5    attributeValues.put("device_did", device.getDID());
6    attributeValues.put("collection_did", device.getCollect_did());
7    attributeValues.put("device_type", device.getDevice_type());
8    attributeValues.put("data", device.getData());
9    attributeValues.put("location", device.getLocation());
10   attributeValues.put("timestamp", device.getTimestamp());
11   List<CredentialAttributes> attributes = CredentialAttributes.
       fromMap(attributeValues);
12   CredentialPreview credPrev = new CredentialPreview(attributes);
13   String connectionId = connId;
14   String credentialdefinitionId = credDefId;
15   Boolean autoIssue= true;
16
17   Optional<V1CredentialExchange> response = baseAgent.
       issueCredentialSend(V1CredentialProposalRequest
18   .builder()
19   .connectionId(connectionId)
20   .credentialDefinitionId(credentialdefinitionId)
21   .autoRemove(true)
22   .credentialProposal(credPrev)
23   .build());
24 }
```

Listing 5.18 – criar uma prévia da credencial

A credencial é enviada chamando o método `"issueCredentialSend"` da instancia do agente de nuvem "baseAgent". Esta chamada envolve a construção de uma requisição de proposta de credencial `"V1CredentialProposalRequest"` com informações como ID de conexão, ID da definição da credencial e a prévia da credencial `"credPrev"` que foi montada com os atributos do dispositivo IoT e a coleta de dados. O resultado é armazenado no objeto `"response"`.

5.2.5.6 Apresentação da credencial

O código da listagem 5.19 facilita a criação de uma prévia de credencial, seguindo o conceito de prova de conhecimento zero. Esse procedimento envolve a construção de um pedido de prova específico, que é elaborado com base em uma definição de credencial fornecida. Para efetivar esse pedido, o código inicia com a preparação

de uma coleção de atributos de prova, utilizando métodos desenvolvidos para impor restrições específicas, tanto de forma individual quanto coletiva, a esses atributos.

```
1  public void CreatePresentProof(String connectionId, String
       credentialDefinitionId, Device device) throws IOException {
2    if (this.attributes == null) {
3      this.attributes = new HashMap<>();
4    }
5    /* Restringe dados ao dispositivo com identificador especifico */
6    addAttributeRestriction(new AttributeRestriction("device_did",
       device.getDID()));
7    /* Restringe dados coletados em uma faixa de tempo especifica */
8    String startTime = "2024-01-01T00:00:00Z";
9    String endTime = "2024-01-31T23:59:59Z";
10   addAttributeRestriction(new AttributeRestriction("timestamp", "
       between " + startTime + " and " + endTime));
11   /* Restringe dados a uma localizacao especifica */
12   addAttributeRestriction(new AttributeRestriction("location",
       device.getLocation()));
13   /* Restringe dados ao tipo de dispositivo */
14   addAttributeRestriction(new AttributeRestriction("device_type",
       device.getDevice_type()));
15   /* Cria o ProofRequest com os atributos */
16   ProofRequest proofRequest = getProofRequest();
17   PresentProofRequest presentProofRequest = PresentProofRequest
18   .builder()
19   .connectionId(connectionId)
20   .proofRequest(proofRequest)
21   .build();
22   /* Envia o pedido de prova */
23   ac.presentProofSendRequest(presentProofRequest);
24   System.out.println("Pedido de prova enviado para " + connectionId
       );
25 }
```

Listing 5.19 – Criar uma prévia da credencial

O processo começa com a verificação da existência de uma coleção de atributos; na ausência desta, uma nova é criada. A seguir, restrições são aplicadas a cada atributo necessário para a composição da prova. Tais restrições podem envolver especificidades do dispositivo IoT, como identificador do dispositivo, tipo de dispositivo, localização e intervalos de tempo, assegurando que os dados atendam aos critérios exigidos pela entidade solicitante.

Após definir as restrições necessárias, o método `"getProofRequest"` é utilizado para compilar um objeto `"ProofRequest"`, que encapsula a solicitação de prova. Esse objeto incorpora todos os atributos anteriormente configurados, juntamente com suas restrições, estabelecendo as bases da solicitação de prova ao definir precisamente quais dados são requeridos e as condições para sua solicitação. Com o `"ProofRequest"` estruturado, o objeto `"PresentProofRequest"`, é criado utilizando o `"connectionId"` e o `"ProofRequest"` como parâmetros. Este objeto concretiza a solicitação completa da prova de conhecimento zero, pronta para ser enviada ao destinatário.

Por fim, o método `"presentProofSendRequest"`, envia o `"PresentProofRequest"` ao destinatário designado. Essa ação desencadeia o processo de verificação de credenciais, fundamentado nos atributos específicos e restrições definidas.

5.3 CONSIDERAÇÕES FINAIS

Este capítulo detalhou as etapas gerais na criação da rede, com o objetivo de proporcionar um entendimento abrangente do desenvolvimento realizado. A implementação de uma rede Hyperledger Indy, ancorada na infraestrutura da rede *Von Network* e utilizando agentes Aries como interface, não apenas facilitou a interação dentro da rede, mas também estabeleceu a proposta de uma ponte segura entre dispositivos IoT e entidades legais. O capítulo busca fornecer uma visão holística do processo, abrangendo desde a criação e registro de DIDs até a elaboração de esquemas, a definição de credenciais e a emissão de credenciais verificáveis. A exposição desses dados, especialmente quando a privacidade é um requisito, pode ser efetuada mediante a apresentação de prova, através da divulgação seletiva de dados pelo detentor para as demais partes interessadas. A convergência dessas tecnologias não somente garante a integridade e autenticidade dos dados coletados, mas também reforça a cadeia de custódia digital em contextos do domínio forense.

6 RESULTADOS E DISCUSSÃO

Este capítulo detalha os experimentos realizados, assim como os resultados obtidos, seguidos por uma análise e discussão. Inicialmente, apresenta-se uma descrição dos objetivos perseguidos e da metodologia adotada para a condução dos experimentos. Posteriormente, exibem-se os resultados, evidenciando os desdobramentos das três fases distintas do experimento. Cada etapa é analisada e discutida, visando proporcionar uma compreensão integral do estudo realizado.

6.1 EXPERIMENTOS

Esta seção apresenta uma descrição detalhada dos objetivos e da metodologia empregada nos experimentos realizados. O principal propósito destes experimentos foi avaliar a eficiência do Gateway no processamento de dados, além do registro de DIDs e da emissão de VCs.

6.1.1 Objetivos do design do experimento

Este experimento foi projetado para responder a perguntas de pesquisa sobre o desempenho e escalabilidade de uma arquitetura para certificação de dados em redes IoT na origem. O objetivo principal é avaliar a capacidade da arquitetura desenvolvida de lidar com altas cargas de dados, tanto em termos de frequência de publicação quanto da quantidade de sensores enviando dados simultaneamente. As métricas utilizadas nos experimentos desempenham um papel importante na avaliação do desempenho e escalabilidade do sistema, facilitando a identificação de gargalos e a proposição de soluções para otimização.

6.1.1.1 Ciclo do experimento

O experimento iniciou-se com a implementação de sensores de temperatura, os quais foram criados utilizando o Node-RED para fins de simulação. Esses sensores são programados para publicar dados em um tópico específico do *broker* Mosquitto, incluindo informações de temperatura, localização e identificação do sensor, formatadas em JSON. O Gateway é configurado para assinar o tópico designado, habilitando-o a receber os dados transmitidos pelos sensores. Para garantir uma conexão segura, certificados SSL *Self-Signed* foram aplicados tanto no Node-RED quanto no Gateway.

O processamento dos dados pelo Gateway envolve a extração de informações do formato JSON, seguido pelo encapsulamento desses dados em uma credencial. Mediante a uma conexão pré-estabelecida, juntamente com esquemas e modelos de credenciais definidos previamente, o sistema é capaz de emitir automaticamente credenciais verificáveis a um agente representativo de uma entidade autorizada. Essa

entidade pode variar entre autoridades judiciárias, órgãos de fiscalização ou instituições forenses, conforme discutido na Subseção 4.2.1.

Além disso, o experimento explora a possibilidade de identificar tanto as publicações quanto os dispositivos dentro da arquitetura, empregando identificadores descentralizados utilizando o método “did:sov”. Este processo tem como objetivo principal analisar a viabilidade do uso deste mecanismo como uma cadeia de custódia de dados. O experimento também investiga se a frequência das transações relacionadas ao registro na blockchain nesse experimento, pode se tornar um fator limitante e se poderá exigir mais pares para manter a rede operacional.

Por último, o experimento analisa a frequência com que as credenciais podem ser emitidas e como essa etapa fundamental da arquitetura se comporta com diferentes níveis de requisições. Para alcançar esses objetivos, foram testadas diferentes configurações da Von Network e dos agentes. Uma vez emitidas as credenciais, a aplicação entra em um estado de "modo de espera". Este estado permite que a aplicação conserve recursos enquanto aguarda novas requisições.

6.1.1.2 Fases do experimento

Foram realizados três testes independentes que buscam analisar os principais processos do sistema: o processamento da mensagem recebida do *broker*, a geração de DIDs para identificar cada mensagem juntamente com o dispositivo que publicou e seus atributos, e por fim, a geração e emissão das credenciais verificáveis. Além disso, as avaliações abrangeram o uso de memória RAM e taxa de ocupação da CPU em todas as fases dos testes.

No primeiro teste, o tempo médio de resposta para o processamento de cada publicação no *broker* e a montagem do pacote que é criado para gerar os atributos que irão compor a credencial. No segundo teste, buscou-se analisar o tempo médio para criação e registros de DIDs na blockchain para identificação das publicações. Por fim, o último teste, focou na emissão de credenciais verificáveis.

O primeiro teste avaliou o tempo médio de resposta para processar cada publicação no *broker*, assim como o tempo necessário para montar o pacote que contém os atributos para a criação da proposta das VCs. No segundo teste, o foco recaiu sobre a análise do tempo médio necessário para criar e registrar DIDs na blockchain, visando à identificação das publicações recebidas no *broker*. Por fim, o terceiro teste concentrou-se na emissão das VCs, considerando o tempo que a aplicação e o agente levam para criar e enviar a credencial. Vale ressaltar que o tempo que o agente "Holder" leva para receber e armazenar as VCs não é considerado neste experimento.

6.1.1.3 Adaptações para a realização do experimento

Para suportar experimentos em uma escala ampliada, foram necessárias adaptações na aplicação desenvolvida. Uma API foi incorporada para permitir requisições em larga escala, por aplicações de teste de carga e medição de desempenho. Isso possibilitou a mensuração do tempo de resposta e o consumo de recursos em diversas fases do processo. As APIs foram criadas para interagir com os principais processos da aplicação, descritos na Subseção 5.2.5. Isso inclui publicar e obter o estado de conexões, DIDs, esquemas, definições de credenciais e credenciais emitidas.

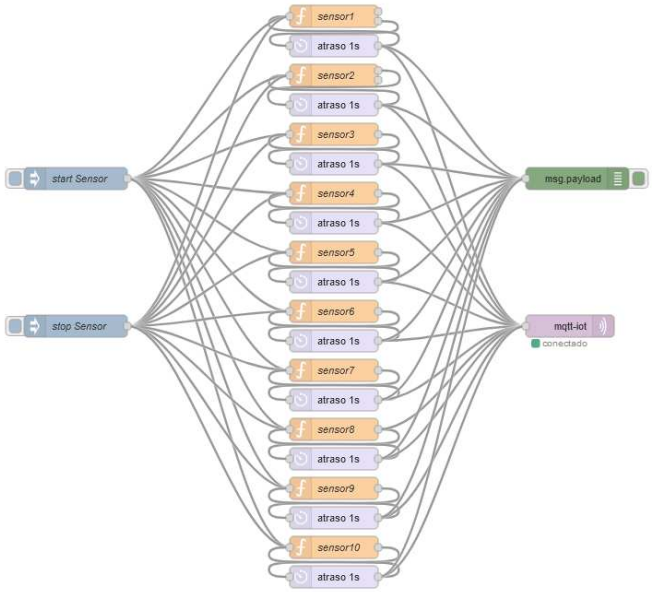
6.1.2 Implementação do experimento

Para a realização dos experimentos, utilizou-se um único sistema computacional, equipado com um processador Intel Core i7-11700 e 16 GB de RAM, executando o sistema operacional Windows 11 Pro.

O software Node-RED foi utilizado para simular sensores de temperatura. Inicialmente, o sistema foi configurado para simular 10 sensores, os quais publicavam mensagens a cada segundo. Posteriormente, a configuração foi expandida para incluir mil sensores, mantendo o intervalo de publicação de dados no *broker* em um segundo.

A Figura 20 ilustra, através de um diagrama do Node-RED, a configuração de uma rede de sensores de temperatura, onde cada sensor possui um atraso de um segundo antes de enviar dados para um *broker* MQTT, facilitando a gestão e análise centralizada dos dados coletados. O sistema permite o início e o término

Figura 20 – Fluxo de automação de sensores de temperatura no Node-RED.



Fonte: elaborada pelo autor (2023).

do monitoramento através de comandos específicos. O nó "msg.log" é utilizado para acompanhar as publicações de temperatura. Posteriormente, essas informações são encaminhadas ao *broker* MQTT através do nó "mqtt-iot". A interconexão e o trajeto dos dados dentro da simulação são evidenciados pelas linhas que conectam os nós, demonstrando o percurso que os dados seguem.

O estudo centrou-se na implementação de dois agentes: o agente emissor e o agente detentor. O agente emissor é responsável pela geração e emissão de credenciais, interação com a blockchain, e estabelecimento de comunicações com diferentes domínios. O agente detentor, por sua vez, é usado para interagir com o emissor e receber as credenciais, permitindo fluxo contínuo de interações nas trocas de informação.

Para a implementação da aplicação, foram utilizadas bibliotecas Java. As entidades que interagem com a aplicação foram acomodadas em contêineres usando Docker e Docker Compose. A lista das tecnologias empregadas neste experimento pode ser visualizada no quadro 2.

A obtenção das métricas foi realizada utilizando a biblioteca Apache Log4j e o aplicativo Apache Jmeter, ferramentas que permitiram um monitoramento preciso. Além disso, a utilização dos recursos de CPU e memória foi conduzido com o auxílio do Netbeans Profiler, permitindo uma avaliação detalhada do consumo de recursos durante os testes. A carga de dados inicial foi gerada através de requisições HTTP POST utilizando o Apache JMeter. Para uma representação mais próxima de um ambiente IoT real, os testes subsequentes foram conduzidos com o protocolo MQTT, que é o padrão predominante neste contexto com o Gateway registrando o início e término de cada etapa do processamento.

6.2 RESULTADOS

Nesta seção, os resultados são dissecados em três etapas distintas do experimento. É importante ressaltar que a latência de rede não foi incorporada nesta análise, visto que todas as interações ocorreram dentro de um ambiente controlado e centralizado em um único sistema computacional. O estudo focou na aferição do tempo médio de resposta de cada operação e do *throughput*.

O *throughput* foi calculado levando em conta o tempo de resposta da aplicação para processar as operações, e é expresso em operações por segundo (ops/s). A fórmula utilizada para determinar o *throughput* é a seguinte:

$$\text{Throughput (ops/s)} = \frac{\text{Número de mensagens}}{\text{Tempo de resposta em milissegundos x 0.001}} \quad (1)$$

Adicionalmente, procedeu-se à avaliação do consumo de recursos do sistema, tais como o uso de memória RAM e taxa de ocupação de CPU. As diferentes fases do

experimento iluminam aspectos variados do sistema, proporcionando uma compreensão ampla sobre seu desempenho.

6.2.1 Fase 1: Capacidade de processamento de dados do gateway

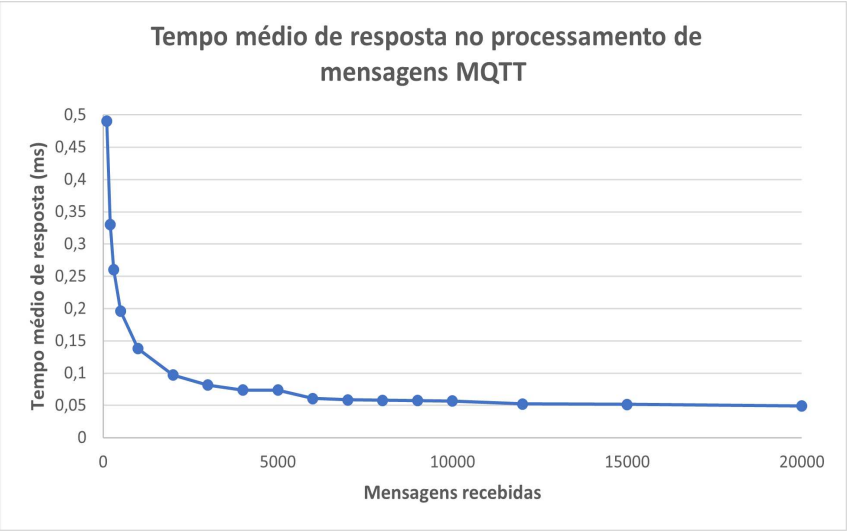
Nesta fase do experimento, avaliou-se a eficiência do Gateway no processamento de dados recebidos através da subscrição de um tópico específico, onde sensores estão ativamente publicando mensagens. Para compreender o desempenho do sistema nesta fase, realizaram-se dois testes de carga. O principal objetivo desses testes foi examinar o comportamento da aplicação sob condições variadas de demanda, abrangendo desde níveis baixos até cenários de alta intensidade. As amostras foram coletadas até que se atingisse um padrão assintótico.

6.2.1.1 Etapa 1: Teste de carga com baixa demanda

O tempo médio de resposta do Gateway, utilizado como indicador de desempenho para o processamento de mensagens, foi medido considerando a frequência de publicação de dados por 10 sensores a cada segundo, ao longo de um experimento com duração de 30 minutos. O total de dados coletados foi de aproximadamente 30 mil mensagens, sendo realizadas três rodadas de testes e gerada uma média do tempo de resposta. As mensagens foram observadas até que um padrão estável fosse alcançado, o que ocorreu após 20 mil mensagens.

A Figura 21 apresenta o gráfico que descreve a relação entre o volume acumulado de mensagens MQTT recebidas e o tempo médio de resposta do Gateway.

Figura 21 – Tempo médio de resposta na primeira etapa da fase 1 do experimento.



Fonte: elaborada pelo autor (2023).

O gráfico demonstra que após o processamento das primeiras mensagens, o sistema rapidamente se ajusta, alcançando uma estabilidade no tempo de resposta que se mantém estável no decorrer das mensagens recebidas.

Os resultados revelam que, após o processamento das primeiras 100 mensagens, o tempo médio inicial para o processamento de cada mensagem foi de 0,49 milissegundos (ms). Houve uma diminuição progressiva desse intervalo, atingindo 0,13 ms após o processamento de 1.000 mensagens e 0,056 ms após 10.000 mensagens. Após o processamento de 15.000 mensagens, o tempo de resposta diminuiu para 0,050 ms, estabilizando-se em 0,049 ms após o processamento de 20.000 mensagens. Este padrão pode sugerir um ponto em que a capacidade de cache e a otimização das conexões atingem seu máximo potencial, resultando em uma resposta estável, mesmo diante de um aumento contínuo no volume de dados processados.

A Tabela 3 apresenta os resultados obtidos nessa fase do experimento, detalhando a quantidade de mensagens processadas, o tempo médio de processamento por mensagem e o *throughput*, expresso em operações por segundo (ops/s), para diferentes volumes de mensagens.

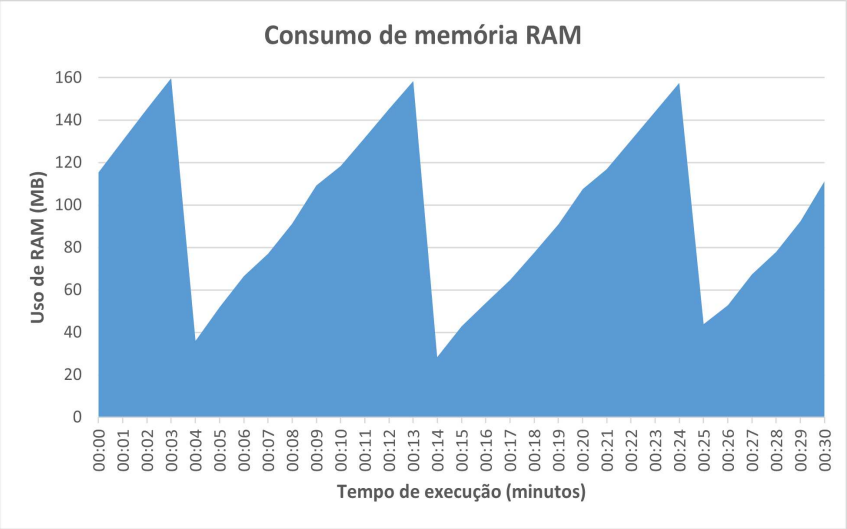
Tabela 3 – Resultados na primeira etapa da fase 1 do experimento.

Test	Mensagens	Tempo médio	Throughput
MQTT	100	0,49 ms	2.040 ops/s
	1000	0,13 ms	7.246 ops/s
	10.000	0,056 ms	17.699 ops/s
	15.000	0,050 ms	20.000 ops/s
	20.000	0,049 ms	20.408 ops/s
	30.000	0,049 ms	20.408 ops/s

O *throughput*, inicialmente avaliado com 100 operações, registrou um valor de 2.040 operações por segundo (ops/s). Ao ampliar a escala para 1.000 operações, observou-se um aumento significativo, com o *throughput* elevando-se para 7.246 ops/s. Com o processamento de 10.000 operações, o *throughput* atingiu o patamar de 17.699 ops/s. Por fim, ao lidar com 20.000 operações, o *throughput* alcançou 20.408 ops/s, mantendo-se estável após o processamento de 30.000 operações, evidenciando um incremento proporcional à quantidade de operações processadas.

Na Figura 22, o gráfico de área ilustra o consumo de memória RAM ao longo de um experimento de 30 minutos. O início se dá em aproximadamente 115 MB, oscilando entre um mínimo de cerca de 28 MB e um pico de cerca de 159 MB. Essas flutuações podem ser atribuídas à atividade do coletor de lixo da JVM (*Java Virtual Machine*), que periodicamente libera a memória ocupada por objetos que não são mais necessários. A média de consumo manteve-se em torno de 100 MB, refletindo um equilíbrio entre o uso e a liberação de recursos de memória ao longo do tempo.

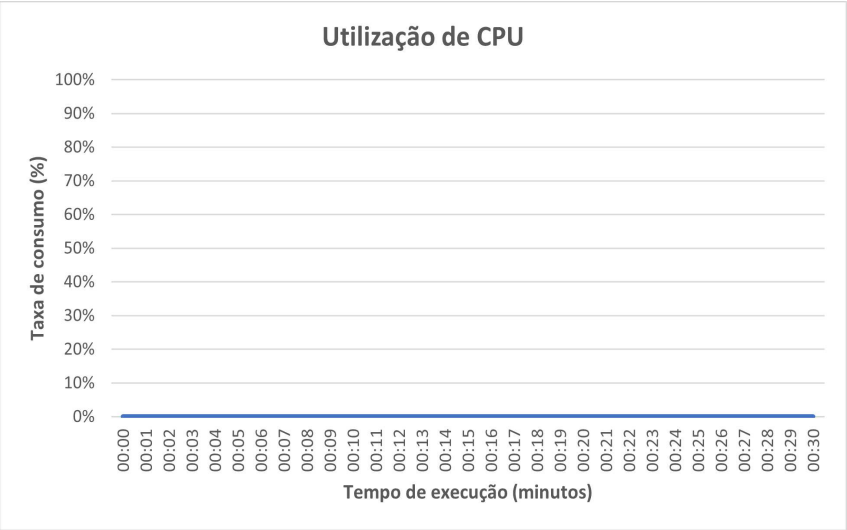
Figura 22 – Uso de memória na primeira etapa da fase 1 do experimento.



Fonte: elaborada pelo autor (2023).

A Figura 23 exibe o gráfico de consumo de CPU ao longo do experimento. O gráfico mostra que a taxa de consumo de CPU manteve-se em 0% desde o começo até o final do processamento. A ausência de picos ou elevações no gráfico indica que a carga de trabalho imposta ao sistema foi bem gerenciada, com a otimização da utilização da CPU sendo efetiva ao longo de todo o período testado.

Figura 23 – Taxa de utilização de CPU na primeira etapa da fase 1 do experimento.



Fonte: elaborada pelo autor (2023).

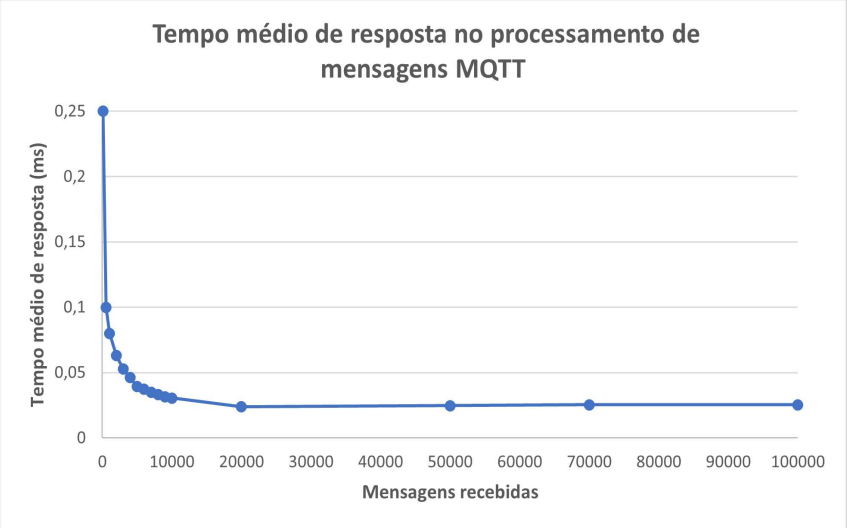
6.2.1.2 Etapa 2: Teste de carga com extrema demanda

Para avaliar a capacidade de resposta e escalabilidade do sistema, a carga de dados foi consideravelmente incrementada. Um total de 1000 sensores foi configurado para publicar mensagens a cada segundo. Durante o teste, uma amostra de 100.000 mensagens foi coletada ao longo de aproximadamente 2 minutos.

Os resultados mostraram que o tempo médio de processamento por mensagem iniciou em 0,26 milissegundos. Observou-se uma melhoria no desempenho do sistema à medida que o experimento prosseguia. Essa otimização progressiva pode ser atribuída a vários fatores, como *caching* de operações, otimização do gerenciamento de *threads* pelo sistema operacional e pelo próprio *broker* MQTT.

A Figura 24 ilustra a relação entre o tempo médio de resposta e o volume de mensagens MQTT processadas.

Figura 24 – Tempo médio de resposta na segunda etapa da fase 1 do experimento.



Fonte: elaborada pelo autor (2023).

Ao processar 1000 mensagens, o tempo médio de processamento foi reduzido para 0,08 ms, e atingiu 0,030 ms com 10.000 mensagens. Com o contínuo ajuste e acomodação da carga de dados, o sistema estabilizou-se após o processamento de 20.000 mensagens, evidenciado por um tempo médio de resposta constante de 0,024 ms. Essa estabilidade foi mantida até o processamento de 70.000 mensagens, momento em que o tempo médio de resposta se elevou para 0,025 ms, mas permaneceu constante até o final da amostra de 100.000 mensagens.

A Tabela 4 ilustra os resultados obtidos na etapa 2 da fase 1 do experimento. As métricas selecionadas para análise incluem o número total de operações realizadas, o tempo médio necessário para processar cada operação e o *throughput*.

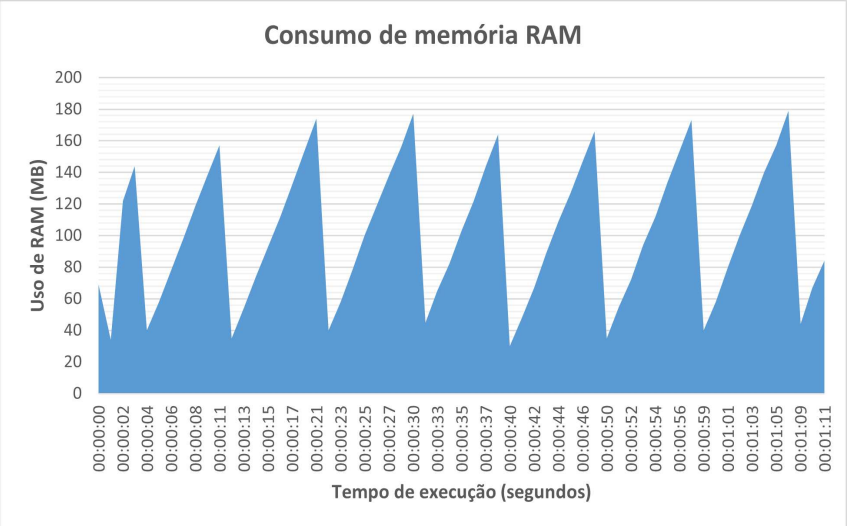
Tabela 4 – Resultados na segunda etapa da fase 1 do experimento.

Test	Operações	Tempo médio	Throughput
MQTT	100	0,26 ms	4.000 ops/s
	1.000	0,08 ms	12.500 ops/s
	10.000	0,030 ms	32.679 ops/s
	20.000	0,024 ms	41.666 ops/s
	30.000	0,023 ms	42.999 ops/s
	50.000	0,024 ms	40.523 ops/s
	70.000	0,025 ms	39.369 ops/s
	100.000	0,025 ms	39.605 ops/s

Quanto ao *throughput*, ao analisar a carga de trabalho, o sistema demonstrou um aumento significativo na capacidade de processamento à medida que o número de mensagens aumentava. Inicialmente, após o processamento de 100 mensagens, registrou-se um valor de 4.000 ops/s, o qual aumentou para 12.500 ops/s com 1.000 mensagens. Após o processamento de 5.000 mensagens, atingiu-se a marca de 25.380 ops/s. Com mais de 10.000 mensagens, a capacidade de processamento continuou a crescer, alcançando 32.679 ops/s. A partir desse ponto, houve uma estabilização em torno de 40.000 ops/s.

A Figura 25 apresenta o consumo de memória RAM ao longo do experimento. Observa-se um padrão cíclico de picos e quedas, evidenciando um aumento no consumo de memória seguido pela liberação de recursos. O consumo inicial de memória foi de 69 MB, variando entre um mínimo de 34 MB e um máximo de 177 MB. A média de consumo de memória ao longo do experimento foi de aproximadamente 102 MB.

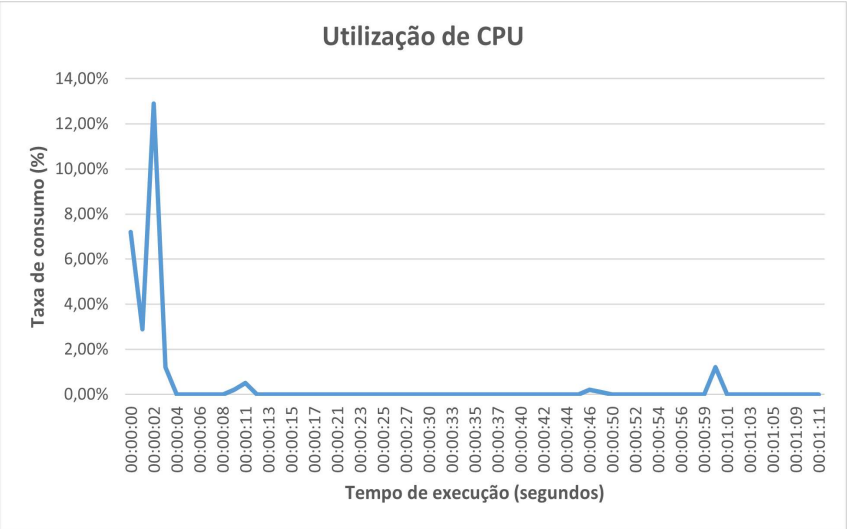
Figura 25 – Uso de memória na segunda etapa da fase 1 do experimento.



Fonte: elaborada pelo autor (2023).

A Figura 26 evidencia a taxa de utilização de CPU ao longo do experimento, marcando um início com um consumo significativo de 7,2%, que rapidamente decresce para 2,9%. Posteriormente, observa-se um pico significativo, onde o consumo atinge 12,9%, antes de se estabilizar em níveis mais baixos. Este comportamento sugere uma inicialização intensiva de processos ou a alocação de recursos, seguida por uma fase de estabilização onde o consumo de CPU se normaliza.

Figura 26 – Taxa de utilização de CPU na segunda etapa da fase 1 do experimento.



Fonte: elaborada pelo autor (2023).

Os picos subsequentes podem indicar atividades pontuais de processamento intensivo, como operações de coleta de lixo, que temporariamente exigem mais do processador. Entretanto, a tendência geral após os picos iniciais mostra um perfil de baixo consumo, indicando que o sistema gerencia eficientemente sua carga de trabalho sem sobrecarregar o CPU de maneira contínua.

6.2.2 Fase 2: Eficiência do registro de transações na blockchain

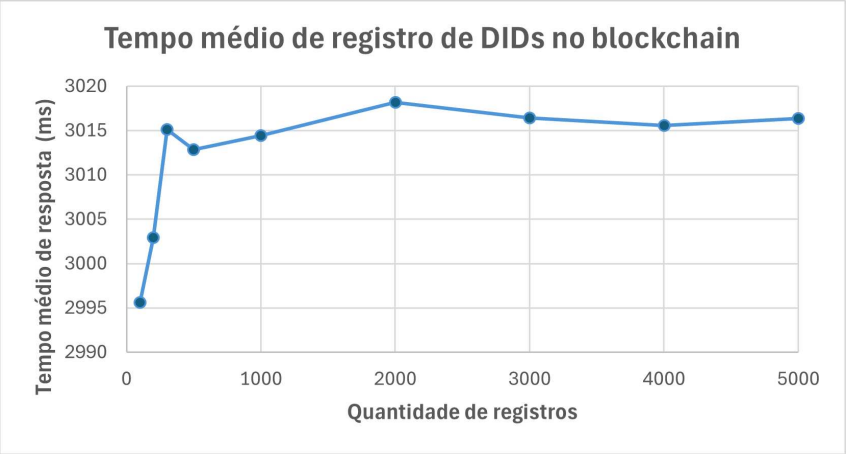
Nesta fase, avaliou-se a eficiência do registro de transações na blockchain com o objetivo de gerar e registrar DIDs que identificam as publicações por dispositivo. Foram registrados DIDs correspondentes a diferentes volumes de dados, e mediu-se o tempo e os recursos necessários para cada operação. A análise considerou amostras contendo cinco mil registros, processados em um período total aproximado de quatro horas e 11 minutos. O experimento envolveu um conjunto de dez sensores programados para publicar mensagens a cada segundo.

Nesta etapa, o tempo de processamento foi considerável, uma vez que o registro do DID na blockchain envolve requisições NYM, e a aplicação aguarda a resposta com o DID registrado. O registro de DIDs envolve a submissão de requisições NYM à

blockchain, que são comandos utilizados para criar ou atualizar identidades na rede. Após o envio dessas requisições, a aplicação precisa aguardar a confirmação da inclusão do DID no registro. Esse processo de espera pela resposta com o DID registrado adiciona consideravelmente ao tempo total de processamento, uma vez que depende da validação da transação pelos nós da rede e da propagação dessa informação.

A Figura 27 apresenta o gráfico do tempo médio de resposta para o processo de registro de DIDs na blockchain. Observa-se um leve aumento no tempo de resposta à medida que a quantidade de registros de DIDs aumenta. O primeiro registro demora 328 ms.

Figura 27 – Tempo médio de resposta em transações com a blockchain.



Fonte: elaborada pelo autor (2023).

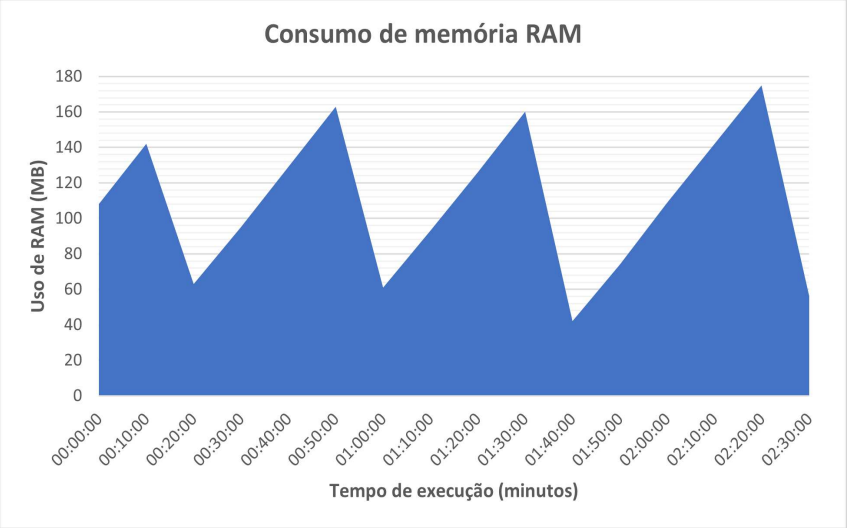
A Tabela 5 ilustra os resultados obtidos nesta fase do experimento. O sistema manteve um *throughput* constante de 0.33 operações por segundo em todas as amostras, equivalente a cerca de 19.8 operações por minuto. No entanto, à medida que a carga aumenta, há sobrecarga nos nós validadores, evidenciando-se pela competição por espaço de bloco na blockchain, resultando em atrasos no processamento das transações. Após 100 operações, o tempo médio é de 2995 ms; após 1000 operações, é de 3014 ms; e ao processar 5000 transações, aumenta para 3016 ms.

Tabela 5 – Resultados da fase 2 do experimento.

Test	Operações	Tempo médio	Throughput
Registro DIDs	100	2.995 ms	0,33 ops/s
	1000	3.014 ms	0,33 ops/s
	2000	3.018 ms	0,33 ops/s
	3000	3.016 ms	0,33 ops/s
	4000	3.015 ms	0,33 ops/s
	5000	3.016 ms	0,33 ops/s

A Figura 28 ilustra o consumo de memória RAM ao longo do experimento, evidenciando flutuações regulares que correspondem às operações de registro na blockchain. A memória RAM utilizada iniciou em 108 MB e oscilou entre 42 MB e 175 MB. A média de consumo manteve-se em 108 MB o que sugere que o sistema não exibiu um aumento proporcional no uso de memória.

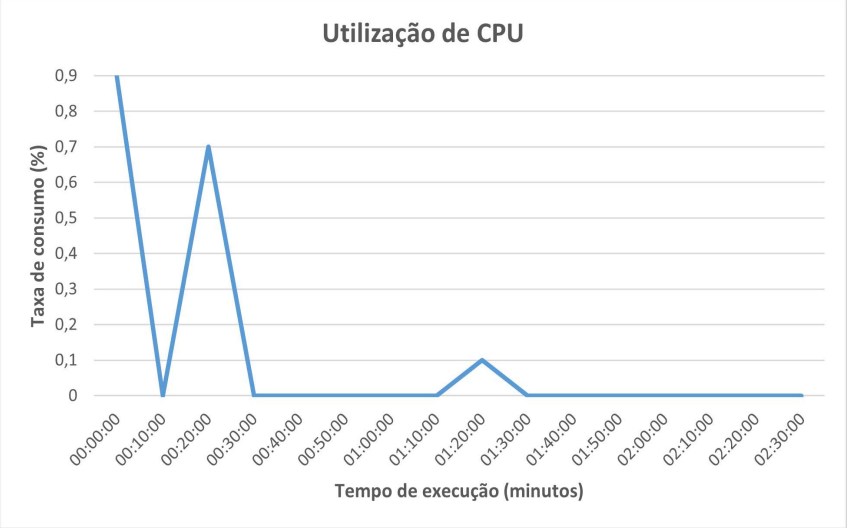
Figura 28 – Uso de memória na fase 2 do experimento.



Fonte: elaborada pelo autor (2023).

A Figura 29 demonstra a utilização de CPU ao longo do tempo durante o experimento, destacando os picos de atividade e os períodos de inatividade.

Figura 29 – Taxa de utilização de CPU na fase 2 do experimento.



Fonte: elaborada pelo autor (2023).

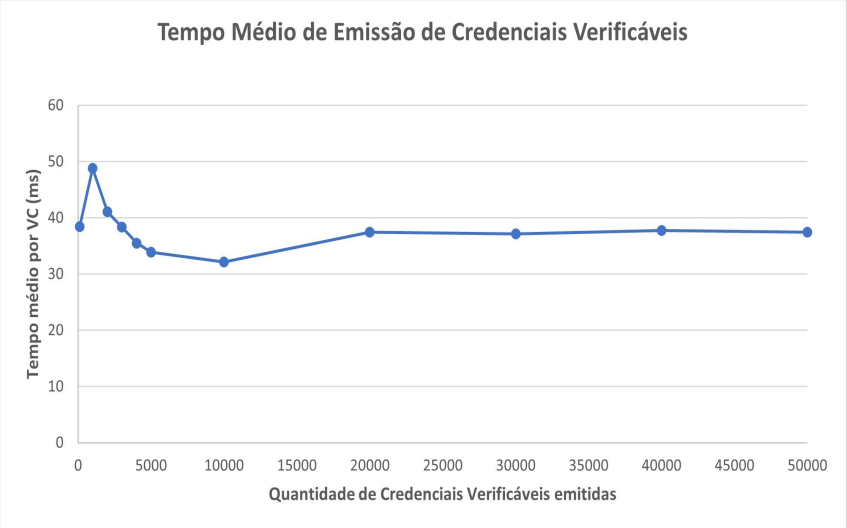
Inicialmente, a taxa de ocupação da CPU atingiu 0,9%, indicando uma breve fase de intensa atividade. Após os primeiros 10 minutos, a taxa de utilização diminuiu. Um pico subsequente de 0,7% foi observado aproximadamente aos 20 minutos. No entanto, por grande parte do experimento, a CPU permaneceu em 0%, sugerindo que a aplicação não exigiu recursos computacionais significativos após a atividade inicial.

6.2.3 Fase 3: Tempo de emissão de credenciais verificáveis

Nesta fase do experimento, o foco principal foi avaliar o tempo necessário para a emissão de credenciais verificáveis. Esta fase, uma extensão do Experimento 1, envolveu a medição do tempo em que o gateway, com uma instancia do agente em nuvem, leva para processar uma requisição - isto é, emitir uma credencial e retornar uma resposta contendo a identificação da credencial emitida. Para esta análise, foram utilizadas amostras de 50 mil registros, e o tempo total de processamento foi aproximadamente de 35 minutos. O teste foi realizado com 1000 sensores, cada um programado para publicar mensagens a cada segundo.

A Figura 30 apresenta o tempo médio necessário para a emissão de credenciais verificáveis em função do volume de credenciais processadas.

Figura 30 – Tempo médio de reposta na fase 3 do experimento.



Fonte: elaborada pelo autor (2023).

Inicialmente, o tempo médio de emissão é relativamente mais alto, mas há uma tendência de diminuição à medida que o número de credenciais emitidas aumenta. A curva começa com um tempo médio superior a 38,42 ms quando são emitidas 100 mensagens, e gradualmente se estabiliza em torno de 37,47 ms após a emissão de 50.000 credenciais. Este comportamento sugere que o sistema pode se tornar mais eficiente com o aumento da carga de trabalho. Possíveis otimizações, como

caching de processos e melhorias na gestão de recursos computacionais, podem estar contribuindo para isso à medida que mais dados são processados.

A Tabela 6 apresenta os dados coletados durante o a terceira fase do experimento, com as métricas do tempo médio de resposta e *throughput* durante o processo de troca de VCs.

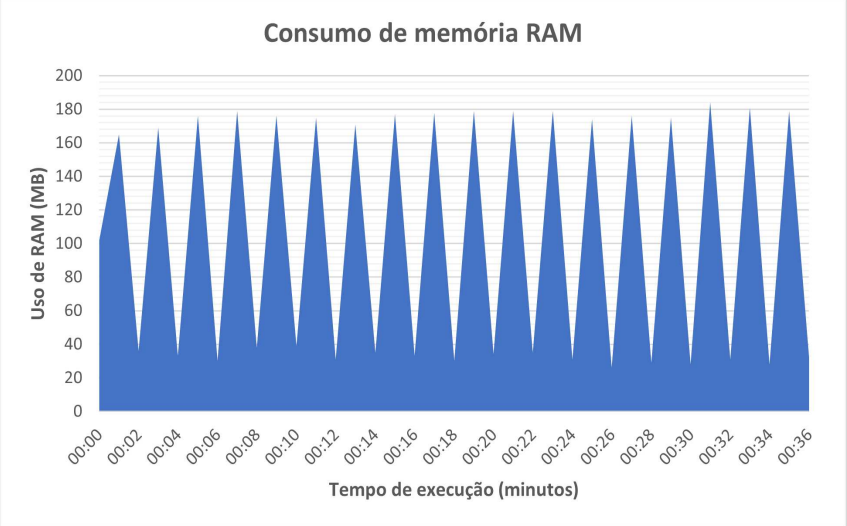
Tabela 6 – Resultados da fase 3 do experimento.

Test	Measure	Value	Throughput
VCs exchange	100	38,42 ms	26 ops/s
	1000	48,801 ms	20 ops/s
	10000	32,125 ms	31 ops/s
	20000	37,446 ms	27 ops/s
	30000	37,108 ms	27 ops/s
	40000	37,758 ms	26 ops/s
	50000	37,465 ms	27 ops/s

Em relação ao *throughput* nesta fase do experimento, foi observado que era de 20 operações por segundo (ops/s) após a emissão de 1000 credenciais. Este valor aumentou para 31 ops/s após 10000 requisições e se manteve em uma taxa de 37 ops/s após 200000 requisições por segundo.

A Figura 31 mostra o comportamento do consumo de memória RAM ao longo de um intervalo experimental de 35 minutos. Inicialmente, o sistema registrou um uso de memória de 165 MB, refletindo o estado inicial de recursos necessários. À medida que o experimento progredia, notaram-se oscilações no uso de memória, que variavam de um mínimo de 29 MB a um máximo de 184 MB.

Figura 31 – Uso de memória na fase 3 do experimento.

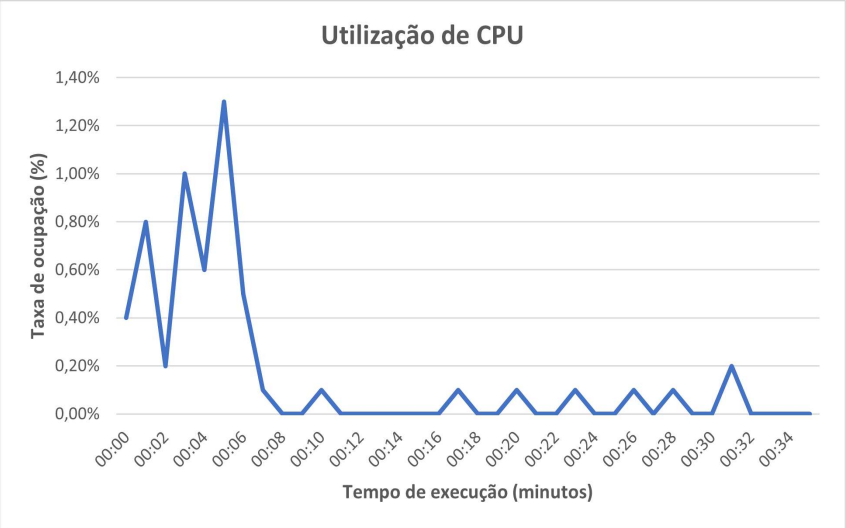


Fonte: elaborada pelo autor (2023).

Apesar dessa variação, a média de consumo manteve-se em 119 MB, o que aponta para uma gestão eficiente de memória ao longo do tempo, mantendo a estabilidade necessária para o funcionamento contínuo e eficaz do sistema.

A Figura 32 ilustra as flutuações na utilização da CPU ao longo de um experimento de 35 minutos. Inicialmente, o uso da CPU foi mais intenso, diminuindo a partir do oitavo minuto e mantendo uma média de ocupação de 0,10%.

Figura 32 – Taxa de utilização de CPU na fase 3 do experimento.



Fonte: elaborada pelo autor (2023).

Apesar das oscilações, a utilização da CPU ao longo do experimento manteve-se relativamente estável, sinalizando uma gestão de carga de trabalho consistente.

6.3 ANÁLISE E DISCUSSÃO

Com base nos resultados obtidos, é possível extrair conclusões significativas. Foi observado que o processo de registro de DIDs impõe uma carga considerável aos nós da blockchain, resultando em tempos de resposta prolongados. Tal constatação indica uma limitação do sistema em processar um alto volume de solicitações simultâneas, comprometendo potencialmente a eficiência e eficácia em cenários que demandam agilidade e processamento em tempo real.

Quanto ao impacto no uso de recursos como RAM e CPU, este mostrou-se minimalista, sugerindo que um aumento no número de dispositivos enviando dados não constitui um problema significativo para a capacidade operacional do sistema. Verificou-se que a quantidade máxima de RAM necessária foi de aproximadamente 180 MB, e a execução de múltiplas trocas de credenciais simultaneamente não afetou de maneira relevante o desempenho dos recursos do sistema.

A melhoria no tempo de resposta com o aumento do número de credenciais emitidas pode ser explicada pelo "aquecimento" da Máquina Virtual Java (JVM), que otimiza o código para execução mais rápida ao longo do tempo (CHEN; CHANG, J.; HOU, 2011). Estruturas de dados em cache e a alocação eficiente de *threads* melhoram à medida que a aplicação processa mais dados, permitindo um gerenciamento mais eficaz de conexões e sessões, especialmente em comunicações via MQTT (DYKSTRA; SRISA-AN; CHANG, J.M., 2002). A biblioteca ACA-Py pode ajustar suas operações internas para lidar com a carga de trabalho de forma mais eficiente (TEAM ARIES, 2021). Além disso, a JVM otimiza suas operações de coleta de lixo, reduzindo a sobrecarga e contribuindo para um desempenho melhor (CHEN; CHANG, J.; HOU, 2011). Essas otimizações são resultado do processo de adaptação e otimização contínua da aplicação e do ambiente de execução, incluindo ajustes automáticos baseados em *profiling* e análise de desempenho.

Destaca-se também que o processo de troca de credenciais se tornou substancialmente mais ágil quando a etapa de registro de DIDs na blockchain foi omitida. Isso ocorre porque, uma vez registrados o esquema e a definição de credencial na blockchain, tais procedimentos não requerem repetição, permitindo a geração e assinatura de múltiplas credenciais com a chave privada já armazenada na carteira digital.

Como resultado, conclui-se que a sobrecarga do registro na blockchain será o principal fator que afeta o desempenho. Se o registro de evidências na blockchain for necessário para compor a credencial, isso pode representar um desafio de escalabilidade. Portanto, é essencial avaliar a necessidade dessa etapa ou considerar propostas alternativas destinadas a reduzir a sobrecarga no blockchain. No entanto, os resultados apresentados são promissores e sugerem que a emissão de credenciais em larga escala para dispositivos IoT é viável, conforme evidenciado no artigo publicado decorrente desse estudo (SANTOS; LOFFI; WESTPHALL, 2023).

6.3.1 Considerações de segurança

É importante considerar aspectos críticos relacionados à segurança e privacidade da arquitetura proposta. Proteger as chaves criptográficas na carteira é essencial nos processos de assinatura e gerenciamento de custódia de dados. Essa proteção pode ser reforçada através do uso de Enclaves Seguros (HYPERLEDGER, 2023b).

Para garantir a privacidade dos atributos do dispositivo e dos dados coletados, pode-se utilizar a técnica ZKP ao compilar apresentações verificáveis antes de compartilhá-las com as partes interessadas. Outro aspecto importante é a agregação, que ocorre quando as partes interessadas coletam informações do mesmo dispositivo, solicitando várias apresentações verificáveis. Isso pode representar desafios à privacidade, mas o ZKP pode resolver isso ao ocultar dados desnecessários.

Além disso, ao lidar com a custódia de dados envolvendo dispositivos IoT e

a criação de VCs, é importante considerar os riscos de segurança. Vários tipos de ataques de *side-channel* podem ser empregados para obter informações sobre as VCs. Por exemplo, mecanismos na Internet podem rastrear dispositivos IoT e os dados que eles geram, como *cookies*, impressões digitais de dispositivos ou informações de localização. O esquema proposto não pode impedir o uso dessas tecnologias de rastreamento se elas estiverem instaladas nos dispositivos ou no *broker*.

Outro risco surge quando links são incorporados nas VCs. Embora a própria credencial esteja protegida contra adulterações, seu conteúdo externo não está tornando os links vulneráveis a modificações por invasores. Uma maneira de mitigar esse risco é gerando um *hash* dos dados externos e incorporando-o na credencial. Outra opção a ser explorada é o uso de uma blockchain privada para armazenar os dados.

6.3.2 Aplicação de credenciais verificáveis em outros setores

Adicionando à análise realizada, é pertinente considerar diferentes cenários nos quais os resultados observados possam ser aplicados e suas implicações. Embora na perícia forense o número de dispositivos possa ser variado, existem diversos contextos em que a emissão de credenciais verificáveis em grande escala se faz necessária e justificável.

No âmbito da saúde, por exemplo, dispositivos IoT podem ser utilizados para monitoramento contínuo de pacientes, coletando dados vitais que, ao serem transformados em credenciais verificáveis, garantem a integridade e a veracidade das informações. Isso é fundamental para diagnósticos precisos e para a tomada de decisões médicas baseadas em dados confiáveis.

Na segurança pública, a utilização de sensores e dispositivos conectados para monitoramento de áreas urbanas e detecção de atividades suspeitas ou perigosas pode ser amplamente beneficiada pela emissão de credenciais verificáveis. Tais credenciais podem servir como prova digital inalterável de eventos, contribuindo para investigações e para a melhoria da segurança de comunidades.

Um caso emergente significativo, conforme mencionado em materiais em cursos da Linux Foundation (CURRAN; HOWARD, 2021b), é a aplicação de dispositivos IoT em contextos industriais, como a medição de emissões de gases de efeito estufa em unidades fabris. A emissão de credenciais verificáveis por esses dispositivos não só assegura a confiabilidade dos dados coletados, mas também facilita a conformidade com regulamentações ambientais, ao prover um registro auditável e à prova de adulteração.

Outro cenário relevante é o uso em postos de combustíveis, onde dispositivos IoT podem emitir credenciais verificáveis baseadas em dados de sensores. Isso pode ser aplicado, por exemplo, para monitorar a manutenção e o uso de veículos, garantindo a acurácia dos dados ao longo da vida útil do automóvel ou assegurando a

conformidade de bombas de combustível com padrões de segurança, mediante a emissão de credenciais verificáveis que podem ser monitoradas remotamente (CURRAN; HOWARD, 2021b).

A Fundação Sovrin, amplamente reconhecida por seu papel pioneiro na promoção da identidade auto-soberana, recentemente expandiu seu escopo de influência com a publicação de um *whitepaper*. Este documento detalha a aplicação da SSI no contexto da IoT, um campo que está rapidamente se expandindo e transformando a maneira como as pessoas interagem com o mundo digital e físico. Intitulado "SSI and IoT", o *whitepaper* está disponível no site oficial da Fundação Sovrin (SOVRIN, 2020) e oferece uma visão abrangente de como a identidade auto-soberana pode ser integrada com dispositivos IoT para promover uma maior segurança, privacidade e eficiência na gestão de identidades digitais e físicas.

Essas considerações destacam a versatilidade e o potencial das credenciais verificáveis em uma ampla gama de aplicações, estendendo-se muito além da perícia forense. A capacidade de gerar credenciais verificáveis em larga escala para dispositivos IoT abre novas possibilidades em diversos setores, demonstrando a viabilidade e a necessidade de soluções escaláveis e eficientes como as discutidas neste estudo.

7 CONCLUSÃO

Este estudo propôs uma arquitetura de gateway para integrar dispositivos IoT com recursos limitados às tecnologias de identidade auto-soberana (SSI). Através desta arquitetura, foi possível certificar dados coletados de ambientes IoT por meio da emissão de credenciais verificáveis (VCs) e estabelecer conexão com outros domínios a partir de identificadores decentralizados (DIDs). Este sistema destacou-se por sua capacidade de assegurar a confiabilidade, integridade e rastreabilidade das informações, elementos vitais para a gestão da cadeia de custódia de dados, por meio de blockchain, sobretudo em contextos forenses que envolvem dispositivos IoT.

Comparado à vasta gama de trabalhos relacionados apresentados, este estudo destacou-se como uma solução abrangente para verificar a integridade dos dados em dispositivos IoT, focando explicitamente na aplicação de SSI para enfrentar esses desafios. Enquanto numerosos estudos (TERZI *et al.*, 2020; DIXIT; CREASEY; RAJARAJAN, 2022; GEBRESILASSIE *et al.*, 2020; LUECKING *et al.*, 2020; FEDRECHESKI *et al.*, 2021; YIN *et al.*, 2022; THEODOULI *et al.*, 2020; FOTOPOULOS *et al.*, 2020; REGUEIRO *et al.*, 2022; SHARMA, P.; GODFREY; TRIVEDI, 2022; DIEGO; REGUEIRO; FERNANDEZ, 2021; KORTESNIEMI *et al.*, 2019; BERZIN *et al.*, 2021) abordaram questões específicas relacionadas ao IoT, como segurança, controle de acesso e eficiência de comunicação em vários cenários, a abordagem deste trabalho foi única em sua concentração nos desafios relacionados a dispositivos IoT com recursos limitados e na custódia de dados nesses ambientes. Isso não apenas evidenciou uma contribuição para o campo, mas também ressaltou o potencial da arquitetura proposta para aprimorar aplicações IoT e promover o compartilhamento seguro de dados.

Os objetivos específicos delineados foram plenamente alcançados. A arquitetura proposta não apenas facilitou a emissão de credenciais verificáveis com base em dados IoT, mas também implementou um registro distribuído verificável para a verificação, abordando diretamente as necessidades de segurança e integridade de dados. Além disso, o trabalho se esforçou na abordagem se utilizar DIDs como mecanismo para identificação de dispositivos e evidencias. Através da implementação prática e da avaliação de desempenho, demonstrou-se a viabilidade e a o aumento significativo na carga de dados evidencia que a arquitetura pode se tornar escalável, consolidando sua posição como uma solução eficaz para os desafios identificados.

Para futuros estudos, recomenda-se a investigação sobre a implementação de canais de nível 2 para melhorar a escalabilidade do sistema. Esses canais poderiam ser explorados para processar transações menos críticas ou menos frequentes, aliviando a carga na cadeia principal e potencialmente melhorando o desempenho geral da rede. Além disso, sugere-se a implementação de um cliente leve que opere diretamente em dispositivos IoT, com foco na otimização do consumo de energia e na eficiência da

utilização da rede. Ademais, prevê-se a realização de comparações detalhadas com outras soluções aplicáveis a diferentes domínios, bem como análises aprofundadas sobre a escalabilidade e a aplicabilidade em cenários reais, visando ampliar o escopo e a efetividade da solução proposta. Portanto, este trabalho não apenas atendeu aos objetivos específicos propostos, mas também contribuiu significativamente para o avanço das tecnologias de SSI aplicadas ao IoT, abrindo caminho para futuras investigações e desenvolvimentos na área.

7.1 LIMITAÇÕES DO TRABALHO

Apesar dos avanços apresentados, este estudo enfrentou algumas limitações importantes. Primeiramente, a ausência de comparações diretas com outros trabalhos semelhantes limitou a possibilidade de avaliar o desempenho relativo da arquitetura proposta em um contexto mais amplo. A singularidade da abordagem, apesar de ser um diferencial importante, restringiu a capacidade de realizar análises comparativas com soluções existentes.

Adicionalmente, a arquitetura desenvolvida não abordou especificamente a autenticação de dispositivos IoT usando tecnologias de identidade auto-soberana, dado às características dos dispositivos restritos, focando mais na integridade e rastreabilidade dos dados. Esse aspecto representa uma área potencial para melhorias futuras, visando uma segurança aprimorada e um controle de acesso mais robusto nos ecossistemas IoT.

Uma limitação adicional que foi identificada refere-se à necessidade de certificação dos dispositivos antes da emissão das credenciais verificáveis. Este processo de certificação prévia dos dispositivos não foi abordado na arquitetura proposta, o que pode representar uma lacuna significativa em termos de segurança e autenticidade.

Outra limitação observada foi a potencial sobrecarga nos nós da blockchain, caso a utilização de DIDs seja recorrente, especialmente em cenários que exigem o registro intenso de DIDs para a identificação das evidências. Tal desafio pode impactar a escalabilidade e a eficiência do sistema, sugerindo a necessidade de explorar soluções de escalabilidade blockchain ou mecanismos alternativos de registro distribuído.

Por fim, a arquitetura proposta não abordou medidas específicas contra invasões em dispositivos IoT, o que poderia comprometer a segurança da rede e a autenticidade dos dados gerados. A inclusão de mecanismos robustos de segurança e recuperação para mitigar esses riscos seria uma valiosa adição à solução atual.

Essas limitações não diminuem a contribuição do trabalho, mas destacam áreas para pesquisa e desenvolvimento futuros. Abordar essas questões em estudos subsequentes poderá fortalecer ainda mais a aplicabilidade e eficácia das tecnologias de SSI em ambientes IoT.

REFERÊNCIAS

AFTAB, Haris; GILANI, Komal; LEE, JiEun; NKENYEREYE, Lewis; JEONG, SeungMyeong; SONG, JaeSeung. Analysis of identifiers in IoT platforms. **Digital Communications and Networks**, v. 6, n. 3, p. 333–340, 2020. ISSN 2352-8648.

ALDOWAH, Hanan; REHMAN, Shafiq; UMAR, Irfan. Security in Internet of Things: Issues, Challenges, and Solutions. *In*: [*S.l.*: *s.n.*], jul. 2019. ISBN 978-3-319-99006-4.

ALEX, M. Edington; KISHORE, R. Forensics framework for cloud computing. **Computers & Electrical Engineering**, v. 60, p. 193–205, 2017. ISSN 0045-7906.

ALGARNI, Sultan; EASSA, Fathy; ALMARHABI, Khalid; ALMALAISE, Abduallah; ALBASSAM, Emad; ALSUBHI, Khalid; YAMIN, Mohammad. Blockchain-based secured access control in an iot system. **Applied Sciences (Switzerland)**, MDPI AG, v. 11, p. 1–16, 4 fev. 2021. ISSN 20763417.

ALLEN, Christopher. **The Path to Self-Sovereign Identity**. [*S.l.*], 2016. Disponível em: `http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html`. Acesso em: 2 set. 2022.

AMAZON, Developer. **Smart Home Skill APIs**. [*S.l.*], 2023. Disponível em: `https://developer.amazon.com/en-US/docs/alexa/device-apis/smart-home-general-apis.html`. Acesso em: 20 mai. 2023.

ARSHAD, Humaira; JANTAN, Aman Bin; ABIODUN, Oludare Isaac. Digital forensics: Review of issues in scientific validation of digital evidence. **Journal of Information Processing Systems**, Korea Information Processing Society, v. 14, p. 346–376, 2 2018. ISSN 2092805X.

BERZIN, Oleg; ANSAY, Rafael; KEMPF, James; SHEIKH, Imam; HENDEL, Doron. The IoT Exchange, mar. 2021.

BONOMI, Silvia; CASINI, M.; CICCOTELLI, Claudio. B-CoC: A Blockchain-based Chain of Custody for Evidences Management in Digital Forensics, 12:1–12:15, 2018.

BRUNNER, Clemens; GALLERSDÖRFER, Ulrich; KNIRSCH, Fabian; ENGEL, Dominik; MATTHES, Florian. DID and VC: Untangling Decentralized

Identifiers and Verifiable Credentials for the Web of Trust ACM Reference Format. ACM, 2020.

CHEN, Kuo-Yi; CHANG, J.; HOU, Ting-Wei. Multithreading in Java: Performance and Scalability on Multicore Systems. **IEEE Transactions on Computers**, v. 60, p. 1521–1534, 2011.

CHOWDHURY, Rajarshi Roy; ANEJA, Sandhya; ANEJA, Nagender; ABAS, Emeroylariffion. Network Traffic Analysis based IoT Device Identification. *In*: p. 79–89.

ĆOSIĆ, Jasmin; ĆOSIĆ, Zoran. Chain of custody and life cycle of digital evidence. **Journal of computer technology and application**, jan. 2012.

CURRAN, Stephen; HOWARD, Carol. **Becoming a Hyperledger Aries Developer**. [*S.l.*], 2021b. Disponível em: `https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS173x+3T2021/`. Acesso em: 4 dez. 2022.

CURRAN, Stephen; HOWARD, Carol. **Introduction to Hyperledger Sovereign Identity Blockchain Solutions: Indy, Aries and Ursa**. [*S.l.*], 2021a. Disponível em: `https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS172x+2T2021`. Acesso em: 1 nov. 2022.

CURRAN, Stephen; PHILIPP, Artur; YILDIZ, Hakan; CURREN, Sam; JURADO, Victor Martinez. **Especificação AnonCreds**. [*S.l.*], 2022. Disponível em: `https://hyperledger.github.io/anoncreds-spec/`. Acesso em: 2 jan. 2023.

DAVIE, Matthew; GISOLFI, Dan; HARDMAN, Daniel; JORDAN, John; O'DONNELL, Darrell; REED, Drummond. The Trust over IP Stack. **IEEE Communications Standards Magazine**, Institute of Electrical e Electronics Engineers Inc., v. 3, p. 46–51, 4 dez. 2019. ISSN 24712825.

DIEGO, Santiago De; REGUEIRO, Cristina; FERNANDEZ, Gabriel Macia. Enabling Identity for the IoT-as-a-Service Business Model. **IEEE Access**, Institute of Electrical e Electronics Engineers Inc., v. 9, p. 159965–159975, 2021. ISSN 21693536.

DIXIT, Akanksha; CREASEY, Max Smith; RAJARAJAN, Muttukrishnan. A Decentralized IIoT Identity Framework based on Self-Sovereign Identity using Blockchain. *In*: p. 335–338.

DYKSTRA, L.; SRISA-AN, W.; CHANG, J.M. An analysis of the garbage collection performance in Sun's HotSpot/sup TM/ Java Virtual Machine. **Conference Proceedings of the IEEE International Performance, Computing, and Communications Conference (Cat. No.02CH37326)**, p. 335–339, 2002.

FEDRECHESKI, Geovane; COSTA, Laisa C P; AFZAL, Samira; RABAEY, Jan M; LOPES, Roseli D; ZUFFO, Marcelo K. A low-overhead approach for self-sovereign identity in IoT, 2021.

FEDRECHESKI, Geovane; RABAEY, Jan M.; COSTA, Laisa C.P.; CCORI, Pablo C. Calcina; PEREIRA, William T.; ZUFFO, Marcelo K. Self-Sovereign Identity for IoT environments: A Perspective. **GIoTS 2020 - Global Internet of Things Summit, Proceedings**, Institute of Electrical e Electronics Engineers Inc., jun. 2020.

FERDOUS, Md Sadek; CHOWDHURY, Farida; ALASSAFI, Madini O. In Search of Self-Sovereign Identity Leveraging Blockchain Technology. **IEEE Access**, v. 7, p. 103059–103079, 2019.

FOTOPOULOS, Filippos; MALAMAS, Vangelis; DASAKLIS, Thomas K.; KOTZANIKOLAOU, Panayiotis; DOULIGERIS, Christos. A Blockchain-enabled Architecture for IoMT Device Authentication. **2nd IEEE Eurasia Conference on IOT, Communication and Engineering 2020, ECICE 2020**, Institute of Electrical e Electronics Engineers Inc., p. 89–92, out. 2020.

GEBRESILASSIE, Samson Kahsay; RAFFERTY, Joseph; MORROW, Philip; CHEN, Liming Luke; ABU-TAIR, Mamun; CUI, Zhan. Distributed, Secure, Self-Sovereign Identity for IoT Devices. **IEEE World Forum on Internet of Things, WF-IoT 2020 - Symposium Proceedings**, Institute of Electrical e Electronics Engineers Inc., jun. 2020.

GUBBI, Jayavardhana; BUYYA, Rajkumar; MARUSIC, Slaven; PALANISWAMI, Marimuthu. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, p. 1645–1660, 7 2013. ISSN 0167739X.

EL-HAJJ, Mohammed; FADLALLAH, Ahmad; CHAMOUN, Maroun; SERHROUCHNI, Ahmed. A survey of internet of things (IoT) authentication schemes. **Sensors (Switzerland)**, MDPI AG, v. 19, 5 mar. 2019. ISSN 14248220.

HYPERLEDGER, Foundation. **Aries Cloudagent Python: Aries OpenAPI Demo**. [*S.l.*], 2023. Disponível em: `https://github.com/hyperledger/aries-cloudagent-python/blob/main/demo/AriesOpenAPIDemo.md#use-the-faber-agent-to-create-an-invitation.`. Acesso em: 1 jun. 2023.

HYPERLEDGER, Foundation. **Aries RFC 0050: Wallets**. [*S.l.*], 2023. Disponível em: `https://github.com/hyperledger/aries-rfcs/blob/main/concepts/0050-wallets/README.md`. Acesso em: 23 set. 2023.

IDC, 2019. **The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025**. v. 76. [*S.l.*], jun. 2019.

INDY, Hyperledger. **Hyperledger Indy SDK**: Hyperledger Foundation Projects INDY. [*S.l.*], 2022. Disponível em: `https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/`. Acesso em: 1 jun. 2022.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Information technology – Security techniques – Guidance on assuring suitability and adequacy of incident investigative method**. [*S.l.*: *s.n.*], 2015. ISO/IEC 27041.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Information technology – Security techniques – Guidelines for identification, collection, acquisition and preservation of digital evidence**. [*S.l.*: *s.n.*], 2012. ISO/IEC 27037:2012.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Information technology – Security techniques – Guidelines for the analysis and interpretation of digital evidence**. [*S.l.*: *s.n.*], 2015. ISO/IEC 27042:2015.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Information technology – Security techniques – Incident investigation principles and processes**. [*S.l.*: *s.n.*], 2015. ISO/IEC 27043:2015.

IP FOUNDATION, Trust over. **Smart Home Skill APIs**. [*S.l.*], 2023. Disponível em: `https://trustoverip.org/`. Acesso em: 20 mai. 2023.

JONG, Laurence de. **Self-Sovereign Identities**: What is a DID? [*S.l.*], 2020. Disponível em: `https://ldej.nl/post/self-sovereign-identities/`. Acesso em: 1 out. 2022.

KENT, Karen; CHEVALIER, Suzanne; GRANCE, Tim; DANG, Hung. **Guide to Integrating Forensic Techniques into Incident Response**. [*S.l.*], ago. 2006. Accessed: 2024-02-25. Disponível em: `https://csrc.nist.gov/pubs/sp/800/86/final`.

KITCHENHAM, Barbara; CHARTERS, Stuart. Guidelines for performing Systematic Literature Reviews in Software Engineering. v. 2, jan. 2007.

KORTESNIEMI, Yki; LAGUTIN, Dmitrij; ELO, Tommi; FOTIOU, Nikos. Improving the Privacy of IoT with Decentralised Identifiers (DIDs). **Journal of Computer Networks and Communications**, v. 2019, p. 1–10, mar. 2019.

LONE, Auqib. Forensic-chain: Ethereum blockchain based digital forensics chain of custody. **Scientific and practical cyber security journal**, v. 1, dez. 2017.

LUECKING, Markus; FRIES, Christian; LAMBERTI, Robin; STORK, Wilhelm. Decentralized Identity and Trust Management Framework for Internet of Things. **IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020**, Institute of Electrical e Electronics Engineers Inc., mai. 2020.

LUX, Zoltán András; BEIERLE, Felix; ZICKAU, Sebastian; GÖNDÖR, Sebastian. Full-text Search for Verifiable Credential Metadata on Distributed Ledgers. *In*: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS). [*S.l.*: *s.n.*], 2019. P. 519–528.

MASOOD, Faraz; FARIDI, Arman Rasool. Distributed Ledger Technology for Closed Environment. *In*: 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom). [*S.l.*: *s.n.*], 2019. P. 1151–1156.

MOYER, Stan. IoT Sensors and Actuators. **IEEE Internet of Things Magazine**, v. 2, n. 3, p. 10–10, 2019.

MOZILLA, Corporation. **WebThings**. [*S.l.*], 2023. Disponível em: `https://webthings.io/gateway/`. Acesso em: 20 mai. 2023.

NIK ZULKIPLI, Nurul Huda; ALENEZI, Ahmed; WILLS, Gary. IoT Forensic: Bridging the Challenges in Digital Forensic and the Internet of Things. *In*: p. 315–324.

PRAYUDI, Yudi; SN, Azhari. Digital Chain of Custody: State of the Art. **International Journal of Computer Applications**, v. 114, p. 1–9, 2015.

RAYES, Ammar; SALAM, Samer. **Internet of Things From Hype to Reality The Road to Digitization Second Edition**. [*S.l.*], 2019.

REGUEIRO, Cristina; GUTIERREZ-AGÜERO, Iván; AGÜERO, Ag¨; ANGUITA, Sergio; DIEGO, Santiago De; LAGE, Oscar. Protocol for Identity Management in Industrial IoT based on Hyperledger Indy. **International Journal of Computing and Digital Systems**, v. 12, p. 2210–142, 1 2022.

SANDVIK, Jens-Petter; FRANKE, K.; ÅRNES, André. Towards a Generic Approach of Quantifying Evidence Volatility in Resource Constrained Devices. **Digital Forensic Investigation of Internet of Things (IoT) Devices**, 2020.

SANTOS, Cristian Alves dos; LOFFI, Leandro; WESTPHALL, Carla Merkle. Ensuring Data Security in the Context of IoT Forensics Evidence Preservation with Blockchain and Self-Sovereign Identities. *In*: MUTHUKKUMARASAMY, Vallipuram; SUDARSAN, Sithu D.; SHYAMASUNDAR, Rudrapatna K. (Ed.). **Information Systems Security**. Cham: Springer Nature Switzerland, 2023. P. 319–338.

SHAH, Makhdoom; SALEEM, Shahzad; ZULQARNAIN, Roha. Protecting Digital Evidence Integrity and Preserving Chain of Custody. **The Journal of Digital Forensics, Security and Law**, ERAU Hunt Library - DIGITAL COMMONS JOURNALS, 2017.

SHARMA, Cheena; GONDHI, Naveen Kumar. Communication Protocol Stack for Constrained IoT Systems. *In*: 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU). [*S.l.*: *s.n.*], 2018. P. 1–6.

SHARMA, Prakhar; GODFREY, W. Wilfred; TRIVEDI, Aditya. When blockchain meets IoT: a comparison of the performance of communication protocols in a decentralized identity solution for IoT using blockchain. **Cluster Computing**, Springer, p. 1–16, dez. 2022. ISSN 15737543.

SOVRIN, Foundation. **Self-Sovereign Identity and IoT**. [*S.l.*], 2020. Disponível em: https://sovrin.org/wp-content/uploads/SSI-and-IoT-whitepaper.pdf. Acesso em: 1 out. 2022.

SPORNY, Manu; LONGLEY, Dave; SABADELLO, Markus; REED, Drummond; STEELE, Orie; ALLEN, Christopher. **Decentralized Identifiers (DIDs) v1.0**: Core architecture, data model, and representations. W3C Recommendation. [*S.l.*], 2022a. Disponível em: https://www.w3.org/TR/did-core/#a-simple-example. Acesso em: 1 set. 2022.

SPORNY, Manu; NOBLE, Grant; LONGLEY, Dave; BURNETT, Daniel C.; ZUNDEL, Brent; HARTOG, Kyle Den. **Verifiable Credentials Data Model v1.1**: Core architecture, data model, and representations. W3C Recommendation. [*S.l.*], 2022b. Disponível em: https://www.w3.org/TR/vc-data-model/. Acesso em: 1 set. 2022.

STATISTA. **Global IOT and non-IoT Connections 2010-2025**. [*S.l.*], 2020. Disponível em: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/.

STOYANOVA, Maria; NIKOLOUDAKIS, Yannis; PANAGIOTAKIS, Spyridon; PALLIS, Evangelos; MARKAKIS, Evangelos. A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches and Open Issues. **IEEE Communications Surveys & Tutorials**, PP, p. 1–1, jan. 2020.

TEAM ARIES, Hyperledger. **Aries Agents in context: The Big Picture**. [*S.l.*], 2021. Disponível em: https://github.com/hyperledger/aries-cloudagent-python/blob/main/docs/GettingStartedAriesDev/AriesBigPicture.md. Acesso em: 15 out. 2022.

TERZI, Sofia; SAVVAIDIS, Charalampos; VOTIS, Konstantinos; TZOVARAS, Dimitrios; STAMELOS, Ioannis. Securing Emission Data of Smart Vehicles with Blockchain and Self-Sovereign Identities. **Proceedings - 2020 IEEE International Conference on Blockchain, Blockchain 2020**, Institute of Electrical e Electronics Engineers Inc., p. 462–469, nov. 2020.

THEODOULI, Anastasia; MOSCHOU, Konstantinos; VOTIS, Konstantinos; TZOVARAS, Dimitrios; LAUINGER, Jan; STEINHORST, Sebastian. Towards a Blockchain-based Identity and Trust Management Framework for the IoV Ecosystem.

GIoTS 2020 - Global Internet of Things Summit, Proceedings, Institute of Electrical e Electronics Engineers Inc., jun. 2020.

TIGHTIZ, Lilia; YANG, Hyosik. A Comprehensive Review on IoT Protocols' Features in Smart Grid Communication, 2020.

TSCHOFENIG, Hannes; ARKKO, Jari; THALER, Dave; MCPHERSON, Danny. **Architectural Considerations in Smart Object Networking**: RFC 7452. [*S.l.*], 2015. Disponível em: `https://www.rfc-editor.org/rfc/pdfrfc/rfc7452.txt.pdf`. Acesso em: 1 set. 2022.

WIDATAMA, Krisna; PRAYUDI, B. Application of RC4 Cryptography Method to Support XML Security on Digital Chain of Custody Data Storage. **International Journal of Cyber-Security and Digital Forensics**, v. 7, p. 230–237, 2018.

YIN, Jie; XIAO, Yang; PEI, Qingqi; JU, Ying; LIU, Lei; XIAO, Ming; WU, Celimuge. SmartDID: A Novel Privacy-preserving Identity based on Blockchain for IoT. **IEEE Internet of Things Journal**, Institute of Electrical e Electronics Engineers Inc., abr. 2022. ISSN 23274662.

Apêndices

APÊNDICE A – CÓDIGOS PARA OPERAÇÕES COM POOL E WALLET

```
1     String walletName = "IoTWallet";
2     String poolName = "IoT_Pool";
3     String stewardSeed = "000000000000000000000000Steward1";
4     String poolConfig = "{\"genesis_txn\": \"C:genesis\"}";
5     String WALLET_CONFIG = "{ \"id\":\"" + walletName + "\", \"
        storage_type\":\"" + "default" + "\"}";
6     String WALLET_CREDENTIALS = "{\"key\":\"8
        dvfYSt5d1taSd6yJdpjq4emkwsPDDLYxkNFysFD2cZY\", \"
        key_derivation_method\":\"RAW\"}";
```

Listing A.1 – Configuração de Wallet e Pool

```
1  public Pool openPool() throws IndyException{
2    pool = Pool.openPoolLedger(POOL_NAME, "{}").get();
3    return pool;
4      }
5  public void closePool() throws IndyException{
6    pool.closePoolLedger().get();
7      }
8  public void createPool() throws IndyException{
9    Pool.createPoolLedgerConfig(POOL_NAME, POOL_CONFIG_PATH).get();
10     }
11 public void deletePool() throws IndyException{
12   Pool.deletePoolLedgerConfig(POOL_NAME).get();
13     }
```

Listing A.2 – Gerenciamento do Pool

```
1  private static String walletName = "IoTWallet";
2  public Wallet openWallet() throws IndyException{
3    walletHandle = Wallet.openWallet(WALLET_CONFIG,
       WALLET_CREDENTIALS).get();
4    return walletHandle;
5      }
6  public void closeWallet() throws IndyException {
7    walletHandle.closeWallet().get();    }
8  public void createWallet() throws IndyException {
9          Wallet.createWallet(WALLET_CONFIG, WALLET_CREDENTIALS).get
             ();    }
10 public void deleteWallet() throws IndyException {
11   Wallet.deleteWallet(WALLET_CONFIG, WALLET_CREDENTIALS).get();    }
```

Listing A.3 – Gerenciamento da Wallet