PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO
MESTRADO EM CIÊNCIAS DA COMPUTAÇÃO

GABRIEL DA SILVA OLIVEIRA PORTAL

PROVENDO SEGURANÇA SENSÍVEL A CONTEXTO DESCENTRALIZADA PARA A INDÚSTRIA 4.0

Porto Alegre
2020

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROVENDO SEGURANÇA SENSÍVEL A CONTEXTO DESCENTRALIZADA PARA A INDÚSTRIA 4.0

GABRIEL OLIVEIRA PORTAL

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Fabiano Passuelo Hessel

Porto Alegre
2020

Ficha Catalográfica

P842p Portal, Gabriel da Silva Oliveira

Provendo segurança sensível a contexto descentralizada para a Industria 4.0 / Gabriel da Silva Oliveira Portal . – 2020.
79p.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Fabiano Passuelo Hessel.

1. Internet das Coisas. 2. Segurança. 3. Segurança Sensível a Contexto. 4. Blockchain. 5. Industria 4.0. I. Hessel, Fabiano Passuelo. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

Gabriel da Silva Oliveira Portal

Provendo Segurança Sensível a Contexto Descentralizada para a Industria 4.0

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciencia da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em _27_ de __Agosto__ de _2020_.

BANCA EXAMINADORA:

Prof. Dr. Raul Ceretta Nunes (UFSM)

Prof. Dr. Cesar Augusto Missio Marcon (PUCRS)

Prof. Dr. Fabiano Passuelo Hessel (PPGCC/PUCRS - Orientador)

DEDICATÓRIA

Dedico este trabalho a meus pais e a todos os gigantes que me ergueram.

“Quem és tu que queres julgar,
com vista que só alcança um palmo,
coisas que estão a mil milhas?”
(Dante Alighieri)

AGRADECIMENTOS

Agradeço a meus pais, Eduardo e Denise, pelo suporte e compreensão durante meu curso de mestrado. Também agradeço pelos ensinamentos e discussões, pois foram fundamentais para me instigar a realizar minha pesquisa.

Agradeço a meu orientador, Fabiano Hessel, por me dar a chance de realizar meu mestrado e por me guiar em uma sala escura cheia de velas. Agradeço também, a meus amigos e colegas de mestrado, Ramão e Everton, pelas dicas, conversas e conhecimentos compartilhados.

A todos que de alguma forma me apoiaram durante minha pesquisa, muito obrigado!

PROVENDO SEGURANÇA SENSÍVEL A CONTEXTO DESCENTRALIZADA PARA A INDÚSTRIA 4.0

RESUMO

Soluções de Internet das Coisas (do inglês, *Internet of Things* ou IoT) vem sendo aplicadas, em uma frequência cada vez maior, a diversos ambientes, como indústria, agricultura e saúde. Por exemplo, as soluções voltadas à indústria 4.0 exploram conceitos como manufatura aditiva e manufactura cognitiva. Tais conceitos têm apresentado melhorias significativas de eficiência ao utilizar paradigmas como computação de borda e descentralização, emergindo como uma tendência. Em contrapartida, características de infraestrutura, como segurança, não foram desenvolvidas especificamente para aplicações descentralizadas. Isto é, abordagens de segurança convencionais, como criptografia e redes definidas por software (do inglês, *Software Defined Networks* ou SDN) tem apresentado pontos de falha ao serem implantadas em aplicações descentralizadas e colaborativas. Enquanto tais abordagens têm foco principalmente na proteção da camada de comunicação, aplicações descentralizadas podem ser compostas por integrantes (módulos) maliciosos, que acabam por comprometer a camada de aplicação. Por meio de ameaças como *Spoofing* e *Tampering*, integrantes maliciosos tem a capacidade de gerar integrantes ou dados "fantasmas", que podem comprometer processos do mundo físico. O comprometimento de processos do mundo físico pode ocasionar perda monetária e até mesmo colocar a vida humana em risco. Assim, surge a necessidade de desenvolver arquiteturas de segurança voltadas para aplicações descentralizadas. Considerando isto, o presente trabalho propõe uma arquitetura que provê uma camada extra de segurança para aplicações descentralizadas. As tecnologias de Blockchain e Segurança Sensível a Contexto (do inglês, *Context-aware Security* ou CAS) tem apresentado eficiência em fornecer, respectivamente, integridade de dados e asserção de processos para a IoT. Assim, a arquitetura proposta combina a utilização destas duas tecnologias, uma vez que elas apresentam uma relação de complementação,

proporcionando uma melhoria da segurança para ambientes descentralizados. A arquitetura foi validada em um cenário de industria 4.0, mais especificamente em um cenário de manufatura aditiva. Os resultados demostraram que a arquitetura foi efetiva para os casos avaliados.

Palavras-Chave: Internet das Coisas, Blockchain, Segurança Sensível a Contexto, Compartilhamento de Contexto, Segurança, Integridade, Confiabilidade, Manufactura aditiva, Industria 4.0.

PROVIDING DECENTRALIZED CONTEXT-AWARE SECURITY TO INDUSTRY 4.0

ABSTRACT

The Internet of Things (IoT) is increasingly being applied to a range of environments, as industry, farming and health. An example is the industry 4.0, that explores concepts as Additive Manufacturing and Cognitive Manufacturing. Such concepts have being presenting major efficiency improvements, exploring trend paradigms as Decentralization and Edge computing. However, infrastructure characteristics as security were not developed specific to decentralized approaches. Hence, standard security approaches as cryptography and Software Defined Networks (SDN) has presented security breaches in decentralized and collaborative environments. While such approaches aims to protect a communication layer, decentralized applications can be composed by malicious members, that compromises an application layer. Through threats as Spoofing and Tampering, malicious members are able to generate ghost members and data, that can compromise a physical world process. The compromising of a physical world process can cause monetary loses and even put the human life in risk, thus, emerges the need for develop security architectures specific for decentralized applications. Therefore, this work proposes an architecture that provides an extra security layer to decentralized applications. The Blockchain and the Context-aware Security (CAS) technologies have being showing efficient to respectively provide data integrity and process assertion for IoT. The proposed architecture combines the use of these two technologies, since they have a complementary relationship, providing an improvement in security for decentralized environments. The architecture was validated in an industry 4.0 scenario, more specifically in an additive manufacturing scenario. The results showed that the architecture was effective for the evaluated cases.

Keywords: Internet of Things, Blockchain, Context-aware Security, Context Sharing, Security, Integrity, Reliability, Additive Manufacturing, Industy 4.0.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AMU – Additive Manufacturing Unit

API – Application Programming Interface

CAS – Context-aware Security

CBOR – Concise Binary Object Representation

DCASTP – Decentralized Context Aware Security Transaction Processor

DOS – Denial of Service

ERP – Enterprise Resource Planning

FASTEN – Flexible and Autonomous Manufacturing Systems for Custom-Designed Products

FBA – Federated Byzantine Agreement

GDPR – General Data Protection Regulation

IOT – Internet of Things

IRENA – International Renewable Energy Agency

JSON – JavaScript Object Notation

LGPD – Lei Geral de Proteção de Dados Pessoai

NIST – National Institute of Standards andTechnology

P2P – Peer-to-peer

PBFT – Pratical Byzantine Fault Tolerance

PEP – Fiware Policy Enforcement Point Proxy

PKI – Public-key Infrastructure

POW – Proof of Work

POS – Proof of Stake

POET – Proof of Elapsed Time

REST – Representational State Transfer

RFID – Radio-frequency identification

SDN – Software Defined Networks

TLS – Transport Layer Security

TP – Transaction Processor

VM – Virtual Machine

SUMÁRIO

1. INTRODUÇÃO

A internet das coisas (do inglês, *Internet of Things* ou IoT) não é mais um paradigma emergente. Segundo um relatório da Gartner de julho de 2019 [2], a IoT já passou pelo ciclo de expectativas altas, atualmente enfrentando uma fase de exploração de casos de uso adequados. Ferramentas como Monitoramento em tempo real, Automação, Manutenção preditiva e Análise de dados, proporcionam para a IoT a capacidade de melhorar eficiência e produtividade em diversos setores. Casos de uso relacionados a setores como saúde, indústria e mobilidade tem ganhado destaque, devido ao grande impacto causado, ou seja, alterando características fundamentais de cada setor por meio de mudanças nos respectivos modelos de negócio [4] [34]. Por exemplo, com cada vez maior frequência, a indústria tem automatizado e monitorado tarefas até então realizadas por mão de obra humana, impactando significativamente em características como custo e tempo de produção [40].

Predições realizadas por organizações como NIST (*National Institute of Standards and Technology*) e IRENA (*International Renewable Energy Agency*), indicam que nos próximos anos um dos principais setores afetados pela IoT é o industrial [34][40]. Tal impacto acompanha o movimento chamado Indústria 4.0 ou IoT industrial, que toma proveito das ferramentas que a IoT proporciona visando gerar aumento de produtividade para a indústria. A implantação da indústria 4.0 gera novos modelos de negócio, integrando características singulares de cada caso de uso com a IoT. Como resultado, surgem aplicações como: cadeias de suprimento automatizadas, manufatura aditiva, produções descentralizadas entre outros [34][18]. Tais aplicações são capazes de melhorar significativamente diversos aspectos da indústria, reduzindo métricas como tempo de produção, tempo de entrega, custo e rastreabilidade [18][35].

Em contrapartida ao grande impacto positivo que a indústria 4.0 pode trazer para diversos ambientes, emergem novos desafios. Com cada vez maior frequência, dados considerados sensíveis, ou seja, dados pessoais ou dados dos quais o comprometimento possa ter impacto no mundo físico, são tratados por meio de sistemas IoT. Como resultado, o comprometimento de tais sistemas por meio de falhas ou ataques pode ter impactos significativamente maiores se comparados a sistemas convencionais. Desta forma, o comprometimento dos mesmos pode causar perdas financeiras e até mesmo colocar a vida humana em risco [1].

Como exemplo, em Junho de 2010, um *Worm* chamado Stuxnet explorou vulnerabilidades de dia zero para alterar as frequências de operação de uma usina nuclear iraniana [12]. Em 2016 foram executados mais de 15,000 ataques de negação de serviço por uma *botnet* chamada Mirai, que utilizou mais de 600k dispositivos IoT infectados para gerar tráfego na rede [3]. Mais recentemente, em 2019, uma família do Mississipi teve a

câmera de sua babá eletrônica hackeada, por meio de uma brecha de segurança em uma campainha eletrônica pertencente a empresa Amazon [39]. Os exemplos citados acima não são os únicos casos de falhas de segurança em dispositivos IoT que podem afetar a indústria 4.0, diversos trabalhos anteriores documentaram outras falhas graves relacionadas a segurança [35][25][3][24]. Adicionalmente, relatórios alertam sobre um crescente número de dispositivos IoT, acompanhados de um crescente número de falhas e vulnerabilidades [34][35][4].

Visando minimizar as questões relacionadas a privacidade e a segurança em ambientes IoT, incluindo aqueles relacionados a indústria 4.0, trabalhos relacionados têm buscado alternativas legislativas e tecnológicas para mitigar falhas de segurança [40]. Por um lado, legislações como o Regulamento Geral sobre a Proteção de Dados (do inglês, *General Data Protection Regulation* ou GDPR) e a Lei Geral de Proteção de Dados Pessoais (LGPD) estabelecem políticas para o tratamento de dados sensíveis. Estas regulamentações delimitam o acesso aos dados pessoais/empresariais que estão armazenados nos dispositivos pessoais ou equipamentos empresariais [29][31]. Além das questões relacionadas a legislação de proteção de dados, outros trabalhos exploram abordagens tecnológicas como virtualização, redes definidas por software (do inglês, *Software Defined Network* ou SDN), segurança sensível a contexto (do inglês, *Context-aware Security* ou CAS) e Blockchain, para tentar garantir a integridade de dados sensíveis e evitar que os componentes de software ou hardware de um sistema sejam comprometidos [10][9][15][19].

Assim como em outras soluções desenvolvidas para as diferentes verticais de negócios, as soluções para a indústria 4.0 também tem necessidade de garantir ou minimizar problemas relacionados a segurança. Questões de segurança relacionadas aos sistemas desenvolvidos para industria 4.0 tem gerado grandes preocupações, devido ao crescente aumento de dispositivos conectados que estão sendo instalados e devido a adoção de soluções descentralizadas, permitindo uma tomada de decisão mais ágil e reduzindo hierarquias organizacionais [18]. Contudo, as soluções de segurança atuais não foram projetadas para atuar em ambientes com estrutura descentralizada e com as restrições impostas, naturalmente, por um ambiente IoT. Ataques como Spoofing e Sybil, onde o atacante pode inserir requisições ou entidades fantasmas nos sistemas que se passam por requisições ou entidades reais [40][35], são bastante comuns em ambientes desta natureza. É possível enfatizar que existe a necessidade de explorar e desenvolver sistemas e arquiteturas de segurança voltadas para a proteção de ambientes descentralizados para a indústria 4.0. Considerando os fatos expostos, acredita-se que uma proposta que combine as abordagens de Blockchain e CAS seja capaz de atender esta demanda.

1.1 CENÁRIO MOTIVACIONAL

Uma arquitetura de segurança que garanta integridade de dados e processos em ambientes descentralizados pode ser aplicada a inúmeros cenários. O cenário destacado neste trabalho está relacionado a indústria 4.0 [40].

A escolha se deu devido a participação do grupo de pesquisa GSE no projeto FASTEN (do Inglês, *Flexible and Autonomous Manufacturing Systems for Custom-Designed Products*), financiado pela União Europeia através da chamada *IoT Pilots* dentro do programa *Horizon* 2020 e pela Rede Nacional de Pesquisas (RNP). Este projeto teve como parceiros as empresas Thyssenkrupp Elevadores e Embraer, além de diferentes universidades brasileiras e europeias. O projeto tem foco no desenvolvimento da indústria 4.0, mais especificamente no conceito de produção aditiva descentralizada. O projeto consiste na utilização de robôs e impressoras 3D geograficamente distribuídas em unidades de manufatura aditiva (do inglês, *Additive Manufacturing Unit* ou AMU), visando melhorar tempos de produção e entrega de partes sobre demanda. Os testes foram realizados na Thyssenkrupp Elevadores e os resultados preliminares demostram que o processo pode otimizar tempos de entrega de partes de 90 para 4 dias.

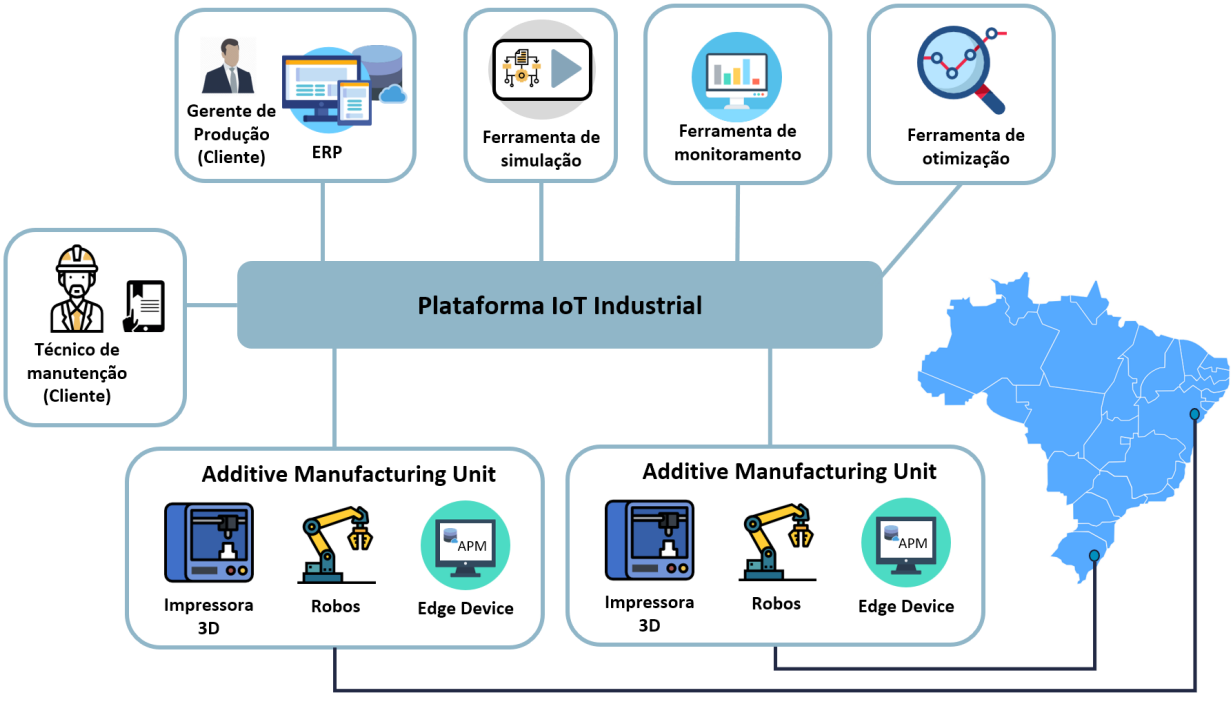

Figura 1.1 – Arquitetura do Projeto FASTEN.

Como pode ser observado na Figura 1.1, o projeto contempla uma plataforma IoT voltada para o meio industrial. O desenvolvimento desta plataforma é um dos objetivos do projeto, sendo ela responsável por conectar as ferramentas de monitoramento, simulação e otimização, visando monitorar e otimizar ordens de impressão. Tais ordens são fornecidas

pelo sistema de gestão empresarial (do inglês, *Enterprise Resource Planning* ou ERP) e técnicos de manutenção. As ordens são escalonadas para diferentes AMUs que correspondem a centros de impressão aditiva mantidos pela empresa ou por empresas autorizadas, contratados para a impressão de peças sob demanda.

Considerando este cenário e os níveis de segurança adotados pelas diferentes empresas parceiras, é possível que uma falha de segurança permita a instanciação de uma AMU maliciosa (AMU *ghost*) que poderia receber ordens legítimas e não executar as mesmas, impedindo que a peça seja produzida. Da mesma forma, um cliente malicioso poderia gerar tráfego fantasma, sobrecarregando uma AMU ou replicando ordens para sobrecarregar a rede. Pode-se destacar, que em um caso mais extremo, um atacante poderia até mesmo realizar um ataque de Spoofing entre a comunicação das AMUs e da Plataforma, sendo capaz de simular uma segunda plataforma maliciosa e enviar ordens adulteradas para diferentes AMUs [13][24].

Neste cenário, abordagens que utilizam criptografia e SDN tornam-se ineficazes. Dado que ordens maliciosas podem trafegar pelo sistema acima da camada protegida por estas abordagens [15][24]. Em outras palavras, com uma entidade da rede comprometida, a mesma pode produzir tráfego malicioso antes do emprego de qualquer medida criptográfica. Estes e outros problemas não são referentes apenas ao caso de uso descrito, mas também a diversos outros casos que utilizam modelos descentralizados ou colaborativos [41][20][34].

A motivação deste trabalho é fortemente relacionada à necessidade de prover segurança para casos de uso baseados em descentralização. A indústria 4.0, segundo trabalhos anteriores, têm apresentado uma tendência à descentralização e distribuição [18][35]. Desta forma, a busca por garantir a integridade de dados e processos em sistemas descentralizados para a indústria 4.0 emerge como a principal motivação do presente trabalho. É válido destacar que buscar meios de compartilhar informações de contexto de modo seguro também é um desafio discutido em trabalhos anteriores e abordada ao longo do presente trabalho [42].

1.2 CONTRIBUIÇÃO

As principais contribuições do presente trabalho podem ser resumidas em:

- A definição de uma arquitetura de segurança descentralizada para garantir integridade de dados e processos em ambientes IoT, especificamente focada para a indústria 4.0. Tal arquitetura faz uso das tecnologias de Blockchain e segurança sensível a contexto.
- A implementação e validação da arquitetura de segurança proposta. A implementação foi realizada utilizando como base a distribuição de Blockchain Sawtooth da

organização Hyperledger [27]. Adicionalmente, também foi implementado um módulo de processamento de regras, utilizando como exemplo a linguagem FROST, além da utilização de algumas ferramentas do projeto FIWARE [37][6].

- A instanciação da arquitetura proposta em um caso de uso real de indústria 4.0, em um projeto de pesquisa com cooperação entre organizações brasileiras e europeias.
- A documentação do estado da arte atual sobre segurança em IoT, Blockchain e segurança sensível a contexto.

1.3 OUTLINE

O restante desta dissertação é organizada da seguinte forma. O Capítulo 2 apresenta o referencial teórico de segurança considerado para o trabalho. O Capítulo 3 apresenta o referencial teórico das tecnologias que formam a arquitetura. O Capítulo 4 apresenta a arquitetura proposta em detalhes, incluindo a implementação da mesma. O Capítulo 5 apresenta os testes realizados para a validação da mesma. Por fim, o Capítulo 6 finaliza o trabalho, discutindo os resultados e documentando futuras direções.

2. INTERNET DAS COISAS E SEGURANÇA

Para melhor compreender a relação entre segurança e IoT, este capítulo apresenta uma arquitetura IoT dividida em camadas e possíveis ameaças a serem realizadas 2.1. A seção 2.2 apresenta a relação entre as ameaças e os requisitos de segurança. Na seção 2.3 é apresentada uma comparação entre os requisitos de segurança atendidos pelos trabalhos relacionados e pela arquitetura proposta. E por fim, na seção 2.4, discute-se algumas considerações à cerca dos temas apresentados neste capítulo.

2.1 ARQUITETURA E SEGURANÇA EM SISTEMAS IOT

Diversos trabalhos organizam a IoT por meio de diferentes arquiteturas [17][4][34]. Trabalhos relacionados com foco em segurança de modo geral, i.e. não necessariamente direcionados para IoT, utilizam com frequência um modelo baseado em camadas, diferenciando superfícies vulneráveis a ataques [11]. Visando estabelecer a relação entre segurança e IoT, optou-se por considerar uma arquitetura IoT também baseada em camadas [17], classificadas como:

- **Camada de Percepção:** Responsável por realizar a interação entre sistemas computacionais e o mundo físico. Esta camada é composta por sensores e atuadores que capturam dados do mundo físico ou realizam tarefas no mesmo. Exemplos destes tipos de dispositivos são: identificadores de radiofrequência (do inglês, *Radio-frequency identification* ou RFID), impressoras 3D e termômetros.
- **Camada de Transporte:** Responsável por estabelecer a comunicação entre as camadas de percepção e de aplicação. Esta camada faz uso de redes de comunicação, como por exemplo Wi-Fi, 5G, LTE, LoRa, SigFox e NB-IoT.
- **Camada de Aplicação:** Responsável por representar a lógica da aplicação e por comunicar com o usuário final do sistema. Esta camada pode ser diretamente relacionada a um ambiente/aplicação, e.g. uma fábrica ou fazenda inteligente. Vale destacar que plataformas de nuvem e plataformas IoT também estão inclusas nesta camada [17].

Cada camada de sistemas IoT contém diferentes características e necessidades relacionadas à segurança. Por exemplo, a camada de percepção está mais vulnerável a ameaças como Revelação de informação [11] devido a proximidade física de um possível indivíduo mal intencionado. Em comparação, a camada de rede apresenta maior vulnerabilidade a ameaças como negação de serviço (do inglês, *Denial of Service* ou DoS) devido

a possíveis limitações de infraestrutura. Da mesma forma, pode-se dizer que a camada de aplicação encontra-se mais vulnerável a ameaças como *Spoofing* [17] devido ao gerenciamento de identidades encontrar-se na mesma. Apesar de cada camada apresentar vulnerabilidades específicas, destaca-se alguns tipos de ataques que se constituem em desafios comuns entre todas camadas, sendo alguns dos mais relevantes [14]:

- ***Spoofing*:** Utilização de credenciais de terceiros para assumir uma identidade diferente em um sistema, usualmente utilizado para obter acesso a conteúdo restrito.
- ***Tampering*:** Realizar alteração de dados para montar um ataque, usualmente utilizado para causar desordem em alguma parte do sistema.
- **Repúdio:** Uma entidade realiza uma ação e não é possível provar que o mesmo realizou tal ação, usualmente utilizado para esconder a identidade de um indivíduo mal intencionado.
- **Revelação de informação:** A revelação de informações restritas para uma entidade que não tem permissão de acesso, usualmente utilizado visando ganhar informações sobre determinado funcionamento do sistema.
- **Negação de serviço:** Impedir que usuários válidos acessem determinados recursos, usualmente realizado sobrecarregando alguma estrutura do sistema.
- **Elevação de privilégio:** Usuário com determinado privilégio de acesso que altera suas permissões para elevar seu nível de acesso, usualmente utilizado para obter informações restritas.

2.2 REQUISITOS DE SEGURANÇA PARA IOT

Discutir segurança em IoT com base em ataques fornece uma visão precisa para identificação de vulnerabilidades ou instanciação em determinado caso de uso. Porém, dada a grande variedade de tipos de ameaças e de diferentes superfícies vulneráveis, ao comparar diferentes arquiteturas e sistemas, convém conduzir a discussão sobre segurança utilizando alguns requisitos como base [14]. Através da análise do estado da arte, é possível identificar quais requisitos protegem os sistemas de quais ameaças, e, posteriormente, agrupar os requisitos de segurança atendidos em cada camada de determinado sistema. A partir da análise, os principais requisitos identificados são:

- **Identificação:** Capacidade do sistema de identificar ou reconhecer identidades externas antes de uma interação.

- **Autenticação:** Capacidade do sistema de verificar se a identidade informada por um indivíduo é legítima.
- **Autorização:** Capacidade do sistema de delegar privilégios de acesso a determinados indivíduos.
- **Detecção de ataques:** Capacidade do sistema de detectar se algum ataque aconteceu ou está acontecendo.
- **Não-repúdio:** Capacidade do sistema de provar o autor de um dado ou ação.
- **Integridade de dados:** Capacidade do sistema de proteger os dados de uma alteração indevida.
- **Auditabilidade:** Capacidade do sistema de verificar se um processo está sendo realizado de acordo com o estabelecido previamente.
- **Anonimidade:** Capacidade do sistema de garantir que a identidade dos indivíduos não seja pública.
- **Confidencialidade:** Capacidade do sistema de manter determinada informação apenas sobre domínio de indivíduos autorizados.
- **Acusação:** Capacidade do sistema de acusar um indivíduo de ter realizado um procedimento.
- **Disponibilidade:** Capacidade do sistema de garantir disponibilidade e o acesso à uma informação sempre que necessário.

Tabela 2.1 – Relação entre requisitos de segurança e ameaças.

Requisitos/Ameaças	*Spoofing*	*Tampering*	Não-repúdio	Revelação	Negação de serviço	Elevação de privilégio
Identificação			✓			
Autenticação	✓					✓
Autorização				✓		✓
Detecção de ataques	✓				✓	
Não-repúdio			✓			
Integridade de dados		✓				
Anonimidade				✓		
Confidencialidade				✓		
Acusação			✓			
Disponibilidade					✓	

O atendimento dos requisitos de segurança corresponde diretamente a capacidade de determinado sistema ou arquitetura de mitigar ou evitar determinados tipos de ameaça [14]. A Tabela 2.1 demonstra a relação entre os requisitos e as ameaças apresentadas, isto é, quais requisitos são capazes de mitigar quais ameaças: **identificação** pode mitigar ameaças de não repúdio; **Autenticação** pode mitigar ameaças de *Spoofing*

e Elevação de privilégio; **Autorização** pode mitigar ameaças de Revelação e Elevação de privilégio; **Detecção de ataques** pode mitigar ameaças de *Spoofing* e Negação de serviço; **Não-repúdio** pode mitigar ameaças de repudiação; **Integridade de dados** pode mitigar ameaças de *Tampering*; **Anonimidade** pode mitigar ameaças de Revelação; **Confidencialidade** pode mitigar ameaças de Revelação; **Acusação** pode mitigar ameaças de Repudiação; e **Disponibilidade** pode mitigar ameaças de Negação de serviço.

2.3 TRABALHOS RELACIONADOS

A necessidade de pesquisar a respeito de meios de aumentar a segurança em ambientes IoT já foi discutida em diversos trabalhos anteriores [40][35][4][21][32][38]. Alguns trabalhos propõe arquiteturas ou mecanismos fortemente relacionados ao proposto no presente trabalho, isto é, propostas de garantir segurança de modo descentralizado [28][43][36] [33][16]. Para fundamentar a abordagem proposta, foram utilizados meios sistemáticos de comparação [35][14]. A Tabela 2.2 agrupa os trabalhos analisados, utilizando os requisitos de segurança discutidos na Seção 2.2, considerando as camadas de Aplicação (A) e Transporte (T) discutidas na seção 2.1.

O trabalho desenvolvido por Ouechtati et al. [28] propõe o desenvolvimento de um *middleware context-aware*. Este *middleware* possibilita que diferentes ambientes IoT estabeleçam conceitos independentes do que é um contexto seguro. Desta forma, a rede de nodos que utiliza tal *middleware* estabelece um sistema de recomendação. De acordo com o sistema de recomendação, diferentes políticas de segurança são executadas, alcançando requisitos relacionados a confidencialidade e controle de acesso.

O trabalho desenvolvido por Matos et al. [21] propõe uma arquitetura que visa prover CAS por meio de compartilhamento de contexto. A arquitetura possibilita que dispositivos heterogêneos compartilhem informações de diferentes granularidades entre diferentes camadas da IoT. Como resultado, dispositivos com diferentes características de hardware e software podem compartilhar informações permitindo que a interpretação dos dados é comum entre todas entidades participantes da rede. Desta forma, o trabalho alcança requisitos relacionados a autenticação e controle de acesso.

O trabalho desenvolvido por Wright et al. [43] propõe o uso de Blockchain e contratos inteligentes para estabelecer confiança em cadeias de suprimento de ambientes IoT. O uso da Blockchain como entidade verificadora em cadeias de suprimento não é novo, sendo um dos casos de uso mais bem sucedidos da tecnologia [34]. O trabalho em específico, realiza uma análise sobre a criação de marcas de posse na Blockchain, simbolizando o local físico de cada produto. Como resultado, a Blockchain mantém um histórico integro de todos os lugares que determinado produto ou peça foi fisicamente armazenada, atuando

simultaneamente como um certificado de validade e alcançando requisitos relacionados a não-repúdio e disponibilidade.

A pesquisa realizada por Fremantle et al. [16] propõe a utilização de Blockchain como um terceiro confiável envolvido, atuando como um consultor descentralizado. Especificamente, a aplicação proposta consiste em implantar uma Blockchain separada da aplicação em questão. Deste modo a Blockchain é consultada para a adição de determinado dado na rede, e posteriormente consultada para verificar a integridade do mesmo dado. Deste modo, aplicações podem explorar a integridade provida por meio da Blockchain sem impactar no consumo de recursos, alcançando requisitos relacionados a Auditabilidade e Não-repúdio.

A pesquisa realizada por Polyzos e Fotiou [32] adapta a criptografia utilizada na Blockchain para o compartilhamento de informações entre dispositivos IoT de forma descentralizada. É proposta uma arquitetura que cria uma infraestrutura de chaves públicas (do inglês, *Public-key Infrastructure* ou PKI) sobre uma rede Blockchain composta por nodos que representam *gateways* responsáveis por conjuntos de dispositivos. Desta forma, a arquitetura garante que mesmo que uma parte da rede seja comprometida, o restante continua em operação, alcançando requisitos como integridade de dados e disponibilidade.

O trabalho desenvolvido por Sharma et al. [36] propõe uma arquitetura chamada DistBlockNet que combina as tecnologias de Blockchain e SDN. A DistBlockNet consiste na utilização de uma Blockchain para atualizar de forma descentralizada e integra tabelas de regras utilizadas para detecção de ataques em SDN. Como resultado, a utilização desta arquitetura cria nodos que mantém a integridade do tráfego de rede de forma distribuída, provendo requisitos como Confidencialidade e Detecção de ataques.

O mecanismo desenvolvido por Venkatapathy [38] utiliza a tecnologia de Blockchain para implementar um *Broker* descentralizado de contexto. Especificamente, cada instância do *Broker* corresponde a um nodo da Blockchain. Quando uma entidade deseja publicar algum dado, é realizada uma transação na Blockchain, enquanto um assinante de determinado dado tem acesso direto aos dados publicados na Blockchain. Desta forma, o mecanismo garante a integridade dos dados distribuídos entre as diferentes instâncias do *Broker* e a disponibilidade dos mesmos.

O trabalho desenvolvido por Mohanty [33] consiste do desenvolvimento de um algoritmo de consenso leve, chamado *Proof-of-Authentication*, que utiliza um modelo de autenticação como algoritmo de consenso. Especificamente, o trabalho propõe um modelo criptográfico que utiliza chaves públicas e privadas para autenticação de Blocos da rede. Desta forma, tornando possível a utilização de uma Blockchain para garantir integridade na troca de dados entre dispositivos IoT com recursos restritos, alcançando requisitos como Disponibilidade e Integridade.

Ao analisar a Tabela 2.2, pode-se observar que trabalhos que tem foco direto na utilização da tecnologia de CAS [28][21] usualmente apresentam o atendimento de requi-

Tabela 2.2 – Trabalhos relacionados e requisitos de segurança.

	Presente Trabalho		[28]		[21]		[43]		[16]		[32]		[36]		[38]		[33]	
	A	T	A	T	A	T	A	T	A	T	A	T	A	T	A	T	A	T
Identificação			✓	✓	✓	✓					✓							
Autenticação	✓		✓	✓	✓	✓											✓	✓
Autorização	✓	✓	✓	✓	✓	✓					✓	✓		✓	✓	✓		✓
Detecção de Ataques	✓		✓	✓									✓	✓				
Não-repúdio	✓	✓					✓	✓	✓		✓	✓			✓	✓		✓
Integridade de Dados		✓					✓		✓		✓	✓		✓	✓	✓	✓	✓
Auditabilidade		✓	✓				✓	✓	✓	✓				✓				✓
Anonimidade		✓			✓		✓	✓	✓									
Confidencialidade	✓		✓	✓	✓	✓			✓				✓	✓	✓	✓		
Acusação					✓													
Disponibilidade	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓

sitos relacionados a autenticação e confidencialidade. Em contrapartida, apresentam uma necessidade por requisitos como Não-repúdio e Integridade de dados, necessidade já discutida em trabalhos anteriores [9]. Em comparação, trabalhos com foco direto na utilização da tecnologia de Blockchain [43][16] usualmente apresentam o atendimento dos requisitos relacionados à integridade, disponibilidade e auditabilidade. Em contrapartida, apresentando uma necessidade por requisitos relacionados a autenticação e detecção de ataques.

O presente trabalho, que combina as abordagens de Blockchain e CAS, consegue alcançar um número maior de requisitos se comparado diretamente a trabalhos que exploram o uso de apenas uma abordagem ou outra. Porém, tal combinação tem como custo o atendimento de alguns requisitos em diferentes camadas. Por exemplo, o requisito de confidencialidade na camada de transporte pode ser atingido por meio da utilização de CAS, porém, ao combinar com a tecnologia de Blockchain, características de implementação da mesma (discutidas no Capítulo 3), não permitem que tal requisito seja alcançado nesta camada. Assim como em outros trabalhos relacionados que buscam a combinação de tecnologias ou desenvolvem arquiteturas específicas [32][36][38][33], são alcançados requisitos específicos em diferentes camadas, apresentando características singulares.

2.4 CONSIDERAÇÕES

Neste capítulo foram apresentadas algumas das principais ameaças para sistemas de IoT e consequentemente para a indústria 4.0 e aplicações descentralizadas. Apresentou-se os requisitos de segurança mais relevantes relacionados a cada ameaça, e posteriormente utilizando uma organização sistemática para comparar a arquitetura proposta com outros trabalhos da atualidade. Trabalhos com foco em uma única abordagem tecnológica apresentaram lacunas relacionadas ao atendimento de certos requisitos. Em contrapartida, o presente trabalho e algumas propostas diferenciadas, como a combinação das tecno-

logias de Blockchain e SDN [36], buscam abranger um maior número de requisitos em algumas camadas relevantes aos respectivos casos de uso.

3. ABORDAGENS TECNOLÓGICAS

Este capítulo apresenta as principais tecnologias utilizadas para o desenvolvimento da arquitetura proposta. Apresenta a tecnologia de Blockchain e os conceitos que compõe a mesma (Seção 3.1). Explica o conceito de contexto e relaciona com CAS (Seção 3.2). E por fim, discute relação das tecnologias com segurança.

3.1 BLOCKCHAIN

A tecnologia de Blockchain tem sua autoria desconhecida, tornando-se popular em 2008 devido a implementação e criação da criptomoeda bitcoin pelo Pseudônimo Satoshi Nakamoto [26]. Atualmente, a tecnologia tem sido empregada em diversos casos de uso, sendo discutida em trabalhos anteriores como uma tecnologia chave para o desenvolvimento de setores como a indústria 4.0 [34] [19]. Seu destaque ocorre principalmente devido a capacidade da tecnologia de garantir integridade e privacidade dos dados.

A tecnologia de Blockchain consiste de uma estrutura de dados descentralizada que replica os dados entre todos os integrantes de uma rede *Peer-to-peer* (P2P) [10]. Cada integrante da rede é chamado de nodo e representa um dispositivo que contém a estrutura da Blockchain. Tal estrutura é composta por: Transações, que representam um conjunto mínimo de dados na Blockchain; e Blocos, que são estruturas responsáveis por armazenar e agrupar conjuntos de transações em forma de uma lista encadeada.

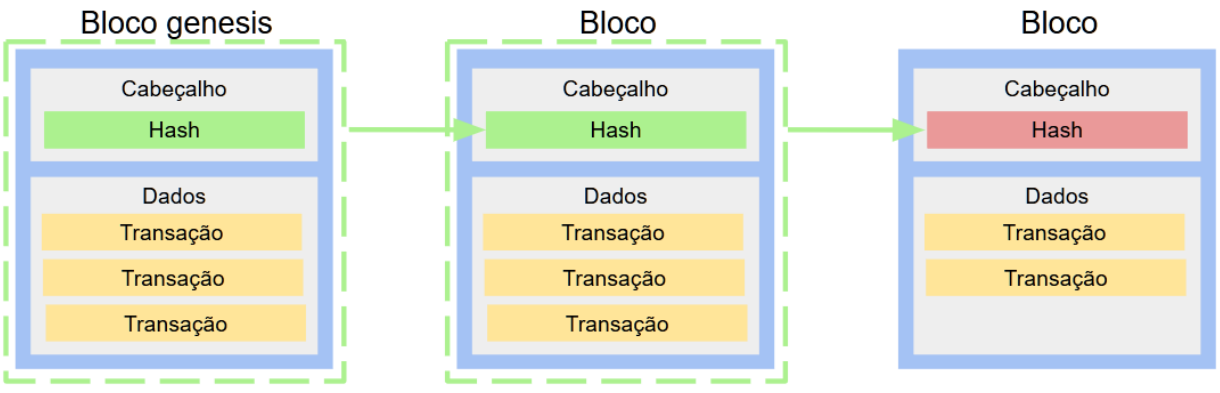

Figura 3.1 – Lista encadeada de blocos.

A Figura 3.1 ilustra a organização em lista encadeada dos Blocos na Blockchain. Cada bloco contém os dados armazenados em transações e um cabeçalho. O cabeçalho armazena um hash gerado a partir de todo conteúdo do Bloco anterior, usualmente utilizando o algoritmo SHA256 [26]. Quando um bloco atinge uma quantidade de transações

definidas pela rede, é gerado um novo bloco contendo o hash do bloco anterior, formando uma lista encadeada que é replicada em todos nodos participantes da rede.

Desta forma, nenhum dado de alguma transação anterior ao bloco mais recente pode ser alterado. Pois resultaria na inconsistência do encadeamento de hashs da lista encadeada. Devido a todos os nodos conterem uma replica da Blockchain, os mesmos tem autonomia para verificar a consistência da lista encadeada, consequentemente garantindo a integridade da lista replicada em cada nodo. Por meio deste processo, a Blockchain alcança uma de suas características fundamentais, a imutabilidade, garantindo que nenhum dos nodos que compõe a rede possa alterar um dado previamente aceito pela mesma.

Uma Blockchain podem ser classificada em duas principais categorias mais bem aceitas pela literatura quanto a composição e domínio da rede [19][20]:

- **Pública ou *Permissionless*:** nenhuma organização mantém controle da rede, permitindo que qualquer indivíduo que deseja participar da rede possa ingressar na mesma.
- **Privada ou *Permissioned*:** uma organização mantém controle da rede, e, portanto, apenas indivíduos conhecidos pela organização podem ingressar na mesma.

3.1.1 ALGORITMOS DE CONSENSO

Considerando que a Blockchain é composta por uma rede descentralizada com diferentes nodos, existe a necessidade de haver um consenso para a criação de novos blocos. A literatura apresenta diferentes algoritmos que podem ser utilizados para estabelecer o consenso, sendo os mais comumente utilizados [7][27]:

- *Practical Byzantine Fault Tolerance* (pBFT): Consiste em um sistema de votos utilizado apenas em Blockchains privadas. Qualquer nodo da rede pode sugerir um novo bloco que é avaliado pelos demais nodos. Quando um bloco é aceito por 2/3 da rede ele é distribuído para os demais nodos que o aceitam, adicionado o mesmo a lista encadeada.
- *Proof of Work* (PoW): Consiste em um sistema de competição que pode ser utilizado em Blockchains públicas ou privadas. Qualquer nodo da rede pode participar da competição, que consiste da busca por um valor hash definido pela rede Blockchain. O primeiro nodo a encontrar tal valor, tem o direito de publicar o bloco, que é distribuído para os demais nodos que o aceitam, adicionado o mesmo a lista encadeada.
- *Proof of Stake* (PoS): Consiste em um sistema de hierarquia utilizado em Blockchain públicas ou privadas. As chances de um nodo gerar um novo bloco são proporcionais a sua hierarquia na rede. Usualmente é utilizado em implementações de moeda digital

por relacionar a quantidade de moeda digital que o nodo possui como a posição do nodo na hierarquia. De outra forma, quanto mais moedas o nodo possuir, maior é a chance do mesmo gerar o próximo bloco da lista encadeada.

- *Proof of Elapsed Time* (PoET): Consiste em um sistema de sorteio utilizado em Blockchains públicas ou privadas. Um tempo de espera aleatório é gerado para cada nodo da rede, o nodo que receber o menor tempo de espera ganha o sorteio. De outra forma, o primeiro nodo que aguardar pelo tempo de espera sorteado tem o direito de publicar o próximo bloco da rede, que é aceito pelos demais nodos. Devido a tal algoritmo ser utilizado no presente trabalho, mais detalhes são descritos no A.

Os algoritmos de consenso desempenham um papel fundamental nas Blockchains, influenciando diretamente nas características da rede tais como: consumo de recursos, escalabilidade e segurança [8]. O algoritmo de PoW é o mais difundido na comunidade científica, surgindo com a implementação do Bitcoin [26]. Este algoritmo é eficiente em alcançar consenso, porém apresenta um grande gasto de recursos computacionais. Algoritmos baseados em voto, como pBFT e *Federated Byzantine Agreement* (FBA), mitigam o grande custo computacional do PoW ao custo de baixa escalabilidade e alto consumo de banda de rede. Paralelamente, alguns algoritmos apresentam características únicas que convém a determinados casos de uso. Como exemplo, o algoritmo PoS apresenta um equilíbrio entre gasto computacional e latência na rede, porém convém apenas a aplicações financeiras. Por outro lado, abordagens baseadas em sorteio, como o PoET, usualmente apresentam um equilíbrio entre escalabilidade e consumo de recursos, às custas de não atender algumas premissas de descentralização para Blockchains públicas.

3.1.2 SMART CONTRACTS

As transações adicionadas na Blockchain podem conter qualquer conteúdo. Trabalhos anteriores já quantificaram o conteúdo presente na Blockchain utilizada pela implementação Bitcoin, identificando diversos formatos de dados que incluem até mesmo arquivos de imagens e áudio [23]. Neste sentido, a Blockchain pode ser extremamente útil para diversos casos de uso. Desta forma, os nodos participantes da rede podem garantir a integridade de algum procedimento acordado entre os mesmos por meio de um código armazenado na Blockchain, também chamado de *Smart Contract*.

No contexto de Blockchain, um *Smart Contract* consiste de um contrato entre os nodos da rede que têm a garantia de execução sem a necessidade de um terceiro confiável. Para isto, cada nodo da Blockchain conta com uma máquina virtual (do inglês, *Virtual Machine* ou VM) e um código armazenado em uma transação. Cada VM contém uma referência que serve de endereço para a transação que armazena o código do *Smart Contract*.

Para execução do mesmo, é enviada uma transação para o respectivo endereço, cujo conteúdo serve como entrada para execução do código. Ao receber uma entrada para o *Smart Contract*, cada VM executa automaticamente o respectivo código, que tem seu resultado validado entre os nodos utilizando o algoritmo de consenso presente na rede. Desta forma, é garantido que todos os nodos da rede tenham o mesmo resultado ao executar um *Smart Contract* [7].

Uma mesma rede Blockchain suporta múltiplos *Smart Contracts* com diferentes características. Os mesmos podem ser escritos utilizando linguagens de programação de alto nível, como Rust e derivados de Ecmascript, que variam de acordo com a implementação da Blockchain [27]. Adicionalmente, o uso de Blockchain tem sido discutido por diversas organizações, como por exemplo o NIST [2], como uma tendência de uso em paradigmas como a IoT. Ainda, os *Smart Contracts* podem ser utilizados em diversas aplicações, que incluem não somente a trocas de criptomoedas, mas cadeias de suprimento automatizadas, transações *machine-to-machine* autônomas, certificações digitais, entre outras [19].

3.2 SEGURANÇA SENSÍVEL A CONTEXTO

Para compreender o que é segurança sensível a contexto (do inglês, *Context-aware Security* ou CAS) é necessário compreender o conceito de contexto e sensibilidade a contexto (do inglês, *Context Awareness*), para depois estabelecer a relação entre ambos termos. Compreende-se como contexto qualquer informação que caracterize o estado de alguma entidade e que seja facilmente interpretável [22]. Uma entidade pode ser uma pessoa, um lugar, um dispositivo ou um objeto relevante para o usuário ou a aplicação.

```
{
    “AMUId”:”1010”,
    “local”:”Porto Alegre”,
    “data”:”14:10:2019”,
    “hora”:”15:03:45”,
    “dispositivoTipo”:”impressora”,
    “dispositivoId”:”023”,
    “status”:”imprimindo”,
    “itemId”:”a133”
}
```

Figura 3.2 – Exemplo de contexto em formato JSON.

A Figura 3.2 apresenta um exemplo de contexto em formato JSON, considerando o caso de uso descrito na Subseção 1.1. Isto é, as informações presentes no arquivo JSON caracterizam o estado de uma impressora 3d. O exemplo inclui informações como a AMU em que a impressora esta alocada, o status de operação da mesma, o item a ser impresso e

a hora que a mesma iniciou a impressão. Uma informação de contexto pode ser classificada em duas principais categorias [10]:

- **Primário:** Contexto obtido sem a necessidade de relacionar informações de diferentes fontes, por exemplo, uma informação recebida por meio de uma API ou dados recolhidos por sensoriamento.
- **Secundário:** Contexto obtido ao relacionar duas ou mais informações, por exemplo, a diferença de tempo entre dois contextos ou uma consulta em um banco de dados.

Uma aplicação *Context-aware* utiliza informações de contexto para prover serviços, tomar decisões ou adaptar-se. Tais aplicações determinam *porque* algo está acontecendo, com base em informações relacionadas a *quem, onde, quando* e *o que* [30]. Por meio desta abordagem, sistemas *Context-aware* apresentam forte integração com o mundo físico.

A CAS consiste da utilização do paradigma de *Context-awareness* com viés de fornecer segurança para um sistema ou ambiente [5], ou seja, consiste em validar se determinada informação ou processo apresenta alguma inconsistência ou acesso não autorizado. Duas ferramentas principais apresentam forte impacto na CAS e são discutidas nas subseções seguintes, o compartilhamento de contexto (do inglês, *Context sharing*) e os modelos de decisão (do inglês, *Reasoning models*).

3.2.1 COMPARTILHAMENTO DE CONTEXTO

O compartilhamento de contexto consiste da capacidade de um sistema sensível a contexto de compartilhar o próprio contexto com outros sistemas ou dispositivos interessados. Tal capacidade está fortemente relacionada ao ciclo de vida do contexto. Conforme demonstrado na Figura 3.3, o ciclo de vida do contexto em um sistema *Context aware* é composto pelas seguintes etapas [30]:

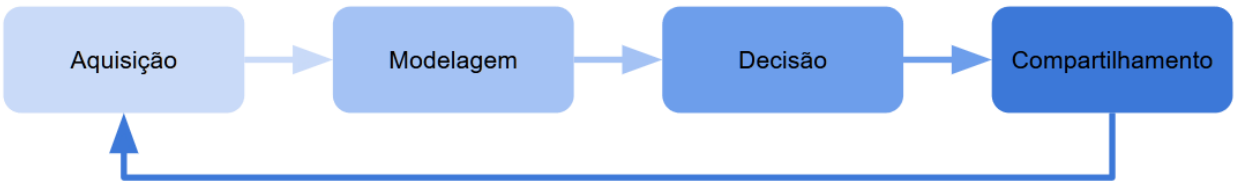

Figura 3.3 – Ciclo de vida do contexto.

- **Aquisição:** Etapa em que a aplicação adquire o contexto primário, realizando sensoriamento ou recebendo o contexto de outras aplicações relacionadas.

- **Modelagem:** Etapa em que ocorre a transformação de contexto primário para secundário, caso necessário. Pode ser classificada de dois modos: estática, gerando sempre um mesmo conjunto de dados; ou dinâmica, adaptando o conjunto de dados gerado conforme os dados recebidos da etapa de aquisição.
- **Decisão:** Etapa em que algum mecanismo é utilizado para uma tomada de decisão. Mais detalhes desta etapa são descritos na subseção seguinte.
- **Compartilhamento:** Etapa em que a aplicação compartilha com as partes interessadas o resultado da decisão tomada na etapa anterior e/ou o contexto atual da própria aplicação.

O compartilhamento de contexto possibilita a interoperabilidade de diferentes domínios de aplicações. Isto é, por meio de plataformas de compartilhamento de contexto, diferentes áreas como saúde, indústria e transporte podem interagir para melhorar a eficiência de processos e a qualidade de serviços [10]. Paralelamente, o compartilhamento de informações de contexto necessita de mais cuidados se comparado ao compartilhamento de dados usuais, uma vez que dados usuais não necessariamente apresentam um formato interpretável, enquanto informações de contexto podem ser facilmente compreendidas. Por este motivo, informações de contexto podem conter informações de grande valor, e, portanto, apresentam maior necessidade de garantir segurança, já discutida em diversos trabalhos anteriores [22][9][42].

3.2.2 MODELOS DE DECISÃO

Um modelo de decisão pode ser compreendido como um meio de adquirir mais conhecimento ao analisar determinada informação de contexto [30]. Os modelos de decisão representam uma das etapas mais importantes da CAS, pois determinam características como eficiência e desempenho [5]. Atualmente, existem uma variedade de modelos de decisão disponíveis, alguns dos principais são [10][22][30]:

- **Regras:** Consiste do estabelecimento de regras que são aplicadas sobre uma ou mais informações de contexto para inferir algum conhecimento. Por exemplo, SE determinada informação ENTÃO executar procedimento ’A’ SENÃO executar procedimento ’B’.
- **Aprendizado de máquina:** Consiste da aplicação de algoritmos de aprendizado de máquina sobre um conjunto de informações de contexto para inferir determinado comportamento. Por exemplo, a aplicação de algoritmos de agrupamento, Naïve Bayes e Árvores de decisão.

- **Lógica difusa:** Consiste do uso de expressões lógicas sobre informações de contexto para deduzir determinado fato. Por exemplo, se determinado objeto não está seco e não está molhado, logo está úmido.

- **Ontologias:** Consiste da aplicação de modelos ontológicos para estabelecer a relação entre as informações de contexto e determinado fato. Por exemplo, o estabelecimento de variáveis, grupos e hierarquias.

Cada modelo de decisão contém características particulares. Regras são adequadas para identificação de eventos e apresentam bom desempenho. Em contrapartida, não são adequadas para aplicações muito complexas. O uso de Algoritmos de aprendizado de máquina é eficiente para detecção de comportamentos e padrões. Porém, é uma abordagem computacionalmente custosa, que por vezes apresenta problemas de desempenho. Abordagens de lógica difusa são adequadas para detectar estados e classificar grupos. Entretanto, são por vezes imprecisas para detecção de eventos e apresentam grande complexidade de manutenibilidade. Ontologias apresentam grande precisão na análise de dados, sendo adequadas para modelos onde variações ínfimas de contexto devem ser consideradas para a tomada de decisão. Porém, apresentam pouca interoperabilidade e implantação custosa, se comparado a outras abordagens. Com as diferentes características dos modelos de decisão, sua escolha deve ponderar as necessidades de cada caso de uso.

3.3 ABORDAGENS TECNOLÓGICAS E SEGURANÇA

Quando a tecnologia de Blockchain emergiu foi compreendida por alguns acadêmicos como uma panaceia para segurança da informação. Porém, assim como outras abordagens, a tecnologia apresenta características pertinentes para alguns casos de uso [2]. Segundo análises da empresa Gartner, a tecnologia já ultrapassou seu período de expectativas altas [2]. Assim, atualmente já existe um mapeamento correto dos principais requisitos que podem ser alcançados pela Blockchain, sendo eles [10]:

- **Autorização:** Por meio da implementação de assinaturas digitais com criptografia assimétrica, somente indivíduos autorizados podem criar transações na rede.

- **Não-repúdio:** O modelo de chaves públicas e privadas utilizado atua como um certificado que prova que um transação foi realizada por determinada chave criptográfica.

- **Integridade de dados:** A lista encadeada utilizada em conjunto com a rede P2P e o algoritmo de consenso garantem que um dado não possa ser alterado sem o conhecimento de todos integrantes da rede.

- **Auditabilidade:** O modelo de privacidade alcançado ao utilizar uma Blockchain garante a transparência de todos procedimentos executados na rede, mantendo privada a identidade dos participantes.
- **Anonimidade:** O modelo criptográfico implementado garante que a identidade dos participantes da rede permaneça privada mesmo em redes públicas.
- **Disponibilidade:** Devido a todos nodos da rede conterem uma cópia da Blockchain, todos os dados estão sempre disponíveis para uma eventual consulta de informações.

O paradigma de CAS não é novo, e vem sendo discutido desde o início dos anos 90 [30]. O mesmo tem adquirido destaque nos últimos anos devido ao crescimento do paradigma de IoT e a grande disponibilidade de informações do mundo físico em tempo real. Além disto, a crescente disponibilidade de informações permite que a CAS explore sua onipresença no mundo físico, alcançando requisitos de segurança como [10]:

- **Identificação:** Ao analisar históricos de contexto, a CAS é capaz de identificar indivíduos ou processos.
- **Autenticação:** Ao analisar alterações de comportamento, a CAS é capaz de identificar se um indivíduo é quem realmente diz ser.
- **Detecção de ataques:** Ao analisar o contexto de um ambiente, a CAS é capaz de identificar alterações de comportamento que correspondam a um ataque.
- **Confidencialidade:** Ao restringir acesso a determinada informação somente sobre determinado contexto, a CAS é capaz de limitar informações a determinados domínios.
- **Acusação:** Utilizando compartilhamento de contexto, a CAS é capaz de identificar potenciais autores de determinado procedimento.

Como pode ser visualizado na Tabela 3.1, as tecnologias de Blockchain e CAS apresentam uma relação de complementariedade. Enquanto a Blockchain garante requisitos relacionados a proteção de dados e decentralização, a CAS prove requisitos relacionados a detecção de ataques e utilização de informações para raciocínio. Pode-se observar tal relação de complementariedade não somente ao analisar requisitos, mas também ao analisar características das tecnologias.

Diversos trabalhos anteriores já levantaram a discussão a respeito da necessidade de segurança das informações de contexto utilizadas para a CAS, especificamente relacionada a necessidade por integridade e disponibilidade [42][22][9]. A tecnologia de Blockchain por sua vez, tem como principal destaque o fornecimento dos requisitos de integridade e

Tabela 3.1 – Complementação de requisitos das tecnologias de Blockchain e CAS.

	Blockchain	CAS
Identificação		✓
Autenticação		✓
Autorização	✓	
Detecção de Ataques		✓
Não-repúdio	✓	
Integridade de Dados	✓	
Auditabilidade	✓	
Anonimidade	✓	
Confidencialidade		✓
Acusação		✓
Disponibilidade	✓	

disponibilidade. Simultaneamente, a Blockchain apresenta necessidade por meios de autenticação e detecção de ataques para os integrantes da rede que comportam os nodos Blockchain [19][20]. Características das quais são discutidas como ponto forte em abordagens de CAS. Assim, o desenvolvimento de uma arquitetura voltada a segurança que combine ambas abordagens, Blockchain e CAS, é capaz de melhorar significativamente a segurança de diversos ambientes e simultaneamente das tecnologias envolvidas. Acredita-se que a combinação de tais tecnologias pode suprir a necessidade de segurança de aplicações IoT e da indústria 4.0, por meio da descentralização das informações de contexto e de asserções heurísticas de processos utilizando tais informações como base.

4. ARQUITETURA PROPOSTA

É possível observar a necessidade de segurança em aplicações IoT e da indústria 4.0 (Seção 1) e o potencial de atingir os requisitos de segurança ao combinar as tecnologias de CAS e Blockchain apresentadas na Seção 3. Desta forma, o presente trabalho propõe o desenvolvimento de uma arquitetura de segurança que utiliza as abordagens de Blockchain e CAS. Como poderá ser visualizado ao longo da Seção 2.3 foi explorado um conjunto de trabalhos relacionados, e, no melhor de nosso conhecimento, a proposta do presente trabalho ainda não foi explorada em pesquisas anteriores.

A arquitetura proposta consiste do uso da tecnologia de Blockchain para armazenar e compartilhar informações de contexto, e do uso da tecnologia de CAS para tomar decisões de segurança com base nas informações armazenadas na Blockchain. Por meio desta abordagem, é estabelecida uma rede na qual todos os integrantes compartilham informações entre si de modo seguro. Cada integrante da rede tem a capacidade de utilizar tais informações seguras para validar se os processos a serem executados são consistentes. Desta forma, é criada uma camada extra de segurança, possibilitando que um conjunto de participantes de uma aplicação descentralizada estabeleçam comunicação para validar a integridade de processos a serem executados na mesma.

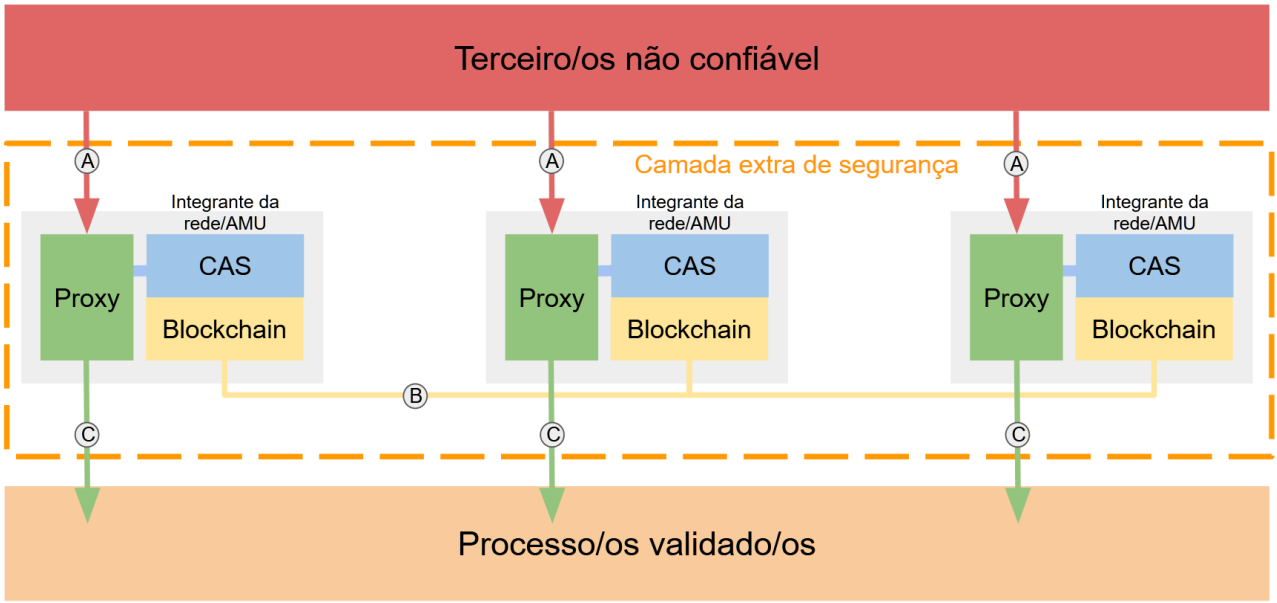

Figura 4.1 – Arquitetura proposta.

A Figura 4.1 ilustra a arquitetura proposta considerando uma aplicação descentralizada, como o caso de uso apresentado na Seção 1.1. Considera-se uma aplicação composta por uma rede de dispositivos ou ambientes inteligentes que recebem ordens e realizam processos. Para melhor compreensão do funcionamento da arquitetura, dividiu-se a mesma em três módulos: Proxy, Blockchain e CAS. Cada integrante da rede contém um

Proxy, um módulo CAS e um módulo Blockchain. As ordens recebidas (A) pelos integrantes da rede são geradas por terceiros não confiáveis (por exemplo, um sistema de ERP ou uma plataforma/middleware de um parceiro), e, consequentemente, podem estar comprometidas. Estas ordens são recebidas pelo módulo Proxy que requisita ao módulo CAS a validação da mesma. O módulo CAS verifica a validade das ordens quando requisitado, e para isto, consome informações do respectivo módulo de Blockchain. Tal módulo, por meio da rede Blockchain (B) armazena de forma integra o contexto atual e histórico de todos os integrantes da rede. Analisando tais informações de contexto em frente a ordem atual, o módulo CAS valida a ordem. Isto é, de acordo com o modelo de raciocínio adotado, o módulo classifica a ordem em válida ou inválida. Sendo que uma ordem válida apresenta conformidade com o modelo de raciocínio, e uma ordem inválida não. Ao receber um retorno do módulo CAS, o Proxy executa o respectivo processo caso a ordem seja válida (C) ou nega sua execução caso seja inválida.

4.1 Módulo de Blockchain

O módulo Blockchain em cada integrante da rede consiste de um nodo Blockchain e de métodos para interação com o respectivo nodo (por exemplo, adicionar e consultar informações de contexto). Considerando que todos os integrantes da rede executam uma aplicação descentralizada específica de interesse da empresa parceria neste trabalho, a Thyssenkrup Elevadores, a Blockchain utilizada é privada. Neste caso, a aplicação consiste em uma rede de centros de impressão 3D para manufatura aditiva composta por nodos no Brasil e América Latina. Cada nodo replica o modelo de arquitetura apresentado, sendo considerado um integrante da rede/AMU (figura 4.1). Mesmo sendo uma Blockchain privada, as questões de segurança se impõem, como por exemplo no caso de um integrante da rede ser clonado ou de uma ordem de impressão ser enviada em duplicidade para diferentes nodos ou para o mesmo nodo, ou ainda no caso de envio de ordem não solicitada (*ghost order*). Quanto ao algoritmo de consenso utilizado, ao ponderar características relacionadas a custo computacional, eficiência e implementação, discutidas na Subseção 3.1.1 e no A, concluiu-se que é adequada a utilização do algoritmo de Prova de tempo gasto (PoET).

Como pode ser visualizado na Figura 4.2, as informações de contexto são armazenadas na Blockchain por meio de transações. Também discutido na literatura como armazenamento *on-chain* [7], garante que cada nodo tenha acesso local as informações armazenadas na Blockchain. Desta forma, é garantido que o respectivo módulo CAS possa acessar informações de contexto históricas até mesmo sem ter acesso ao restante da rede.

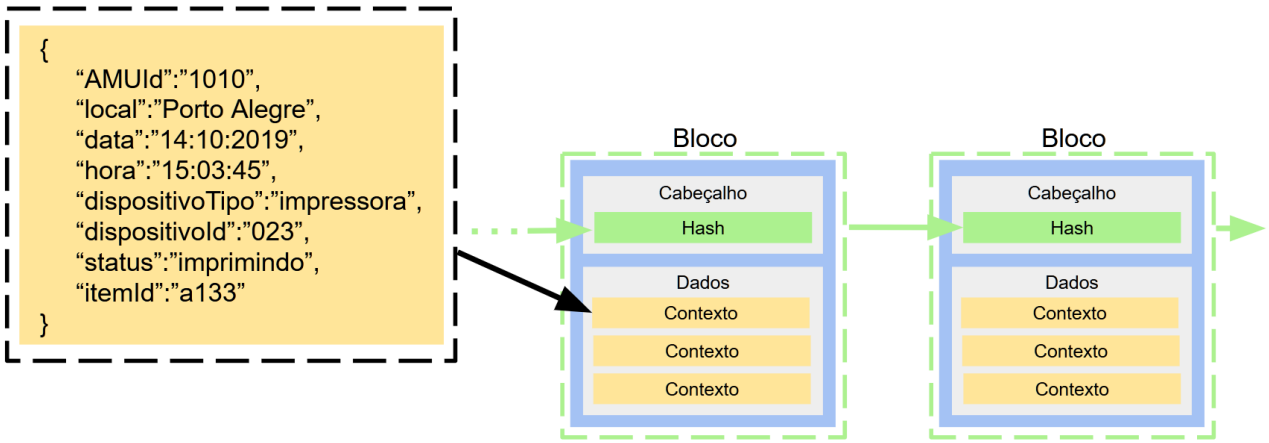

Figura 4.2 – Contexto armazenado na Blockchain.

4.2 Módulo de CAS

Cada integrante da rede possui um Módulo CAS que implementa um modelo de decisão. Para garantir a integridade do modelo de decisão adotado entre todos os integrantes da rede, tal módulo é implementado através de um Smart Contract. Ao considerar características relacionadas a desempenho, implementação e adequação para o caso de uso do exemplo descrito na Subseção 1.1, optou-se por utilizar um modelo de decisão baseado em regras. O modelo de regras adotado na implementação da arquitetura proposta compara o contexto atual do integrante da rede, que corresponde a ordem atualmente em processamento, com o contexto histórico armazenado na Blockchain. Tal contexto histórico inclui informações sobre ordens processadas por todos integrantes da rede, possibilitando a identificação de inconsistências na realização de processos.

A Figura 4.3 exemplifica o processo do modelo de decisão baseado em regras adotado pelo mecanismo proposto. Neste exemplo está sendo considerado o caso de uso descrito na Seção 1.1 e uma regra limitando a diferença de tempo entre impressões de uma AMU a 4 horas. Ao receber uma ordem de impressão A para validação, o mecanismo consulta no módulo Blockchain informações prévias de contexto (B, C e D). Logo, o modelo compara a informação A com as respectivas informações B,C e D obtidas da consulta. Ao comparar o campo "hora"dos contextos A e D obtém-se como resultado um intervalo inferior a 4 horas, que corresponde ao tempo mínimo para impressão de uma peça (conforme definido pela Thyssenkrupp Elevadores). Desta forma, o modelo identifica uma possível ordem clonada no sistema, pois o intervalo de tempo não corresponde ao esperado segundo a regra definida e ao esperado no mundo físico.

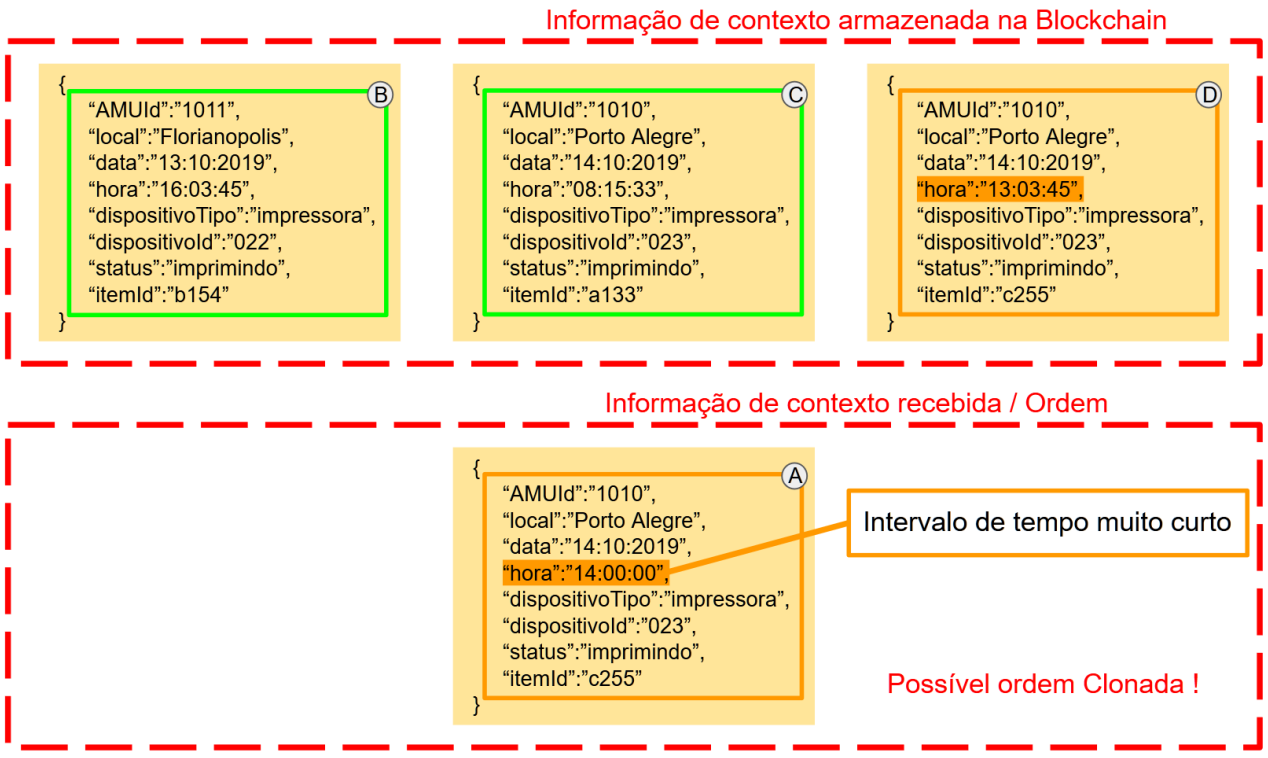

Figura 4.3 – Exemplo de modelo de decisão baseado em regras.

4.3 Modelo de Segurança

A arquitetura proposta (Figura 4.1) tem por objetivo impedir que ordens maliciosas sejam executadas por integrantes da rede. Tendo em vista que abordagens como segurança na camada de transporte (do inglês, *Transport Layer Security* ou TLS) se mostram ineficientes contra ameaças como *Spoofing*, na camada de aplicação [24], podendo ocasionar problemas como ordens e terceiros clonados, assumiu-se que ordens de origem não confiável (A) podem estar comprometidas.

Assume-se que todos os módulos de um integrante da rede estão fisicamente alocados em um mesmo ambiente computacional. Desta forma, todos os módulos estão em um mesmo computador ou dispositivo, não sendo possível interferir na comunicação dos mesmos. Exceto, se o atacante obter acesso físico ao equipamento ou realizar um ataque de *Side channel*, ambos fora do escopo deste trabalho. Adicionalmente, não foram consideradas vulnerabilidades de hardware, excluindo as mesmas do escopo do presente trabalho.

A comunicação entre cada um dos participantes da rede (B) é protegida por meio dos requisitos de segurança presentes na Blockchain, com ênfase em integridade e disponibilidade. Desta forma, a arquitetura mitiga ameaças da CAS relacionadas a integridade e a disponibilidade [42]. Por outro lado, vulnerabilidades especificamente voltadas para a tecnologia de Blockchain, como, por exemplo, ataques de 51%, não são abordadas.

Considerando as premissas de segurança assumidas nos parágrafos anteriores, a arquitetura garante que ordens comprometidas (A) sejam executadas (C) somente se forem consideradas legítimas por todos integrantes da rede. Desta forma, pode-se evitar que uma série de ameaças como *Spoofing*, negação de serviço ou elevação de privilégio, comprometam processos realizados no mundo físico.

4.4 IMPLEMENTAÇÃO

A arquitetura proposta foi implementada utilizando como base os softwares Hyperledger Sawtooth e Fiware Policy Enforcement Point Proxy (PEP) [27][37]. O Sawtooth consiste de uma implementação de Blockchain desenvolvida pela fundação Linux no projeto Hyperledger [8]. Diferente das estruturas providas por abordagens como Ethereum ou Fabric, a implementação Sawtooth não utiliza como base estruturas como de moeda ou token digital. Isto é, a mesma fornece uma implementação "limpa"de Blockchain que permite o desenvolvimento de aplicações específicas (e.g. cadeias de suprimento e armazenamento de dados on-chain).

O PEP, também chamado Wilma, consiste de uma implementação em código aberto de uma ferramenta de segurança do FIWARE [37]. Tal ferramenta tem por função verificar se determinado cliente tem acesso a determinado recurso, e permitir ou não acesso pelo mesmo. O PEP utiliza para verificar se determinado cliente tem acesso a tal recurso, o software FIWARE Keyrock. O Keyrock atua com um conjunto de ferramentas FIWARE, armazenando identidades e respectivas permissões de acesso que são consultadas pelo PEP.

A Figura 4.4 ilustra a implementação da arquitetura proposta instanciada para o caso de uso descrito na Seção 1.1. É possível relacionar os softwares descritos nesta seção com os itens descritos na Figura 4.1. Um cliente corresponde ao indivíduo ou software terceiro não confiável que envia ordens para cada integrante da rede. Como exemplo, podemos considerar um cliente como um técnico de manutenção em campo, de acordo com o caso de uso motivacional do presente trabalho. Cada integrante da rede representa fisicamente um computador de chão de fábrica presente em cada AMU. Os softwares PEP e Hyperledger Sawtooth estão implantados em tal computador, e correspondem respectivamente a implementação do Proxy e dos módulos de Blockchain e CAS. A impressora representa o processo a ser validado, consistindo da impressão 3d de determinada parte.

A implementação dos módulos de Blockchain e CAS com o software Sawtooth pode ser subdividida em Blockchain, API Cliente e DCASTP (abreviatura do inglês, *Decentralized Context Aware Security Transaction Processor*). A Blockchain consiste propriamente de um nodo da Blockchain, que armazena as informações de contexto. Tais informações correspondem ao histórico de ordens processadas pela rede, incluindo ordens

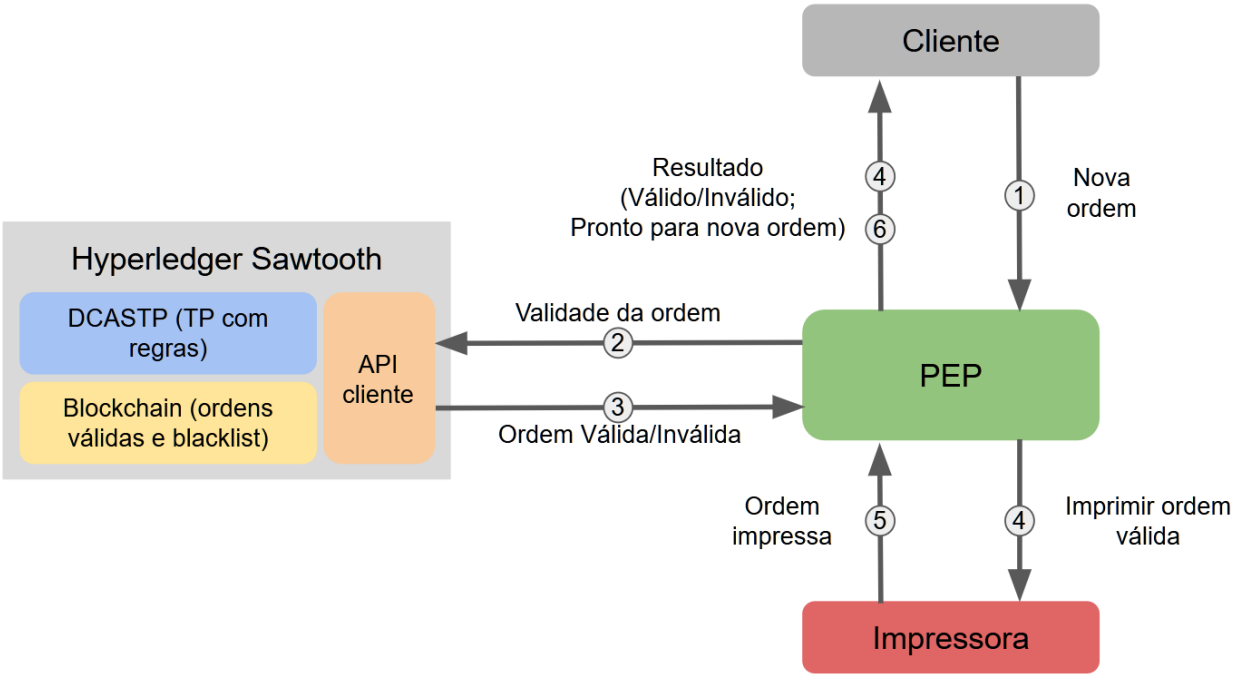

Figura 4.4 – Implementação da arquitetura proposta.

classificadas como válidas e ordens classificadas como inválidas, para a formação de uma *Blacklist*.

A API Cliente consiste de uma interface do Sawtooth para gerenciar a Blockchain, adicionando dados, consultando dados e disparando *Smart Contracts*. Na mesma, é implementada a lógica da *Blacklist* e o início da validação de CAS. A lógica da *Blacklist* consiste de uma consulta na Blockchain que verifica se o cliente que enviou a ordem que está sendo processada apresenta em seu histórico algum comportamento malicioso. Caso afirmativo, a ordem em processamento é classificada como inválida sem executar o processamento de CAS. Desta forma é evitado que a CAS seja alocada para processar uma ordem que possivelmente também está corrompida. Caso o cliente não apresente em seu histórico comportamento malicioso (durante um período configurável de tempo), a API Cliente dispara o DCASTP. O DCASTP consiste de um Processador de Transações (do inglês, *Transaction Processor* ou TP), que equivale a um *Smart Contract* (discutido na subseção 3.1.2) na implementação de Blockchain Sawtooth. Neste TP é implementada a lógica para o processamento de regras da CAS.

A Figura 4.5 ilustra o fluxo da arquitetura proposta, que é dividido nas seguintes etapas:

1. O cliente envia uma nova ordem para o PEP.

2. O PEP requisita para o Sawtooth a validação da ordem por meio do envio para a API Cliente.

3. Após o Sawtooth adicionar a respectiva ordem na Blockchain e executar a decisão da CAS, o mesmo retorna o respectivo resultado para o PEP.
4. Caso a ordem seja considerada Válida (i.e. não foram encontradas inconsistências segundo as regras definidas), o PEP envia a mesma para a impressora e notifica o Cliente. Caso a ordem seja inválida, o PEP apenas envia uma notificação ao cliente, negando a execução da ordem na impressora.
5. Quando a impressora concluir o trabalho de impressão, a mesma notifica o PEP.
6. O PEP notifica o Cliente que a impressora concluiu o serviço de impressão.

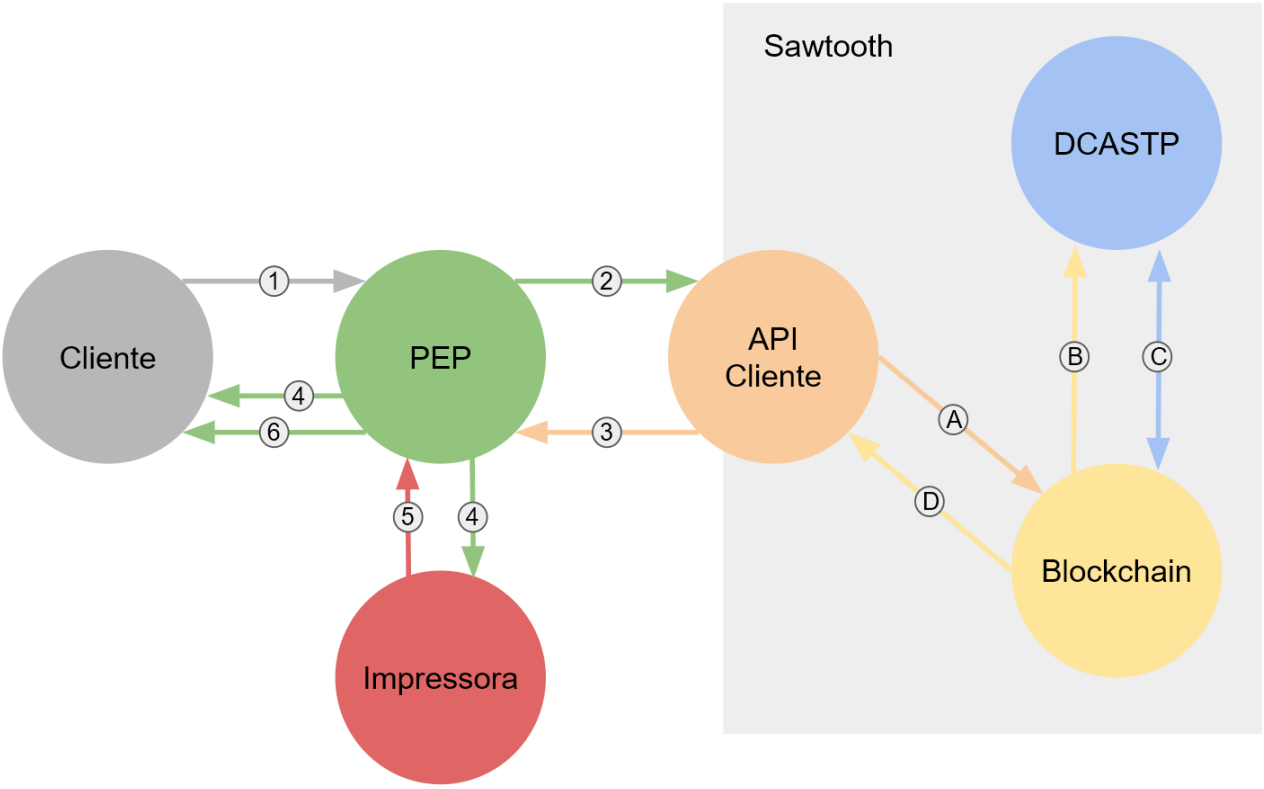

Figura 4.5 – Fluxograma implementação da arquitetura proposta.

Também é possível visualizar na Figura 4.5 o fluxo de execução do software Sawtooth, que pode ser compreendido nas seguintes etapas:

A. A API Cliente recebe uma ordem para validação e executa a lógica da *Blacklist*, isto é, consulta na Blockchain se a AMU de origem da ordem teve alguma ordem comprometida em um período de tempo (este período pode ser configurado). Caso afirmativo, as etapas B, C e D são ignoradas, definindo a ordem como inválida. Este processo evita que AMUs possivelmente comprometidas gerem ordens inválidas na tentativa de consumir recursos da aplicação. Caso a respectiva AMU não tenha ordens prévias inválidas no período de tempo definido, a API Cliente constrói uma transação e envia para o nodo Blockchain.

B. O nodo armazena os dados na Blockchain, iniciando o DCASTP. Nesta etapa é executado o algoritmo de consenso.

C. O DCASTP busca informações de contexto armazenadas na Blockchain, utilizando as mesmas para validar a respectiva ordem e armazenando o resultado novamente na Blockchain.

D. Quando o resultado da validação é armazenado na Blockchain, o mesmo é retornado para a API Cliente.

Para propósito de validação, o **Cliente** foi implementado na forma de um simulador que envia ordens para o PEP. Tal cliente foi implementado utilizando a interface de programação (do inglês, *Application Programming Interface* ou API) Request da tecnologia Nodejs, que fornece um ambiente para execução de código Javascript em lado servidor. Tal implementação envia uma ordem de impressão em formato *JavaScript Object Notation* (JSON) por meio de uma requisição de Transferência de Estado Representacional (do inglês, *Representational State Transfer* ou REST) para o PEP.

Para a implementação da arquitetura proposta, alterou-se o **PEP** para utilizar o software desenvolvido com a ferramenta Sawtooth, possibilitando validar as permissões de determinado cliente. Isto é, ao invés de consultar a ferramenta Keyrock, o PEP passa a consultar a ferramenta desenvolvida correspondente aos módulos de Blockchain e CAS.

```
1   const cbor = require('cbor');
2
3   const payloadBytes = cbor.encode({
4       amuId:"AMU_1101",
5       clientId:"Client_A",
6       partId:"pulley_b154",
7       local:"Florianopolis",
8       dispositivoTipo:"impressora",
9       dispositivoId:"022",
10      time: new Date(),
11      reqId:"735e4d097b"
12  });
```

Figura 4.6 – Agrupando dados no *Payload* e codificando com CBOR.

Para disparar um TP é necessário adicionar dados a um determinado endereço na Blockchain. Para isto, a **API cliente** necessita construir uma transação e adicionar a mesma na Blockchain. Uma transação é constituída de um *Payload* e um *Header*. Como pode ser visualizado na Figura 4.6, o *Payload* consiste dos dados a serem armazenados na Blockchain, ou seja, informações de contexto que correspondem a uma ordem. Por padrão definido pelo Sawtooth, os mesmos são codificados com Representação Binária Concisa de Objeto (do inglês, *Concise Binary Object Representation* ou CBOR), que consiste em um formato de dados capaz de diminuir significativamente o tamanho de mensagens.

```
const transactionHeaderBytes = protobuf.TransactionHeader.encode({
    familyName: 'dcastp',
    signerPublicKey: signer.getPublicKey().asHex(),
    ...
    payloadSha512: createHash('sha512').update(payloadBytes).digest('hex')
}).finish()
```

Figura 4.7 – Implementação do *Header*.

A Figura 4.7 ilustra a implementação da construção do *Header* da transação. Cada *Header* deve conter uma referência para a família de transações utilizada, a chave pública pertencente a respectiva API cliente e um Hash gerado a partir do *Payload*. A família de transações corresponde diretamente ao nome atribuído ao TP, este campo indica qual TP deve processar a transação. A chave pública, gerada a partir do algoritmo criptográfico de curvas elípticas secp256k1, atua como um certificado para provar que a transação foi gerada pela respectiva instância da API Cliente. E por fim, o Hash gerado a partir do *Payload* garante a integridade entre o *Header* e o *Payload*.

```
const signature = signer.sign(transactionHeaderBytes);
const transaction = protobuf.Transaction.create({
    header: transactionHeaderBytes,
    headerSignature: signature,
    payload: payloadBytes
})
```

Figura 4.8 – Implementação da transação.

A Figura 4.8 ilustra a implementação de uma transação. Cada transação é composta por um *Payload* e um *Header* mais a assinatura da respectiva instância da API cliente. Para enviar às respectivas transações para a Blockchain, as mesmas devem ser agrupadas em uma estrutura chamada Batch. Um Batch é composto por uma ou mais transações, um Header que contém as respectivas assinaturas das transações e uma assinatura da própria API cliente que gera o Batch. Vale destacar que as transações em um Batch equivalem às transações adicionadas na Blockchain discutidas na Seção 3.1, porém os Batches não correspondem a Blocos. Tais Batches representam um tipo de formulário definido pela implementação Sawtooth que, ao serem enviados para o endereço de um nodo Blockchain na rede, são adicionados a mesma.

Na implementação Sawtooth, cada nodo da Blockchain é também chamado de Validador. Para implantação dos validadores na arquitetura, que em conjunto correspondem propriamente a **Blockchain**, utilizou-se conteinerização em Docker. O uso do software Docker realiza a virtualização em nível de sistema operacional, possibilitando a rápida implantação de pacotes de software. Desta forma, foi possível implantar diversos validadores

utilizando configurações padrão. Para definir tais configurações é utilizado um arquivo chamado Docker-compose (o Docker-compose utilizado no presente trabalho pode ser visualizado no B).

Para instanciar um validador, diversas configurações devem ser definidas. As configurações utilizadas no presente trabalho foram definidas considerando o presente caso de uso e recomendações de trabalhos anteriores [8]. O algoritmo de consenso foi definido como PoET, discutido no A, utilizando tempo máximo para o sistema de loteria de 3 segundos, que corresponde ao tempo mínimo que mantém estabilidade do algoritmo. Foi definido um número máximo de 100 Batches por Bloco, um número máximo de 2000 transações por Bloco e habilitado o processamento paralelo. Desta forma, considerando o caso de uso, os validadores na rede podem adicionar grandes quantidades de informações por Batch em paralelo, evitando que tais configurações influenciem na performance. Vale destacar que por motivo de testes a descoberta de nodos foi definida como estática, necessitando que os nodos da rede sejam definidos manualmente.

```
const transactionProcessor = new TransactionProcessor("tcp://SAWTOOTHBLOCKCHAIN:4004")
transactionProcessor.addHandler(new TransactionHandler())
transactionProcessor.start();
```

Figura 4.9 – Iniciando TP em nodo da Blockchain.

Como ilustrado no código presente na Figura 4.9, um TP é conectado a Blockchain no momento que o mesmo é iniciado por uma função *start*. Para isto, o mesmo recebe como parâmetro o endereço referente ao validador e um objeto *TransactionHandler*, que é sobrescrito com a lógica referente ao TP. Vale destacar que para a execução de um TP, todos os validadores da rede devem conter uma instância diferente de um mesmo TP.

O objeto que sobrescreve o *TransactionHandler* foi nomeado de **DCASTP** (abreviatura do inglês, *Decentralized Context Aware Security Transaction Processor*). Neste TP encontra-se o processamento das regras de CAS. A Figura 4.10 ilustra a lógica utilizada no DCASTP. A mesma consiste em iterar as informações de contexto obtidas por meio de uma consulta na Blockchain e as regras definidas. Consequentemente, combinando regras e contextos em busca de inconsistências. O resultado de tal busca é posteriormente armazenado na Blockchain e comunicado ao PEP.

As combinações entre regras e contextos são transformadas em formato interpretável para a linguagem Javascript por meio de uma função *parse*. Isto é, tal função transforma expressões como ’60M’ em um formato compreensível para a linguagem de implementação do TP. Neste caso, ele irá compreender como sendo uma hora. Desta forma, pode ser definido em formato de regras, além de comparações entre campos de ordens e contextos, expressões que incluem características como intervalos de tempo e comparação de *timestamps*. A Figura 4.11 ilustra um exemplo de regra que classifica como inválida uma ordem de impressão enviada em um intervalo de tempo inferior a uma hora para uma

```
contexts.forEach((context)=>{
    rules.forEach((rule)=>{
        if(eval(parse(rule.rule,context))){
            setCriticality(rule);
        }
    })
}
```

Figura 4.10 – Código simplificado do DCASTP.

mesma AMU. Após transformar a combinação de regras e contextos em um formato interpretável, uma função *eval* verifica se tal regra foi atendida. O resultado é então armazenado na Blockchain, incluindo, um campo *operator* que armazena a frequência do atendimento da regra, e um campo *criticality* que corresponde a uma classificação da criticidade da regra.

```
{
    rule:"request.amuId == context.amuId && request.time - context.time < 60M",
    operator:"single",
    criticality: 3
}
```

Figura 4.11 – Exemplo de regra.

Para representar a **impressora** foi implementado um servidor REST utilizando a biblioteca Express da tecnologia Nodejs. Tal servidor realiza uma simulação de impressão, que ao receber uma ordem espera um período de tempo configurável que seria referente ao tempo de uma impressão. Para implantação em um cenário real, este servidor deve ser substituído por alguma interface de controle de impressão, como por exemplo o software Octoprint.

4.5 Conclusão

Nesta sessão foi apresentada a arquitetura proposta e como a implementação da mesma foi conduzida. Alguns aspectos da implementação podem ser melhorados em implementações futuras. Por exemplo, é possível cogitar a possibilidade de desenvolvimento de uma implementação de Blockchain específica para o caso Thyssenkrupp, utilizando abordagens como indexação de Blocos ou consenso leve, já discutidos em trabalhos anteriores [32][33]. De mesma forma, o modelo de raciocínio poderia ser fortemente melhorado por meio do uso de abordagens como aprendizado de máquina [30]. Porém, o objetivo do presente trabalho é a validação do conceito apresentado pela arquitetura proposta.

5. VALIDAÇÃO

Este capítulo apresenta os testes realizados para a validação da arquitetura proposta. O principal objetivo de tais testes é verificar a funcionalidade e a performance da arquitetura para o caso de uso descrito na Subseção 1.1. A funcionalidade do sistema deve ser verificada ao instanciar na arquitetura situações que podem ser geradas por meio de ameaças de segurança, consequentemente, verificando a capacidade da mesma de manter a integridade da aplicação. Por outro lado, a performance da mesma deve ser verificada ao implantar a arquitetura considerando as características presentes no caso de uso e analisar métricas como latência e consumo de recursos.

Todos os testes foram realizados em um notebook com sistema operacional Ubuntu 16.04 LTS (64 bits), processador Intel core i7-7700HQ 2.8GHz com 16 GB de memoria RAM e GPU GeForce GTX 1060 6GB. A Figura 5.1 ilustra a implantação da arquitetura em ambiente de testes. Uma rede Docker foi utilizada para instanciar os validadores da rede, isto é, a Blockchain a ser utilizada. Os demais módulos foram instanciados na máquina local, sendo que cada Validador contém anexado a ele os respectivos API Cliente, DCASTP, PEP, Cliente e Impressora.

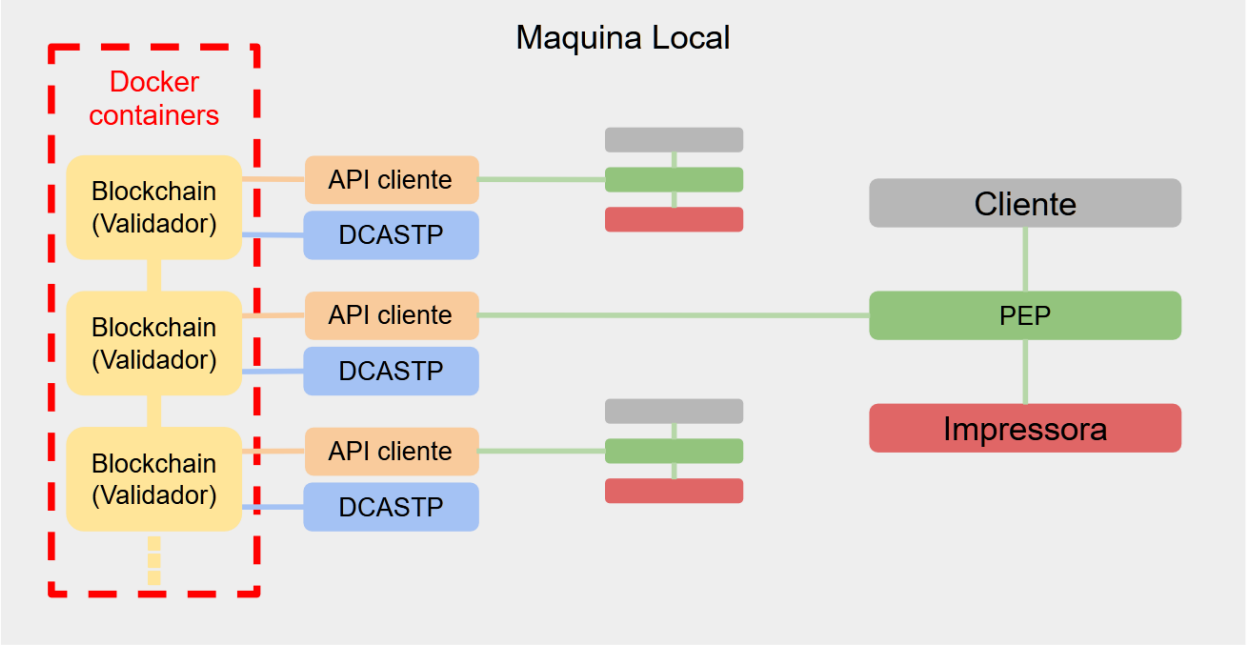

Figura 5.1 – Ambiente de testes.

5.1 TESTES DE FUNCIONALIDADE

Em um cenário real de produção aditiva, cada parte a ser impressa tem atualmente um período mínimo de 4 horas de trabalho de impressão (conforme definido pela empresa parceira Thyssenkrupp elevadores). Considerando isto, pode-se compreender que ordens recebidas por uma mesma AMU em um período de tempo inferior a 4 horas consiste de uma ordem inválida, pois a mesma não concluiu seu atual trabalho de impressão. Tais ordens inválidas poderiam ser geradas por mau uso do sistema, ou a partir de ameaças como *Spoofing*, *Tampering* ou Elevação de privilégio, que possibilitariam a personificação de um atacante como um cliente legítimo capaz de gerar ordens clonadas. Com este cenário, validamos principalmente as funcionalidades da arquitetura referentes a capacidade da mesma de classificar ordens, operar com intervalos de tempo e permitir ou não a execução de ordens classificadas.

Para validar a capacidade da arquitetura de mitigar tal tipo de ameaça, especificamos uma regra definindo um intervalo mínimo de 4 horas para execução de ordens em uma mesma AMU. Tal regra foi implantada em uma rede que simula 5 AMUs. Na sequência, foram enviadas duas ordens a uma mesma AMU com intervalo de tempo entre elas inferior a 4 horas. A primeira ordem foi classificada como válida e teve sua execução realizada. Enquanto a segunda ordem, foi classificada como inválida, tendo sua execução negada pela arquitetura. O processo foi realizado 10 vezes, sendo que todos apresentaram o comportamento esperado, permitindo a execução da 1 ordem e negando a execução da segunda. Vale destacar que também foram realizados 10 testes considerando o cenário inverso. Isto é, após enviar uma primeira ordem válida, a segunda ordem foi enviada com um intervalo de tempo superior a 4 horas, sendo classificada também como válida. Novamente, todos testes apresentaram o comportamento esperado.

Em um cenário real de produção aditiva, a rede de AMUs deve apresentar uma divisão de carga uniformemente relativa a necessidade geográfica de produção de partes. Isto é, podemos compreender que ocorrências como ordens semelhantes sendo enviadas a um conjunto de diferentes AMUs representa uma tentativa de inundar a rede. Desta forma, ordens idênticas não podem ser executadas simultaneamente em diferentes AMUs, dado que representam ordens clonadas no sistema. Tais ordens clonadas podem ser geradas a partir de ameaças como *Spoofing*, *Tampering* ou Elevação de privilégio com a intenção de realizar negação de serviço. Com este cenário, validou-se principalmente as funcionalidades da arquitetura referentes a capacidade da mesma de classificar ordens, permitir ou não a execução de ordens classificadas e considerar informações de contexto compartilhadas entre diferentes AMUs.

Para validar a capacidade da arquitetura de mitigar tal tipo de ameaça, especificou-se uma regra definindo que uma ordem é inválida caso outra AMU tenha executado uma

ordem semelhante em um intervalo de tempo de 15 minutos. Também, foi verificado se uma ordem foi requisitada por um mesmo cliente, para produção de uma mesma parte e em uma mesma quantidade. Tal regra foi implantada em uma rede que simula 5 AMUs. Durante o teste, foi enviado para uma AMU 'A' uma primeira ordem classificada como válida e tendo sua execução realizada. Após isto, em um intervalo de tempo inferior a 15 minutos foi enviada uma segunda ordem idêntica a primeira para uma AMU 'B'. A segunda ordem teve sua execução negada pela arquitetura, classificando a mesma como inválida. Tal processo foi realizado 10 vezes, sendo que todos apresentaram o comportamento esperado, classificando a primeira ordem enviada para uma AMU 'A' como válida e uma segunda ordem idêntica para uma AMU 'B' como inválida. Vale destacar que também foram realizados 10 testes considerando o cenário inverso. Ou seja, foi enviada uma ordem válida para uma AMU 'A' e, em um intervalo inferior a 15 minutos, foi enviada uma segunda ordem diferente para uma AMU 'B', que foi classificada como válida. Novamente, todos testes apresentaram o comportamento esperado.

Para que uma ordem seja adulterada a respectiva identidade ou comunicação de sua origem deve estar comprometida. Desta forma, pode-se assumir que a origem de uma ordem invalida, é de mesma forma, comprometida. Isto posto, ordens provenientes de clientes que já tiveram ordens classificadas como inválidas, durante um período de tempo, não podem ser classificadas como válidas. Com este cenário, validou-se principalmente as funcionalidades da arquitetura referentes a utilização de informações de contexto compartilhadas entre diferentes AMUs e negar a execução de uma ordem de origem comprometida por período configurável de tempo (*Blacklist*).

Para validar a capacidade da arquitetura de evitar que ordens de origem comprometida sejam executadas, a mesma contém um mecanismo que define uma ordem como inválida, caso sua origem tenha requisitado uma ordem invalida durante um período de tempo configurável. Isto é, um cliente que teve uma ordem inválida não poderá requisitar novas ordens durante um período de tempo. Para os testes, foi configurado um período de tempo de 24 horas, em uma rede que simula 5 AMUs. O teste consistiu em enviar uma ordem, propositalmente inválida para uma AMU 'A'. Em um período de tempo inferior a 24 horas, foi enviada uma segunda ordem requisitando uma parte diferente, porém com um mesmo cliente de origem. A segunda ordem foi classificada como inválida, e teve sua execução negada pela arquitetura. Tal processo foi realizado 10 vezes, sendo que todas iterações apresentaram o comportamento esperado, negando uma segunda ordem de mesma origem de uma primeira ordem inválida. Também foram realizadas 10 iterações testando o tempo de expiração de tal funcionalidade. Ou seja, foi enviada uma primeira ordem invalida para uma AMU 'A', e após um período de 24 horas, foi enviada uma segunda ordem requisitando uma parte diferente, porém com um mesmo cliente de origem. Para este caso, a segunda ordem foi classificada como válida, ou seja, apresentando o resultado esperado.

5.2 TESTES DE PERFORMANCE

Pode-se compreender como performance da arquitetura os tempos de execução do processo de validação introduzido pela mesma ao respectivo caso de uso. Isto é, o intervalo de tempo introduzido entre o envio de uma ordem por um cliente e o início da execução de tal ordem em uma impressora (etapas 1, 2, 3 e 4 ilustradas na Figura 4.4). Algumas características de configuração, como número de regras utilizadas ou quantidade de integrantes na rede, podem ser alteradas de acordo com a implantação. Para realização dos testes, considerou-se conjuntos de possíveis configurações para uma rede de AMUs do caso de uso motivacional do presente trabalho.

Cabe ao respectivo gerente de infraestrutura do caso de uso a definição das regras a serem utilizadas na arquitetura proposta. Como, por exemplo, as regras definidas para os testes da subseção anterior. Desta forma, emerge como necessidade a avaliação da performance da arquitetura ao operar com diferentes conjuntos de regras.

Para avaliar a performance da arquitetura ao operar com diferentes conjuntos de regras, criou-se uma regra que invariavelmente classifica as ordens como válidas. Tal regra foi replicada um número de vezes correspondente aos conjuntos de regras utilizados para avaliar a performance da arquitetura. Desta forma, é garantido que as regras configuradas são processadas, possibilitando a simulação do uso de diferentes conjuntos de regras. Para a avaliação, considerou-se conjuntos de 0, 1, 10, 50 e 100 regras implantadas em uma rede que simula 5 AMUs. Para cada um dos conjuntos foram enviadas 1000 ordens, documentando os respectivos resultados de performance da arquitetura.

A Figura 5.2 ilustra o gráfico correspondente a média de performance da arquitetura em milissegundos (ms) gerado para cada conjunto de regras. A configuração de 0 regras corresponde a não realização do processamento das mesmas, uma vez que a arquitetura apenas utiliza o TP para classificar uma ordem diretamente como válida. Como é possível observar ao comparar as médias de tempo de 0 regras e 10 regras, existe uma diferença de performance de aproximadamente 400 ms. Tal diferença corresponde diretamente a latência introduzida pelo processamento de regras, enquanto o restante corresponde a execução do TP. Também pode-se observar que a utilização de 10 a 100 regras apresenta uma média de performance muito semelhante. Isto indica que o aumento da quantidade de regras utilizadas, comprovadamente até a marca de 100, não causa impacto significativo na latência de performance da arquitetura.

Considerando que a arquitetura consiste de um sistema descentralizado, uma das principais características que definem a implantação da mesma é o número de integrantes na rede. Isto é, características de desempenho em relação ao número de nodos da arquitetura podem ser limitantes chave para implantação da mesma em diferentes casos de uso.

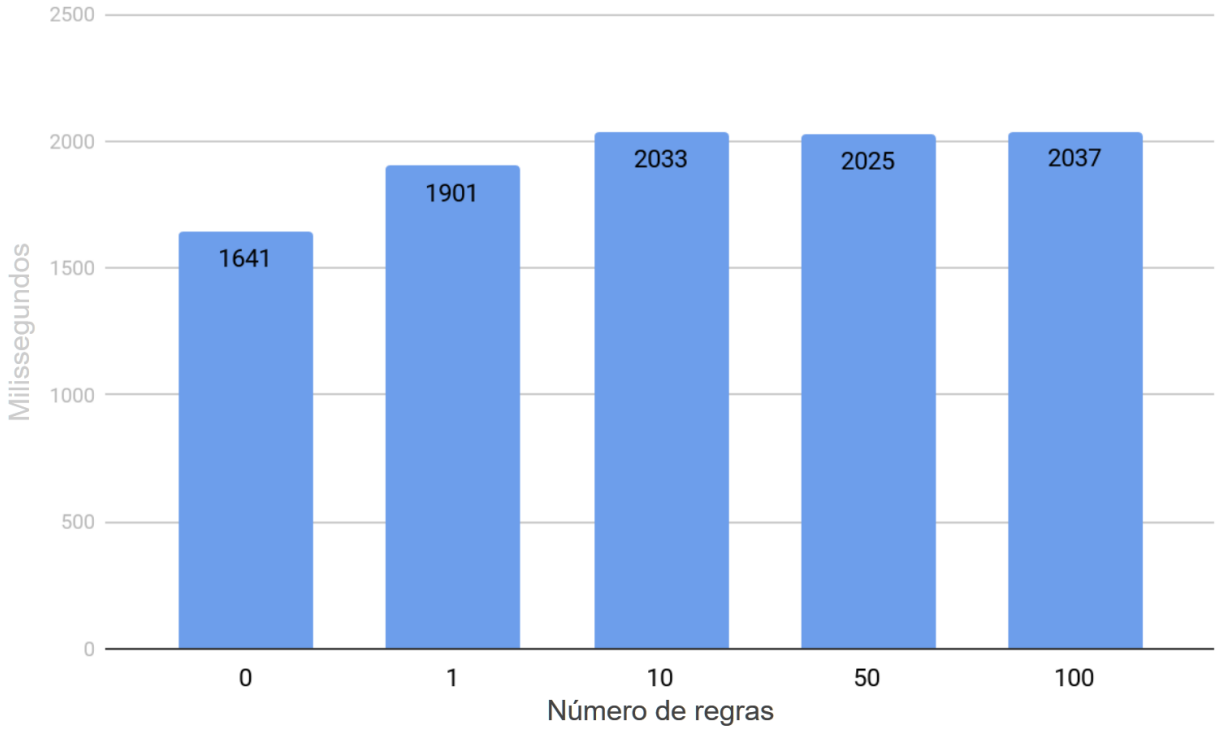

Figura 5.2 – Tempos de execução do processo de validação de acordo com o número de regras utilizadas.

Isto posto, torna-se essencial verificar a performance da mesma de acordo com diferentes configurações de rede.

Para avaliar a performance da arquitetura de acordo com o número de integrantes na rede, foi configurado um conjunto de 5 regras que invariavelmente classificam as ordens como válidas. Tais regras foram implantadas em configurações de rede simulando 1, 3, 5, 10 e 25 AMUs. Sendo que, uma configuração de rede com 25 AMUs corresponde a uma configuração ideal para o caso de uso motivacional do presente trabalho. Desta forma, foi possível avaliar a diferença de latência na performance da arquitetura de acordo com diferentes configurações de rede. Para cada configuração de rede foram enviadas 100 ordens, documentando os respectivos resultados de performance da arquitetura.

A Figura 5.3 ilustra o gráfico correspondente a média de performance da arquitetura em milissegundos (ms) gerado para cada configuração de rede de AMUs. Pode-se observar que diferente do número de regras a serem utilizadas, o número de integrantes na rede exercem grande impacto na performance da arquitetura. Não obstante, ao comparar os tempos de processo de validação das diferentes configurações de rede, heuristicamente visível ao observar as amostras das redes com 5 e 10 nodos, é possível notar a existência de uma escalabilidade positiva de 17,5%. De outra forma, quanto mais integrantes a rede possuir, menor passa a ser a latência introduzida por participante adicional na rede. Essa propriedade ocorre devido a assincronia apresentada durante a execução dos TPs.

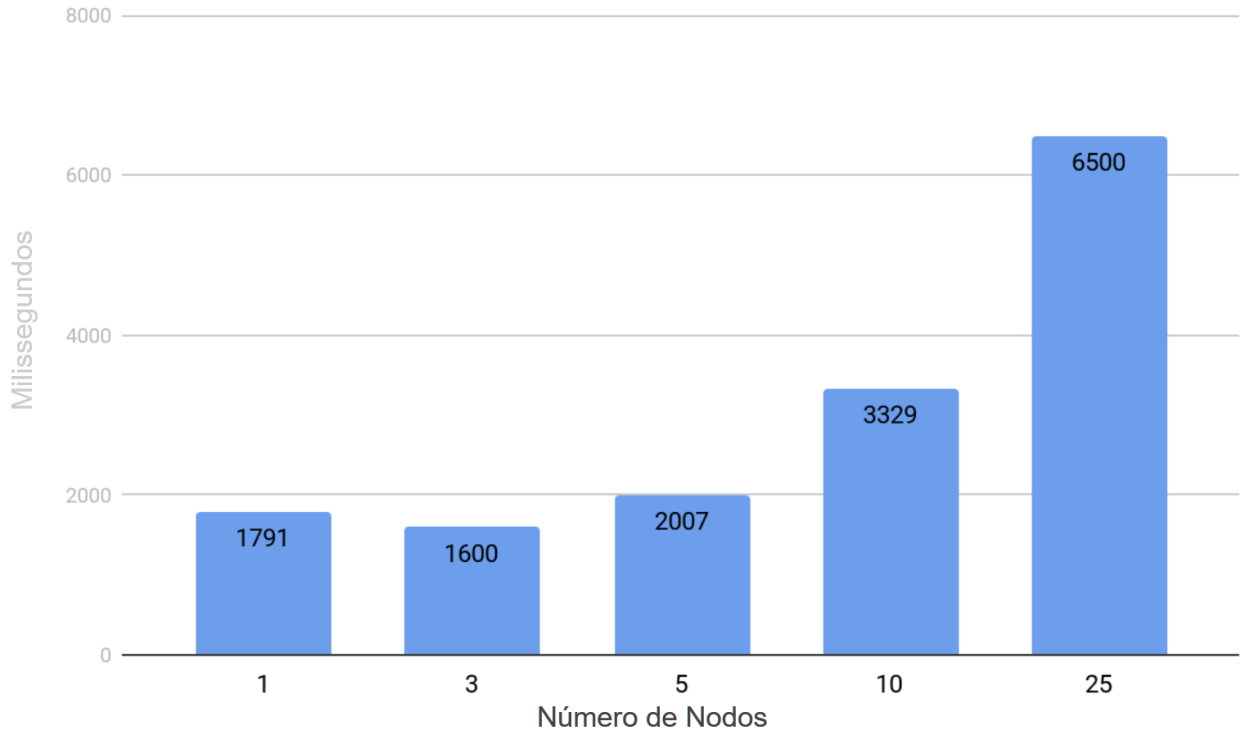

Figura 5.3 – Tempos de execução do processo de validação de acordo com o número de nodos na rede.

Devido a utilização da tecnologia de Blockchain, cada participante da rede necessita armazenar uma cópia de todos os dados trafegados pela mesma. Como resultado, a capacidade de armazenamento dos dispositivos que compõem a rede é um dos principais limitantes para a implantação da arquitetura. Desta forma, torna-se essencial a verificação do espaço de armazenamento necessário para manter a arquitetura durante determinado período de tempo.

Para avaliar o espaço de armazenamento necessário para manter a arquitetura proposta operacional por determinado tempo, utilizou-se um formato de ordem de impressão de tamanho definido, o mesmo utilizado no caso de uso motivacional, semelhante ao apresentado na Figura 3.2, e escalou-se para uma quantidade estimada de tempo. Para calcular a quantidade de dados utilizada em um período de tempo pela rede Sawtooth [27] deve ser somado o tamanho do conteúdo de cada *Payload* das transações com o tamanho das estruturas de Bloco e multiplicado pelo respectivo tempo. Considerando o pior caso, onde é publicada uma transação por Bloco, cada Bloco contendo uma ordem tem tamanho de 2,07Kb. Considerando que cada serviço de impressão leva 4 horas, em 24 horas uma AMU publica no máximo 6 Blocos, que corresponde a 12,42Kb de dados. Desta forma, uma rede com 25 AMUs produziria no máximo 310,5Kb por dia. Considerando estes números, podemos estimamos qual o armazenamento consumido pela arquitetura em períodos de 1, 2, 5, 7 e 10 anos.

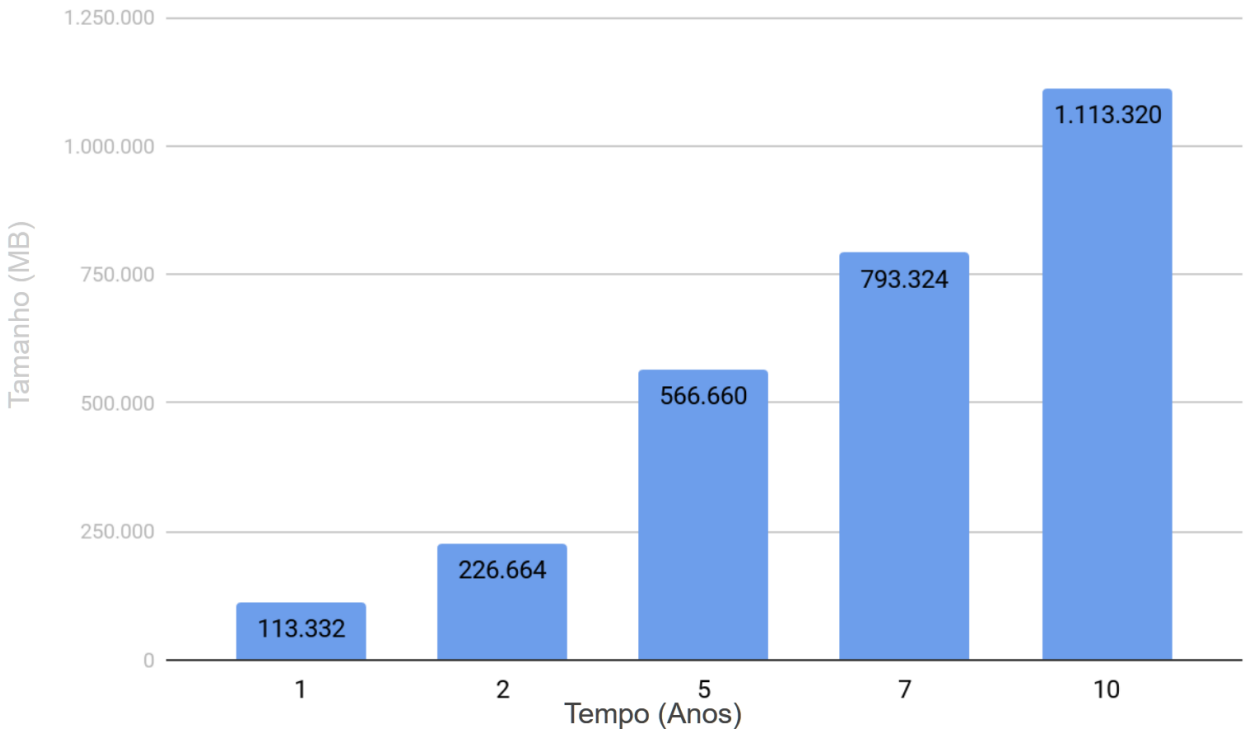

Figura 5.4 – Escalabilidade do tamanho do módulo Blockchain em cada integrante da rede por anos.

A Figura 5.4 ilustra o gráfico correspondente ao armazenamento consumido pela arquitetura (MB) em cada período de tempo (Anos). Podemos observar que diferente da métrica de performance em relação ao número de nodos, a quantidade de armazenamento necessária não apresenta uma escalabilidade positiva. Isto é, a quantidade de armazenamento consumida pela arquitetura é proporcional ao tempo que a mesma é utilizada. Apesar disto, para ultrapassar o consumo de 1Gb, considerando o caso de uso motivacional, a arquitetura demoraria aproximadamente 10 anos. Quantidade de armazenamento que atualmente é alcançada até mesmo por computadores *single-board* atuais, como por exemplo, um Raspberry Pi.

5.3 DISCUSSÃO

Com base nos testes de funcionalidade executados, pode-se concluir que a arquitetura é eficiente em mitigar ameaças relacionadas a segurança de informação. Adicionalmente, destaca-se que os testes de funcionalidade realizados foram elaborados com principal objetivo de validar as funcionalidades da arquitetura. Portanto, ao implantar a arquitetura, mais regras podem ser elaboradas, delimitando o comportamento esperado da aplicação em questão (e.g. explicitando o horário permitido de operação ou verificando se determinada AMU tem recursos para produção de determinada parte).

Com relação aos testes de performance, identificou-se que trabalhos relacionados que exploram abordagens convencionais como criptografia e SDN apresentam melhores tempos de latência se comparados a presente arquitetura [36]. Entretanto, a presente arquitetura propõe um viés diferente de segurança ao adicionar uma camada extra de validação de processos. Desta forma, trabalhos que apresentam propostas semelhantes também apresentam tempos de latência semelhantes a presente proposta, e, em alguns casos, até piores [38].

Pesquisas anteriores já comprovaram que redes Sawtooth, com mais de 5 nodos em Docker, apresentam pior performance se comparada a uma implantação em cenário real [8]. Destaca-se que os valores obtidos durante os testes de performance correspondem a uma média pessimista de latência, indicando que uma implantação em máquinas reais tende a ter melhor performance. Adicionalmente, não foram avaliadas características de consumo de processamento, uma vez que as mesmas já são extensivamente discutidas em trabalhos relacionados [33][8][7].

6. CONSIDERAÇÕES FINAIS

Neste capítulo é apresentada a conclusão deste trabalho. São listadas as publicações obtidas ao longo do curso de mestrado, sendo que todas apresentam relação com o mesmo. Também são apresentadas as conclusões realizadas a respeito da arquitetura proposta. E por fim, são apresentados os possíveis trabalhos futuros com relação ao presente trabalho.

6.1 PUBLICAÇÕES

Para propósito de validação e divulgação da arquitetura proposta, foram realizadas as seguintes publicações ao longo do trabalho:

- **Privacy and security of Internet of Things devices [10], em Real-Time Data Analytics for Large Scale Sensor Data (livro Academic Press), 2019.**

  Este artigo apresenta uma discussão sobre IoT e segurança, e o referencial teórico sobre CAS e Blockchain.

- **Developing Industry 4.0 solutions for Decentralized Production: A new approach for Additive Manufacturing Units, em XXXIX Encontro Nacional de Engenharia de Produção, 2019.**

  Este artigo apresenta o caso de uso discutido ao longo do presente trabalho, documentando a relevância da produção aditiva na indústria 4.0 e a necessidade por arquiteturas de segurança.

- **An Edge Decentralized Security Architecture for Industrial IoT Applications, em IEEE World Forum on Internet of Things 2020, 2020.**

  Este é o principal artigo do presente trabalho, ele apresentada a arquitetura proposta e discute a aplicação da mesma para o caso de uso.

6.2 CONCLUSÃO

A IoT é um paradigma de computação em constante evolução que tem adaptado novas tecnologias e modelos de negócio para melhorar a eficiência de diversos setores. Neste sentido, uma das aplicações da indústria 4.0 tem ganho destaque ao realizar produção aditiva com abordagens de impressão 3D e produção descentralizada [18]. Identificou-

se na literatura a existência de uma necessidade por explorar soluções para prover segurança à indústria 4.0. Especificamente, identificou-se que a indústria tem apresentado uma tendência à descentralização, paralelamente, existindo uma falta de arquiteturas de segurança voltadas para aplicações descentralizadas.

A tecnologia de Blockchain tem como uma das principais características a capacidade de garantir integridade de dados em ambientes descentralizados. Paralelamente, a tecnologia de CAS é capaz de prover asserção de processos para integrantes de uma aplicação descentralizada. Garantindo desta forma, que cada processo executado na aplicação seja validado de acordo com uma política de segurança. Isto posto, identificou-se que ambas tecnologias têm potencial para melhorar a segurança de aplicações descentralizadas.

Foi proposta uma arquitetura de segurança descentralizada que combina as abordagens de Blockchain e CAS. A arquitetura compartilha informações de contexto de forma integra em uma rede descentralizada, utilizando tais informações para uma asserção de segurança baseada em regras. Desta forma, por meio de regras preestabelecidas, é formada uma camada extra de segurança que impede que processos comprometidos sejam executados no mundo físico.

Ao longo do trabalho foi documentada a implementação da arquitetura proposta. A mesma foi dividida em módulos correspondentes às funções realizadas. Cada módulo foi descrito e relacionado com a respectiva tecnologia utilizada para implementação do mesmo. Foram especificados detalhes técnicos de implementação e configuração de cada um dos módulos.

Para validar as capacidades da arquitetura, foram realizados testes. Os testes foram divididos em duas categorias, funcionais e performáticos. Os testes funcionais validaram algumas características de operação e funcionamento da mesma. Como por exemplo, a capacidade da mesma de compartilhar informações de contexto e a capacidade de tomada de decisão utilizando tais informações. Adicionalmente, comprovando a efetividade da arquitetura em mitigar comprometimentos de segurança derivados de ameaças como *Spoofing* ou *Tampering*. Por outro lado, os testes performáticos validaram características da arquitetura relacionadas a escalabilidade e desempenho da mesma. Por meio destes testes, mapeamos o desempenho da arquitetura considerando diferentes configurações da mesma. Fornecendo uma métrica para avaliação de implantação em casos de uso futuros.

Considerando os resultados obtidos ao analisar a arquitetura, concluí-se que é viável a implantação da mesma. No entanto, a arquitetura apresenta tempos de latência significativamente maiores que abordagens convencionais, como as que utilizam criptografia. Entretanto, a arquitetura proposta provê maior abrangência na proteção contra ameaças de segurança. Desta forma, convém a ponderação da necessidade de maior segurança ou tempo de resposta de acordo com o respectivo caso de uso. Como exemplo, considerou-se o caso de uso motivacional do presente trabalho, o tempo mínimo para impressão de uma

parte é de 4 horas, enquanto o tempo médio introduzido pelo uso da arquitetura em uma rede com 25 AMUs e 5 regras é de 6500 milissegundos. Isto é, a validação provida pelo uso da arquitetura para o respectivo caso de uso, adiciona ao tempo total de execução do processo um atraso de 0,045%, valor aceitável para uma arquitetura de segurança.

6.3 TRABALHOS FUTUROS

Futuramente, a implementação da arquitetura deve ser melhorada em relação a sua implantação. Isto é, deve ser desenvolvida uma API e uma interface mais apropriada para a definição de regras. Bem como, deve ser analisada a possibilidade de armazenar as regras na própria Blockchain, por meio de um TP de votação. Para isto, métricas de latência introduzidas por tal processo devem ser coletadas e documentadas.

A IoT tem apresentado tendência à descentralização e computação na borda. A descentralização na indústria 4.0 é um exemplo significativo deste movimento. Entretanto, diversas áreas também apresentam tal tendência, e assim como na indústria 4.0, tem necessidade por arquiteturas voltadas para segurança. Considerando isto, a arquitetura proposta ao longo do presente trabalho também deve ser explorada para outros domínios, sendo melhorada de acordo com os respectivos casos de uso. Considerando as pesquisas bibliográficas realizadas ao longo deste trabalho, podemos antecipar algumas possíveis melhorias necessárias. Tais melhorias incluem, a exploração de Blockchain e algoritmos de consenso leve, bem como o uso de aprendizado de máquina para o processamento de CAS.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Abie, H.; Balasingham, I. “Risk-based adaptive security for smart iot in ehealth”. In: Proceedings of the 7th International Conference on Body Area Networks, 2012, pp. 269–275.

[2] Alfonso, V. “Hype cycle for the internet of things”, Relatório Técnico, Gartner, 2019, 33p.

[3] Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J. A.; Invernizzi, L.; Kallitsis, M.; et al.. “Understanding the mirai botnet”. In: Proceedings of the 26th USENIX Security Symposium, 2017, pp. 1093–1110.

[4] Atzori, L.; Iera, A.; Morabito, G. “The internet of things: A survey”, *Computer Networks*, vol. 54–15, Out 2010, pp. 2787–2805.

[5] Brezillon, P.; Mostefaoui, G. K. “Context-based security policies: a new modeling approach”. In: Proceedings of the Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp. 154–158.

[6] Cheung, K.; Huth, M. R.; Kirk, L. M.; Lundbæk, L. N.; Marques, R. “The frost language”, Relatório Técnico, Xain, 2018, 21p.

[7] Christidis, K.; Devetsikiotis, M. “Blockchains and smart contracts for the internet of things”, *IEEE Access*, vol. 4, Mai 2016, pp. 2292–2303.

[8] Corso, A. “Performance analysis of proof-of-elapsed-time (poet) consensus in the sawtooth blockchain framework”, Dissertação de Mestrado, University of Oregon, UO, 2019, 61p.

[9] Covington, M. J.; Fogla, P.; Zhiyuan, Z.; Ahamad, M. “A context-aware security architecture for emerging applications”. In: Proceedings of the 18th Annual Computer Security Applications Conference, 2002, pp. 249–258.

[10] Das, H.; Dey, N.; Balas, V. E. “Real-Time Data Analytics for Large Scale Sensor Data”. Academic Press, 2019, 298p.

[11] Farooq, M.; Waseem, M.; Khairi, A.; Sadia Mazhar, P. “A critical analysis on the security concerns of internet of things (iot)”, *International Journal of Computer Applications*, vol. 111–2, Fev 2015, pp. 1–6.

[12] Farwell, J. P.; Rohozinski, R. “Stuxnet and the future of cyber war”, *Survival*, vol. 53–1, Jan 2011, pp. 23–40.

[13] Felten, E. W.; Balfanz, D.; Dean, D.; Wallach, D. S. “Web spoofing: An internet con game”, *Software World*, vol. 28–2, Fev 1997, pp. 6–8.

[14] Firesmith, D. “Specifying reusable security requirements”, *Journal of Object Technology*, vol. 3–1, Jan 2004, pp. 61–75.

[15] Flauzac, O.; González, C.; Hachani, A.; Nolot, F. “Sdn based architecture for iot and improvement of the security”. In: Proceedings of the 29th International Conference on Advanced Information Networking and Applications Workshops, 2015, pp. 688–693.

[16] Fremantle, P.; Aziz, B.; Kirkham, T. “Enhancing iot security and privacy with distributed ledgers - a position paper”. In: Proceedings of the 2nd International Conference on the Internet of Things, Big Data and Security, 2017, pp. 344–349.

[17] Frustaci, M.; Pace, P.; Aloi, G.; Fortino, G. “Evaluating critical security issues of the iot world: Present and future challenges”, *IEEE Internet of Things Journal*, vol. 5–4, Out 2017, pp. 2483–2495.

[18] Kemper, H. G. “Application-pull and technology-push as driving forces for the fourth industrial revolution”, *The International Journal of Business Informatics*, vol. 6–4, Abr 2014, pp. 239–242.

[19] Khan, M. A.; Salah, K. “Iot security: Review, blockchain solutions, and open challenges”, *Future Generation Computer Systems*, vol. 82–167, Mai 2018, pp. 395–411.

[20] Kshetri, N. “Can blockchain strengthen the internet of things?”, *IT Professional*, vol. 19–4, Ago 2017, pp. 65–72.

[21] Matos, E.; Tiburski, R. T.; Amaral, L. A.; Hessel, F. “Providing context-aware security for iot environments through context sharing feature”. In: Proceedings of the International Conference On Trust, Security And Privacy In Computing And Communications, 2018, pp. 1711–1715.

[22] Matos, E.; Tiburski, R. T.; Amaral, L. A.; Hessel, F. “Context information sharing for the internet of things: A survey”, *Computer Networks*, vol. 166–1389, Jan 2020, pp. 128–138.

[23] Matzutt, R.; Hiller, J.; Henze, M.; Ziegeldorf, J. H.; Müllmann, D.; Hohlfeld, O.; Wehrle, K. “A quantitative analysis of the impact of arbitrary blockchain content on bitcoin”. In: Proceedings of the International Conference on Financial Cryptography and Data Security, 2018, pp. 420–438.

[24] Meyer, C.; Schwenk, J. “Sok: Lessons learned from ssl/tls attacks”. In: Proceedings of the International Workshop on Information Security Applications, 2013, pp. 189–209.

[25] Miller, B.; Rowe, D. “A survey scada of and critical infrastructure incidents”. In: Proceedings of the ACM Research in Information Technology, 2012, pp. 51–56.

[26] Nakamoto, S. “Bitcoin: A peer-to-peer electronic cash system”, Relatório Técnico, Manubot, 2019, 7p.

[27] Olson, K.; Bowman, M.; Mitchell, J.; Amundson, S.; Middleton, D.; Montgomery, C. “Sawtooth: An introduction”, Relatório Técnico, The Linux Foundation, 2018, 7p.

[28] Ouechtati, H.; Azzouna, N. B.; Said, L. B. “Towards a self-adaptive access control middleware for the internet of things”. In: Proceedings of the International Conference on Information Networking, 2018, pp. 545–550.

[29] Parliament, E. “General data protection regulation”, *Official Journal of the European Union*, vol. 59–294, Mai 2016, pp. 1–88.

[30] Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. “Context aware computing for the internet of things: A survey”, *IEEE Communications Surveys Tutorials*, vol. 16–1, Mai 2014, pp. 414–454.

[31] Pinheiro, P. P. “Proteção de de Dados Pessoais: Comentários à Lei n. 13.709/2018-LGPD”. Saraiva Educação, 2020, 116p.

[32] Polyzos, G. C.; Fotiou, N. “Blockchain-assisted information distribution for the internet of things”. In: Proceedings of the International Conference on Information Reuse and Integration, 2017, pp. 75–78.

[33] Puthal, D.; Mohanty, S. P.; Nanda, P.; Kougianos, E.; Das, G. “Proof-of-authentication for scalable blockchain in resource-constrained distributed systems”. In: Proceedings of the International Conference on Consumer Electronics, 2019, pp. 1–5.

[34] Ratka, S.; Anisie, A.; Boshel, F.; Goussous, N. “Internet of things”, Relatório Técnico, International Renewable Energy Agency, 2019, 28p.

[35] Sadeghi, A. R.; Wachsmann, C.; Waidner, M. “Security and privacy challenges in industrial internet of things”. In: Proceedings of the 52nd IEEE Design Automation Conference, 2015, pp. 1–6.

[36] Sharma, P. K.; Singh, S.; Jeong, Y.-S.; Park, J. H. “Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks”, *IEEE Communications Magazine*, vol. 55–9, Set 2017, pp. 78–85.

[37] Valadares, D. C. G.; Silva, M. S. L.; Brito, A. E. M.; Salvador, E. M. “Achieving data dissemination with security using fiware and intel software guard extensions (sgx)”. In: Proceedings of the Symposium on Computers and Communications, 2018, pp. 1–7.

[38] Venkatapathy, A. K.; Hompel, M. “A decentralized context broker using byzantine fault tolerant consensus”, *Ledger*, vol. 4, Abr 2019, pp. 8.

[39] Vigdor, N. “Somebody’s watching: Hackers breach ring home security cameras”. Capturado em: https://www.nytimes.com/2019/12/15/us/Hacked-ring-home-security-cameras.html, Dez 2019.

[40] Voas, J.; Kuhn, R.; Laplante, P.; Applebaum, S. “Internet of things (iot) trust concerns (draft)”, Relatório Técnico, National Institute of Standards and Technology, 2018, 50p.

[41] Wang, G.; Wang, B.; Wang, T.; Nika, A.; Zheng, H.; Zhao, B. Y. “Defending against sybil devices in crowdsourced mapping services”. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, 2016, pp. 179–191.

[42] Wrona, K.; Gomez, L. “Context-aware security and secure context-awareness in ubiquitous computing environments”, *Annales Universitatis Mariae Curie-Sklodowska*, vol. 4–1, Fev 2006, pp. 332–348.

[43] Yu, B.; Wright, J.; Nepal, S.; Zhu, L.; Liu, J.; Ranjan, R. “Iotchain: Establishing trust in the internet of things ecosystem using blockchain”, *IEEE Cloud Computing*, vol. 5–4, Ago 2018, pp. 12–23.

APÊNDICE A – PROOF OF ELAPSED TIME

O algoritmo de prova de tempo gasto (do inglês, Proof of Elapsed Time ou PoET) é uma proposta promissora, que teve origem em uma parceria entre a empresa Intel e sua extensão de guarda de software (do inglês, Software Guard Extension ou SGX) [8]. Sendo implementado na distribuição de Blockchain Sawtooth, o mesmo consiste de uma tentativa de adaptar o algoritmo de PoW de modo a eliminar seu gasto excessivo de recursos tanto para Blockchains privadas quanto publicas. Para isto, o mesmo explora uma abordagem de loteria, utilizando tempos de espera como registro para realização do sorteio.

A execução do algoritmo tem como pré requisito que todos os participantes da rede tenham uma identidade única. Para isto, cada nodo da rede tem um par de chaves publica e privada. Todo nodo armazena a respectiva chave privada e as chaves publicas de todos os demais nodos da rede. Com esta configuração, o mesmo pode participar das iterações da PoET [27].

A cada iteração do algoritmo, todos participantes da rede geram um valor aleatório que corresponde a um tempo de espera para o nodo. O primeiro nodo a concluir o respectivo tempo de espera gera um novo bloco para a lista encadeada, contendo as devidas transações, o tempo de espera realizado e sua assinatura com a respectiva chave privada. Desta forma, os demais nodos da rede conseguem verificar a identidade do nodo ganhador do sorteio e verificar que seu tempo de espera foi efetivamente menor que o próprio tempo.

Para garantir a honestidade do nodo na geração do par de chaves e dos tempos de espera a cada iteração, ou seja, garantir que um nodo não utilize identidade de terceiros e não simule tempos curtos de espera, é recomendada a execução do algoritmo em um ambiente confiável de execução (do inglês, Trusted Execution Environment ou TEE) [27]. No caso da implementação utilizada na distribuição de Blockchain Sawtooth, que é explorada no presente trabalho, é utilizado TEE baseado em Hardware. O Hardware utilizado consiste da extensão da guarda de software da Intel (do ingês, Software Guard Estension ou SGX). Vale destacar que a grande maioria dos processadores e Hardwares distribuídos pela empresa intel após 2015, contem suporte a SGX [8]. A utilização de TEE não é obrigatória para a operação do algoritmo, embora não seja recomendado o uso do mesmo sem TEE em Blockchains publicas [8]. Na ausência de TEE a implementação Sawtooth utiliza um teste que detecta e impede publicação de Blocos com muita frequência por um mesmo nodo, já discutido em trabalhos anteriores como o suficiente para garantir integridade em redes privadas [27].

O algoritmo tem como forte vantagem o baixo custo computacional se comparado ao tradicional PoW. Também apresentando uma alternativa concisa para problemas encontrados em algoritmos como pBFT e PoS, relacionados a alto uso da rede e a necessidade de estruturas de incentivo. Por outro lado, o mesmo não adiciona ganhos significativos em

taxas de transações, uma vez que necessita que os dispositivos aguardem em inatividade por tempo relativo ao tamanho da rede [8]. Adicionalmente, sua necessidade de execução em TEE para Blockchains publicas é questionada em alguns trabalhos anteriores, devido a necessidade de confiança em um terceiro envolvido (SGX) ferir alguns princípios de descentralização [27].

APÊNDICE B – CONFIGURAÇÃO DO VALIDADOR SAWTOOTH

```
validator-0:
  image: hyperledger/sawtooth-validator:chime
  container_name: sawtooth-validator-default-0
  ports:
    - "4004:4004" # Node port address
  expose:
    - 4004
    - 5050
    - 8800
  volumes:
    - poet-shared:/poet-shared
  command: "bash -c \"\
      sawadm keygen --force && \
      mkdir -p /poet-shared/validator-0 || true && \
      cp -a /etc/sawtooth/keys /poet-shared/validator-0/ && \
      while [ ! -f /poet-shared/poet-enclave-measurement ]; do sleep 1; done && \
      while [ ! -f /poet-shared/poet-enclave-basename ]; do sleep 1; done && \
      while [ ! -f /poet-shared/poet.batch ]; do sleep 1; done && \
      cp /poet-shared/poet.batch / && \
      sawset genesis \
        -k /etc/sawtooth/keys/validator.priv \
        -o config-genesis.batch && \
      sawset proposal create \
        -k /etc/sawtooth/keys/validator.priv \
        sawtooth.consensus.algorithm.name=PoET \
        sawtooth.consensus.algorithm.version=0.1 \
        sawtooth.poet.report_public_key_pem=\
        \\\"$$(cat /poet-shared/simulator_rk_pub.pem)\\\" \
        sawtooth.poet.valid_enclave_measurements=$$(cat /poet-shared/poet-enclave-measurement) \
        sawtooth.poet.valid_enclave_basenames=$$(cat /poet-shared/poet-enclave-basename) \
        -o config.batch && \
      sawset proposal create \
        -k /etc/sawtooth/keys/validator.priv \
          sawtooth.poet.target_wait_time=3 \
          sawtooth.poet.initial_wait_time=5 \
          sawtooth.publisher.max_batches_per_block=100 \
          sawtooth.validator.max_transactions_per_block=2000 \
        -o poet-settings.batch && \
      sawadm genesis \
        config-genesis.batch config.batch poet.batch poet-settings.batch && \
      sawtooth-validator -v \
        --bind network:tcp://eth0:8800 \
        --bind component:tcp://eth0:4004 \
        --bind consensus:tcp://eth0:5050 \
        --peering static \
        --endpoint tcp://validator-0:8800 \
        --scheduler parallel \
        --network-auth trust
  \""
  environment:
    PYTHONPATH: "/project/sawtooth-core/consensus/poet/common:\
      /project/sawtooth-core/consensus/poet/simulator:\
      /project/sawtooth-core/consensus/poet/core"
  stop_signal: SIGKILL
```

Figura B.1 – Arquivo Docker Compose utilizado para configuração do Validador Sawtooth.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br