UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Controle de acesso baseado em *tokens* para nuvem de coisas: Uma análise entre segurança e desempenho aplicado a dados sensíveis de saúde

Wellington da Silva Martins

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Wellington da Silva Martins

Controle de acesso baseado em *tokens* para nuvem de coisas: Uma análise entre segurança e desempenho aplicado a dados sensíveis de saúde

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Julio Cezar Estrella

USP – São Carlos
Dezembro de 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S586c Silva Martins, Welington
Controle de acesso baseado em tokens para nuvem
de coisas: Uma análise entre segurança e desempenho
aplicado a dados sensíveis de saúde / Welington
Silva Martins; orientador Julio Cezar Estrella. --
São Carlos, 2020.
106 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2020.

1. Internet das Coisas. 2. Controle de Acesso.
3. Tokens JSON. 4. Criptografia. 5. Avaliação de
Desempenho. I. Cezar Estrella, Julio, orient. II.
Título.

Wellington da Silva Martins

Token-based access control for the cloud of things. An
analysis between safety and performance applied to
sensitive health data

Master dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-
USP, in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Julio Cezar Estrella

USP – São Carlos
December 2020

*Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.*

AGRADECIMENTOS

Agradeço primeiramente à Deus, pois até aqui me ajudou o Senhor. Agradeço a minha família, Marli Aparecida da Silva Martins (mãe), Adrieli da Silva Martins (irmã) e José Gabriel Martins (pai in memoriam) que sempre me apoiaram e me incentivaram a estudar e batalhar pelos meus objetivos. Em todos os momentos de desespero e anseio em desistir, sempre me encheram de fé e ânimo para prosseguir. Agradeço a minha melhor amiga e amada Giovana Eduarda dos Santos que entrou na minha vida na metade deste percurso mas que se tornou essencial para o término deste trabalho. Agradeço aos meus grandes amigos Ezequias dos Reis Silva e Vinícius Aires Barros que muito contribuíram no início deste trabalho, pois sem vocês ele não existiria. Agradeço aos meus grandes companheiros e irmãos Gabriel Tomiatti Andreazi, Davi Conte, Guilherme Martins, Gesiel Rios, Antônio Marcos Almeida Ferreira e Leonardo Araruna que me acompanharam nesta jornada desde do início e trilharam o mesmo caminho e se tornaram pessoas das quais levarei para sempre no coração. Além disso, outras pessoas que não poderia deixar de mencionar são Leonildo Azevedo, Matheus Saldanha e todos os demais companheiros do LASDPC que contribuíram para a finalização desta dissertação.

Gostaria de deixar um muito obrigado a todos os professores que fizeram parte da minha vida acadêmica, desde daquela primeira escola simples na zona rural de Cabo Verde-MG (Escola Municipal Castro Alves) onde aprendi a ler e escrever e que hoje nem existe mais, depois o período de colegial no Major Leonel e Pedro Saturnino de Magalhães que conheci pessoas que carregou comigo até hoje, passando pela graduação no IFSULDEMINAS que foi um período maravilhoso e que me proporcionou um intercâmbio no Instituto Politécnico de Leiria - PT e por fim até a chegada à Universidade de São Paulo. Todos os professores que contribuíram para este progresso, muito obrigado.

Reservo um agradecimento especial aos professores Paulo Sérgio Lopes de Souza, Sarita Mazzini Bruschi e por ultimo mas não menos importante ao meu orientador e mentor Julio Cezar Estrella que me deu a oportunidade e o desafio de ingressar nesta aclamada instituição e desenvolver esta pesquisa. Foi um período de incontáveis desafios, mas o Prof. Julio sempre manteve-se paciente e buscando explorar o máximo de potencial que eu tinha, além de sempre confiante no desenvolvimento do trabalho, buscando inclusive parcerias que pudessem enriquecer a pesquisa. Graças a oportunidade que me foi ofertada minha vida mudou totalmente, tanto no aspecto acadêmico quanto pessoal e profissional. Por todo apoio, auxílio, paciência e participar ativamente no laboratório estando sempre em contato com os alunos inclusive em momentos de descontração, eu deixo meus mais sinceros agradecimentos.

*“Consagre ao Senhor tudo o que você faz,
e os seus planos serão bem-sucedidos.”
(Provérbios 16:3)*

RESUMO

MARTINS, W. S. **Controle de acesso baseado em *tokens* para nuvem de coisas: Uma análise entre segurança e desempenho aplicado a dados sensíveis de saúde**. 2020. 106 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

A Internet das Coisas é considerada um dos pilares da quarta revolução tecnológica que tem transformado a forma de viver das pessoas, possibilitando melhorias na qualidade de vida, aumento da produtividade e economia de custos. Como toda nova tecnologia, vários desafios surgiram e precisam ser superados para que a Internet das Coisas obtenha um nível estável de utilização. Entre os desafios enfrentados, um dos mais importantes está relacionado à segurança e a privacidade dos usuários, como por exemplo, o controle de acesso. Um dos modelos mais utilizados para o controle de acesso neste universo, é o modelo baseado em capacidades que utiliza *tokens* para autenticação e autorização de usuários e dispositivos. Estes *tokens* são normalmente protegidos por criptografia para prover integridade e confidencialidade. Porém, a escolha da criptografia e das informações contidas nos *tokens* podem afetar drasticamente o desempenho de aplicações que exigem alta responsividade. Assim, o objetivo deste trabalho foi avaliar o impacto que estas escolhas geram no tempo de resposta final para o usuário. Definimos e construímos um protótipo de arquitetura que possibilitou avaliar dois dos principais algoritmos para assinatura de *tokens* (HMAC e RSA). Nosso intuito foi quantificar o melhor *trade-off* em termos de segurança e desempenho para dados sensíveis de saúde gerados a partir de dispositivos inteligentes. A finalidade foi auxiliar os desenvolvedores de aplicações na melhor configuração de *tokens* para o controle de acesso em suas aplicações e verificar quais variáveis mais impactam no tempo de resposta final. Para alcançar este objetivo, adotamos o processo de Avaliação de Desempenho de Sistemas Computacionais junto de um conjunto de planejamento de experimentos que auxiliaram na avaliação do protótipo construído. Além disso, coletamos métricas de consumo de memória e CPU visando avaliar o consumo em sistemas com baixo poder computacional.

Palavras-chave: Internet das Coisas, Controle de Acesso, Tokens JSON, Criptografia, Avaliação de Desempenho.

ABSTRACT

MARTINS, W. S. **Token-based access control for the cloud of things. An analysis between safety and performance applied to sensitive health data.** 2020. 106 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

The Internet of Things is considered one of the pillars of the fourth technological revolution that has transformed the way people live, enabling improvements in the quality of life, increased productivity and cost savings. Like all new technologies, a number of challenges have arisen and must be overcome in order for the Internet of Things to achieve a stable level of use. Among the challenges faced, one of the most important is related to the security and privacy of users, such as access control. One of the most used models for access control in this universe is the model based on capabilities that use tokens to authentication and authorization of users and devices. These tokens are normally protected by encryption to provide integrity and confidentiality. However, the choice of encryption and the information contained in the tokens can drastically affect the performance of applications that require high responsiveness. Thus, the objective of this work was to evaluate the impact that these choices generate in the final response time for the user. We defined and built an architectural prototype that made it possible to evaluate two of the main algorithms for signing tokens (HMAC and RSA). Our aim was to quantify the best trade-off in terms of safety and performance for sensitive health data generated from smart devices. The purpose is to assist application developers in the best token configuration for access control in their applications and verify which variables most impact the final response time. To achieve this goal, we adopted the Computer Systems Performance Assessment process together with a set of planning experiments that helped in the evaluation of the prototype built. In addition, we collect memory and CPU consumption metrics to assess consumption in systems with low computational power.

Keywords: Internet of Things, Access Control, JSON Web Tokens, Cryptography, Performance evaluation.

LISTA DE ILUSTRAÇÕES

Figura 1 – Estatísticas de Crescimento da <i>IoT</i> ((BARROS, 2019) Adaptado de (WITCHALLS; CHAMBERS, 2013)).	8
Figura 2 – Estatísticas sobre o Impacto do Crescimento da <i>IoT</i> ((BARROS, 2019) Adaptado de (WITCHALLS; CHAMBERS, 2013)).	9
Figura 3 – Impacto do Crescimento da <i>IoT</i> na economia (Adaptado de (MANYIKA <i>et al.</i> , 2013)).	10
Figura 4 – Arquitetura 3 e 5 Camadas da <i>IoT</i> (BARROS, 2019) e (NUNES, 2018) . . .	11
Figura 5 – Fluxo abstrato do Protocolo OAuth 2.0	25
Figura 6 – Escolhendo o fluxo de concessão, adaptado de (RICHER; SANOS; GLAZER, 2017)	26
Figura 7 – Modelo de um <i>Token JSON</i>	28
Figura 8 – Técnicas de Avaliação de Desempenho (Adaptado de (SANTANA <i>et al.</i> , 1994))	34
Figura 9 – String de Busca formatada na <i>Scopus</i>	39
Figura 10 – Método de Seleção de trabalhos (Adaptado de (WHITE; NALLUR; CLARKE, 2017))	40
Figura 11 – Identificação da Lacuna de Pesquisa	48
Figura 12 – IOT-SHASEC - Modelo Abstrato	52
Figura 13 – Instância IoT-SHASEC	53
Figura 14 – Sequência de execução da IoT-SHASEC	54
Figura 15 – IOT-SHASEC - Interações dos componentes	56
Figura 16 – Resource Owner	56
Figura 17 – Plano de Teste para carga de trabalho utilizando o JMeter	57
Figura 18 – Construção da requisição no Apache JMeter	59
Figura 19 – Construção da requisição no Apache JMeter	60
Figura 20 – Configuração do arquivo de <i>log format</i>	60
Figura 21 – Configuração do arquivo de <i>log Proxy/Load Balance</i>	61
Figura 22 – Descrição do <i>Token</i> para IOT-SHASEC	63
Figura 23 – Bases utilizadas	64
Figura 24 – Composição do tempo de resposta	66
Figura 25 – Configuração do Agente Perfmon	71
Figura 26 – Servidores monitorados	72
Figura 27 – Cenário de teste	73
Figura 28 – Plano de teste para carga de trabalho utilizando o JMeter	74

Figura 29 – Custo com Proxy	75
Figura 30 – Custo com Load Balance	76
Figura 31 – Análise usando criptografia simétrica	77
Figura 32 – Análise usando criptografia assimétrica	78
Figura 33 – Influência de fatores principais	78
Figura 34 – Influência de fatores de segunda ordem	79
Figura 35 – Tempo de Resposta com RSA - 10 mil Proxy	80
Figura 36 – Tempo de Resposta com RSA - 10 mil Load balance	80
Figura 37 – Monitoramento de CPU e RAM	81
Figura 38 – Análise da Latência ao longo do Tempo com Proxy	82
Figura 39 – Análise da Latência ao longo do Tempo com o Load Balance	82
Figura 40 – Análise da distribuição dos dados	83
Figura 41 – Leitura x Escrita com Proxy	85
Figura 42 – Leitura x Escrita com Load balance	85
Figura 43 – Análise usando Proxy e Load Balance para Escrita	86
Figura 44 – Monitoramento de CPU e RAM	87

LISTA DE TABELAS

Tabela 1 – Algoritmos <i>JWS</i> para Assinatura (JONES, 2015a)	30
Tabela 2 – Algoritmos <i>JWE</i> para Encriptação (JONES, 2015a)	30
Tabela 3 – Bases de Pesquisa	39
Tabela 4 – Dados do formulário de extração	41
Tabela 5 – Principais trabalhos selecionados	41
Tabela 6 – Tabela de composição do código chave/valor	65
Tabela 7 – Componentes do tempo de resposta	67
Tabela 8 – Decomposição do tempo de resposta	67
Tabela 9 – Configuração <i>clusters</i> do <i>LaSDPC</i>	72
Tabela 10 – Parametrização dos fatores e níveis do experimento	73
Tabela 11 – Configuração experimental	73
Tabela 12 – Valores fixos associados aos parâmetros do experimento	74
Tabela 13 – Tempo de Resposta (RSA) - Proxy x Load Balance	80
Tabela 14 – Valores fixos associados aos parâmetros do experimento	84
Tabela 15 – Valores variáveis associados aos parâmetros do experimento	84
Tabela 16 – Combinação experimental	84

LISTA DE ABREVIATURAS E SIGLAS

ABAC	Attribute-based Access Control
ABE	Attribute-Based Encryption
ACE	Authentication and Authorization for Restrash Entries
AWS	Amazon Web Services
CAPBAC	Capability Based Access Control
CoAP	Constrained Application Protocol
DaaS	Data as a Service
DBaaS	Database as a Service
DCAF	Delegated CoAP Authentication and Authorization Framework
EaaS	Ethernet as a Service
EC	Critério de Exclusão
ECC	Elliptic Curve Cryptography
EPC	Electronic Product Code
FPGA	Field Programmable Gate Array
GAE	Google App Engine
GPS	Global Positioning System
HMAC	Hash-based Message Authentication Code
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HVAC	heating, ventilating and air conditioning
IaaS	Infrastructure as a Service
IC	Critério de Inclusão
ICAP	Secure Identity-based Capability System
IdP	Identity Provider
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPMaaS	Identity and Policy Management as a Service
JOSE	JavaScript Object Signing and Encryption
JSON	JavaScript Object Notation
JWE	JSON Web Encryption

JWS	JSON Web Signature
JWT	JSON Web Token
MAC	Message Authentication Code
Multi-OrBAC	MultiOrganization-Based Access Control
NFC	Near Field Communication
OASIS	Organization for the Advancement of Structured Information Standards
OrBAC	organization-based access control
OSI	Open System Interconnection
P2P	Peer-to-Peer
PaaS	Platform as a Service
PAP	Policy Administration Point
PBAC	Policy Based Access Control
PDP	Policy Decision Point
PEP	Policy Enforcement Point
Poly-OrBAC	access control framework for Inter-Organizational Web Services
PoP	Proof of Possession
PUB	Função Unclonable Física
RBAC	Role-based Access Control
RFID	Radio-Frequency Identification
S2aaS	Sensing as a Service
SAaaS	Sensing and Actuation as a Service
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SDN	Software Defined Networking
SEaaS	Sensor Event as a Service
SenaaS	Sensor as a Service
SLA	Service Level Agreement
SN	Sensor Networks
SWT	Simple Web Token
TACIoT	Trust-Aware Access Control System
TBAC	Token-Based Access Control
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPM	Trusted Platform Module
UBAC	User-Based Access Control
UCIoT	Usage Control in IoT
UCON	Access control solutions based on usage control

URL	Uniform Resource Locator
WoT	Web of Things
WSN	wireless sensor network
XACML	Xtensible Access Control Markup Language
XML	eXtensible Markup Language

SUMÁRIO

1	INTRODUÇÃO	1
1.1	Motivação	3
1.1.1	<i>Objetivos Gerais e Específicos</i>	4
1.2	Estrutura	5
2	REFERENCIAL TEÓRICO	7
2.1	Considerações Iniciais	7
2.2	Internet das Coisas	7
2.2.1	<i>Arquitetura IoT</i>	9
2.2.2	<i>Computação em Nuvem</i>	11
2.3	Desafios da Internet das Coisas	13
2.4	Segurança em Internet das Coisas	16
2.5	Controle de Acesso	18
2.6	Protocolos e <i>Frameworks</i>	23
2.6.1	<i>Linguagem de Marcação de Controle de Acesso Extensível (XACML)</i>	23
2.6.2	<i>OAuth 2.0</i>	24
2.6.3	<i>JSON Web Tokens</i>	27
2.7	Avaliação de Desempenho de Sistemas Computacionais	30
2.7.1	<i>Técnicas de Avaliação de Desempenho</i>	33
2.8	Considerações Finais	36
3	TRABALHOS RELACIONADOS	37
3.1	Considerações Iniciais	37
3.2	Mapeamento Sistemático	37
3.2.1	<i>Método de Pesquisa</i>	38
3.2.2	<i>Bases de pesquisa</i>	38
3.2.3	<i>Critérios de Inclusão e Exclusão</i>	39
3.2.4	<i>Processo de Seleção de Leitura</i>	40
3.2.5	<i>Extração de dados</i>	40
3.3	Análise Crítica dos Trabalhos Relacionados	41
3.4	Identificação do Problema	47
3.4.1	<i>Domínio da Aplicação</i>	48
3.5	Considerações Finais	49

4	ARQUITETURA IOT-SHASEC	51
4.1	Considerações Iniciais	51
4.2	A Arquitetura IOT-SHASEC	51
4.2.1	<i>Fluxo de execução IOT-SHASEC</i>	53
4.2.2	<i>Componentes da IOT-SHASEC</i>	55
4.3	Composição do tempo de resposta	65
4.4	Considerações Finais	66
5	AVALIAÇÕES EXPERIMENTAIS	69
5.1	Considerações Iniciais	69
5.2	Avaliação de desempenho	69
5.3	Monitoramento	70
5.4	Ambiente de Configuração	71
5.4.1	<i>Desenho do experimento A</i>	72
5.4.1.1	<i>Conclusões do Experimento A</i>	82
5.4.2	<i>Desenho do Experimento B</i>	83
5.4.2.1	<i>Conclusões do Experimento B</i>	86
5.5	Considerações Finais	87
6	CONCLUSÃO	89
6.1	Publicações e Submissões	89
6.2	Contribuições e Limitações	90
6.3	Trabalhos futuros	91
	REFERÊNCIAS	93

INTRODUÇÃO

Atualmente, vivemos um período de grande revolução tecnológica, em que conceitos como dispositivos inteligentes, carros e casas inteligentes ganharam notoriedade de centros de pesquisa nos últimos anos ([ORACEVIC; DILEK; OZDEMIR, 2017a](#)). Essa revolução envolve muitas áreas do conhecimento e da indústria como: saúde pública, transportes, agricultura e muitas outras, e tem proporcionado um aumento massivo de dados gerados pelos diversos dispositivos existentes ([ATZORI; IERA; MORABITO, 2010](#)). Em razão disso, um termo que tem se destacado pela indústria e academia ao redor do mundo é a Indústria 4.0. Este termo surgiu pela primeira vez na feira de Hannover em 2011 na Alemanha. Em Outubro de 2012 o grupo de trabalho sobre Indústria 4.0, apresentou um conjunto de recomendações de implementação para o governo alemão. Em Abril de 2013, na mesma feira, o trabalho final foi publicado ([OKANO, 2017](#)). O termo indica as quatro grandes transformações tecnológicas que ocorreram ao longo da história até os dias atuais.

A primeira transformação ocorreu com a primeira Revolução Industrial no século XVIII com o aperfeiçoamento da máquina à vapor *Newcomen* por James Watt na Indústria Têxtil gerando um novo modelo de economia e contribuindo pela riqueza da época ([BECKERT, 2015](#)). A segunda grande transformação tecnológica conhecida como Indústria 2.0, foi a substituição de vapor por eletricidade, a transformação da corrente alternada e os meios avançados de isolamento que eram necessários ([CARR, 2008](#)), ([KIEBITZ, 1931](#)), ([SIEMENS, 1889](#)). A terceira grande transformação tecnológica ou Indústria 3.0 surgiu na década de 70 com a montagem automatizada em série com o aparecimento de eletrônicos, máquinas de inserção de componentes, o aparecimento de robôs para realização de trabalhos mecânicos entre outros ([SCHMIDT et al., 2015](#)). A quarta grande transformação ou Indústria 4.0 se caracteriza sobretudo pela mescla das principais tecnologias para automação, coleta, processamento e comunicação de dados. Ela envolve os conceitos de Computação em Nuvem, Big Data, Sistemas Ciber-Físicos e Internet das Coisas ([CHHETRI et al., 2017](#)), ([JARADAT et al., 2017](#)).

A Internet das Coisas do, inglês (*Internet of Things (IoT)*) é um dos principais pilares responsáveis pela atual revolução tecnológica. Segundo [Koreshoff, Robertson e Leong \(2013\)](#) e [Nunes \(2018\)](#), Internet das Coisas é a mais recente evolução da Internet, que incorpora bilhões de dispositivos inteligentes que se comunicam e que aproveitam seus dados e funcionalidades para fornecer novos serviços inteligentes e produtos que beneficiem a sociedade. De acordo com [Aazam et al. \(2016\)](#), o principal objetivo da *IoT* é fornecer uma infraestrutura de rede com protocolos de comunicação interoperáveis e *softwares* que permitam a conexão e incorporação entre diferentes tipos de dispositivos e diferentes redes a qualquer momento.

Grande parte dos dados que são gerados a partir de dispositivos *IoT* são processados pela Computação em Nuvem. Isso se deve geralmente aos baixos níveis de poder computacional dos dispositivos embarcados. Quando tecnologias de Internet das Coisas se integram à tecnologias de Computação em Nuvem, surge o conceito de Nuvens de Coisas ([AAZAM; HUH, 2014a](#)). Uma pesquisa realizada por [Witchalls e Chambers \(2013\)](#) mostrou que em até 2020 deve haver mais de 50 bilhões de dispositivos conectados que representará em média de 6.58 dispositivos por pessoa. A quantidade de dispositivos conectados geram um aumento significativo de dados, sobrecarregando a rede e até mesmo o poder de processamento da nuvem. Para contornar esse desafio, surge o conceito de Computação na Borda do inglês *Fog Computing* e *Edge Computing*. De forma geral, este conceito implica em trazer parte do processamento realizado na nuvem para a borda da rede com o propósito de proporcionar uma maior velocidade no processamento de aplicações críticas. [Hu et al. \(2017\)](#) explicam ainda que parte dos desafios existentes na nuvem também são transferidos para a borda da rede.

Como toda tecnologia emergente, novos desafios surgiram e precisam ser mitigados para que se chegue em um nível estável de utilização. Entre alguns problemas relacionados à Indústria 4.0 e mais precisamente em Internet das Coisas, estão: **Confiabilidade dos Dados, Mobilidade, Disponibilidade, Escalabilidade, Desempenho, Problemas de Interoperabilidade, Gerenciamento dos Dados e Segurança** ([CHOUDHARY; JAIN, 2016](#)) e ([AL-FUQAHA et al., 2015](#)). Embora alguns autores possam enumerar algumas variações, estes desafios tendem a estar em consenso na maioria dos trabalhos analisados. Dentre estes desafios, [Skarmeta, Hernandez-Ramos e Moreno \(2014\)](#) afirmam que os problemas relacionados à segurança tem se destacado em relação aos demais. Quando se analisa especificamente problemas relacionados à segurança, surgem novos desafios. [Pawar e Ghumbre \(2016\)](#) enumera os principais desafios de segurança como: **Confidencialidade, Integridade, Autenticação, Privacidade, Gestão de Confiança, Aplicação de Políticas e Controle de Acesso**.

Todos estes desafios tem sido amplamente discutidos na literatura, principalmente soluções que envolvem o Controle de Acesso em *IoT*. [Ouaddah et al. \(2017\)](#), apresentam um estudo comparativo dos principais modelos de controle de acesso trazendo para um contexto de Internet das Coisas. Estas soluções fazem parte do modelo tradicional de comunicação aplicado em redes de computadores tais como o Protocolo de Controle de Transmissão do, inglês (*Transmission*

Control Protocol (TCP)) e o Protocolo de Interconexão do, inglês (*Internet Protocol (IP)*). No entanto, o intuito é fazer com que estas soluções de controle de acesso se adaptem ao novo paradigma da *IoT*. Dentre as diversas soluções apresentadas por Ouaddah *et al.* (2017), o Modelo Baseado em Capacidades do inglês (*Capability Based Access Control (CAPBAC)*) é um dos mais utilizados para Internet das Coisas, devido às suas características de **Simplicidade, Flexibilidade, Distribuição, Usabilidade, Delegação e Revogação de Suporte** necessárias no universo de "coisas" (OUADDAH *et al.*, 2017).

Uma capacidade, (conhecida em alguns sistemas como chave) é um *token* de autoridade transmissível e sem expiração, que apresenta um conjunto de valores que fazem referência exclusiva a um objeto e que descreve quais são suas permissões de acesso. (GUSMEROLI; PICCIONE; ROTONDI, 2013a). A capacidade contém direitos concedidos à entidade que o possui. Este conceito foi introduzido inicialmente por Dennis e Horn (1966) como *token*, *ticket* ou chave que permite ao possuidor permissão para acessar uma entidade ou objeto em um sistema computacional. Na maioria dos casos, estes *tokens* são construídos por asserções no formato de Linguagem de Marcação Extensiva do inglês *eXtensible Markup Language (XML)* ou em Notação de Objetos *JavaScript* do inglês *JavaScript Object Notation (JSON)*.

Os trabalhos mais recentes que fazem uso dos *tokens* para auxiliar no processo de controle de acesso, tem dado preferência aos modelos referentes ao formato *JSON*. Isso decorre principalmente em razão do seu menor tamanho, o que também demanda menor custo computacional de processamento o que é importante em dispositivos *IoT* restritos. Os *tokens* em asserções *JSON* são denominados como *JSON Web Tokens*, os quais, fazem uso de criptografia para prover integridade e confidencialidade dos dados (JONES; BRADLEY; SAKIMURA, 2015).

1.1 Motivação

Entre os diversos desafios existentes em Internet das Coisas, os que se relacionam com a segurança de forma geral tem se destacado em relação aos demais. O controle de acesso é um dos principais desafios que envolve a segurança de dados, levando ao surgimento de diferentes e diversas pesquisas neste sentido. As soluções que envolvem o uso de *tokens* para prover autenticação e autorização tem se tornado popular se comparada às soluções tradicionais de controle de acesso. Isso é devido a capacidade dos *tokens* melhor se adaptarem ao ambiente *IoT*, principalmente os *tokens* em formato *JSON*. Normalmente, se utilizam algoritmos de criptografia para proteger as informações contidas nos *tokens* e vários algoritmos podem ser aplicados para prover diferentes requisitos e níveis de segurança. Porém, sempre que há a necessidade de mais segurança, o desempenho de uma aplicação tende a ser afetado. Em ambientes de saúde e telemedicina normalmente é necessário desempenho adequado com mais segurança, mas não encontramos trabalhos que façam um estudo considerando o *trade-off* entre estas duas variáveis.

Entre as diversas soluções de controle de acesso no contexto de *IoT*, sobretudo nas

soluções que abordam o uso de *tokens* não foram encontrados trabalhos que avaliassem o impacto causado por variações de criptografia na construção dos *tokens*. Não foram encontradas métricas que auxiliassem à tomada de decisão para a escolha de uma configuração de *token* mais apropriada. Este processo de seleção é importante no universo de Internet das Coisas, dado que um algoritmo pode consumir mais ou menos recursos computacionais do dispositivo, o que pode afetar diretamente o desempenho e a segurança da aplicação. A maior parte dos trabalhos abordados, não relatam claramente quais são os algoritmos utilizados na elaboração de seus *tokens*, nem mesmo seu processo de construção. Os poucos trabalhos que o fazem, normalmente não justificam o porquê da escolha de um algoritmo em detrimento dos demais possíveis e tendem a fazer uma avaliação de desempenho da proposta de forma superficial.

Beltrán, Calvo e González (2017) afirmam que o processo de validação dos *tokens* na aplicação que eles propuseram é uma das tarefas mais demoradas e que consome muita energia, sendo o processo de verificação da assinatura uma das etapas mais complexas. A motivação deste projeto de mestrado surge justamente para lidar com a lacuna identificada. O intuito é prover um estudo com abordagens que auxilie na melhor seleção de um algoritmo criptográfico para *tokens*, possibilitando um *trade-off* entre desempenho e segurança. O intuito é contribuir com a literatura para a seleção de algoritmos de criptografia mais apropriados na utilização de *tokens*, principalmente os *tokens* de formato *JSON*, de modo que o impacto causado no custo computacional seja o menor possível.

1.1.1 Objetivos Gerais e Específicos

Esse trabalho apresenta um estudo que se utiliza do processo de avaliação de desempenho que objetiva identificar variáveis que podem impactar no processamento e no tempo de resposta final para o usuário. Para auxiliar neste estudo e avaliação propomos um protótipo de arquitetura de controle de acesso que é a base para as avaliações realizadas. Nosso foco, considera primordialmente a avaliação de desempenho para comprovar estatisticamente o quanto a escolha de um modelo criptográfico pode afetar no desempenho da aplicação.

Assim este trabalho tem como propósito geral o estudo e a avaliação de desempenho envolvendo controle de acesso em Internet das Coisas que consideram o uso de *tokens* para transferência, autenticação e autorização. O estudo avalia algoritmos de assinatura de *tokens* em ambientes de criticidade como por exemplo em ambientes de saúde pública utilizados para a realização de controle de acesso. Além disso, foi importante a construção de um protótipo arquitetural denominado (IOT-SHASEC) que possibilitasse a realização das avaliações. Nosso objetivo visa a seleção de algoritmos criptográficos para quantificar qual o melhor *trade-off* em termos de segurança e desempenho em um ambiente de dados sensíveis, a fim, de auxiliar na melhor configuração de *token* para o cenário de aplicação. Para alcançar este objetivo, este projeto aborda os seguintes objetivos específicos:

- Definir cenários e avaliações experimentais para estudos que envolvem segurança e desempenho com algoritmos de assinatura de *tokens*;
- Avaliar possíveis variáveis que possam impactar no tempo de resposta por uma perspectiva de desempenho;
- Definir e construir um protótipo de arquitetura para controle de acesso que sirva como base e referência para as avaliações realizadas;

1.2 Estrutura

Esta dissertação está organizada da seguinte forma: O Capítulo 2 apresenta o referencial teórico deste projeto de pesquisa destacando conceitos necessários para compreensão do projeto tais como: Internet das Coisas, Computação em Nuvem, Nuvem de Coisas, Desafios em *IoT*, Soluções de Controle de Acesso, OAuth, *Tokens JSON* e Avaliação de Desempenho. O Capítulo 3 apresenta os trabalhos relacionados utilizados como referência para esta dissertação expondo a metodologia adotada para a definição da *string* de busca e para seleção dos estudos primários, acrescentando-se ainda, uma análise crítica dos principais trabalhos. No Capítulo 4 é apresentada a abordagem proposta, detalhando, e em seguida a arquitetura desenvolvida, as ferramentas e os *frameworks* utilizados e em seguida no Capítulo 5 são apresentados os resultados alcançados com o desenvolvimento deste trabalho. Por fim, no Capítulo 6 são apresentadas as conclusões e as contribuições geradas, bem como as limitações e os trabalhos futuros.

REFERENCIAL TEÓRICO

2.1 Considerações Iniciais

Este Capítulo têm por objetivo apresentar o referencial teórico que embasa esta dissertação de mestrado. Na Seção 1, apresentamos conceitos fundamentais sobre Internet das Coisas. Na Seção 2, apresentamos os conceitos referentes ao processamento dos dados em Computação em Nuvem. Na Seção 3, apresentamos os principais desafios referentes a *IoT*. Nas seções 4 e 5, apresentamos as principais soluções de controle de acesso em Internet das Coisas. Na Seção 6, apresentamos os conceitos referentes aos *tokens JSON* e uma tabela com os algoritmos de criptografia existentes. Por fim, são apresentadas as considerações finais deste capítulo.

2.2 Internet das Coisas

A Internet das Coisas refere-se a uma nova abordagem sobre o uso de objetos conectados por meio da Internet que objetiva transformar a vida cotidiana das pessoas. Segundo [Koreshoff, Robertson e Leong \(2013\)](#), a *IoT* refere-se a uma visão na qual "coisas" como objetos, lugares e ambientes estão conectados por meio da Internet que passará de uma rede global de computadores para uma rede global de objetos. A *IoT* é constituída por equipamentos inteligentes que vão desde sensores multimodais a produtos inteligentes como relógios, bicicletas, semáforos e muitos outros ([SU et al., 2015](#)).

Muitas aplicações têm sido desenvolvidas utilizando esse novo conceito de conexão de objetos em rede, assim, para uma consulta abrangente sobre estes diversos domínios de aplicações o trabalho de [Borgia \(2014\)](#) pode ser consultado. Como exemplo básico destas aplicações destacam-se, sistemas de monitoramento e controle de termostatos e *HVAC* do inglês (*heating, ventilating and air conditioning (HVAC)*) que permitem casas inteligentes, automação industrial, transporte inteligente, saúde eletrônica e diversas aplicações na indústria que objetivam

oferecer aos usuários melhores serviços e maior qualidade de vida (AL-FUQAHA *et al.*, 2015) e (SHI *et al.*, 2018). Contudo, o leque de aplicações é gigantesco e cresce a cada dia, sobretudo, em aplicações de emergência e desastres naturais, onde a tomada de decisão humana é difícil.

Al-Fuqaha *et al.* (2015) explica que a *IoT* possibilita que objetos físicos vejam, ouçam, pensem e executem trabalhos, fazendo com que eles "conversem" em conjunto, compartilhem informações e coordenem decisões. Por outro lado, podemos dizer que a *Internet of Things* engloba uma mescla de diversas tecnologias de *hardware*, tais como: Identificadores de rádio-frequência do, inglês (*Radio-Frequency Identification (RFID)*), Comunicação por Campo de Proximidade, do inglês (*Near Field Communication (NFC)*), Redes de Sensores, do, inglês (*Sensor Networks (SN)*) e de tecnologias de *softwares* tais como *middlewares* e *frameworks* (WHITMORE; AGARWAL; XU, 2015).

Ao longo do tempo, estima-se que a *IoT* tenha aplicações significativas que ajude no crescimento da economia mundial. Evans (2011) e Gartner (2016) destacam que em 2020 a quantidade de dispositivos inteligentes conectados à Internet será de aproximadamente 50 bilhões. O estudo ainda afirma que o impacto da *IoT* será em todas as áreas da indústria e sociedade, estimando receitas previstas em 263 bilhões de dólares.

Neste sentido, o relatório econômico Witchalls e Chambers (2013) aponta que cerca de 30% dos líderes empresariais acreditam que a *IoT* permitirá novas oportunidades de negócios. Além disso, o estudo mostra que 29% dos líderes acreditam que a *IoT* inspirou a criação de novas práticas ou modelos de processos e 23% deles creem que será provocada uma mudança significativa no modelo estratégico de negócio de suas empresas. A Figura 1 ilustra o restante das estatísticas mencionadas.

Figura 1 – Estatísticas de Crescimento da *IoT* ((BARROS, 2019) Adaptado de (WITCHALLS; CHAMBERS, 2013)).

Figura 2 – Estatísticas sobre o Impacto do Crescimento da *IoT* ((BARROS, 2019) Adaptado de (WITCHALLS; CHAMBERS, 2013)).

É fato que a Internet das Coisas têm revolucionado o modo como vivemos a cada dia impactando em diversas áreas da indústria, academia e estimulando a economia como já mencionado anteriormente. Entre as diversas áreas que a *IoT* tem impulsionado, as aplicações de saúde e fabricação têm se destacado gerando maior impacto econômico. Aplicações de saúde, tais como: saúde móvel *m-Health* e *Telecare*, que permitem o atendimento médico, prevenção, diagnóstico, tratamento e serviços de monitoramento a serem entregues de forma eficiente através de mídia eletrônica deverão gerar crescimento de 1,1 a 2,5 trilhões de dólares anualmente pela economia global até 2025 (MANYIKA *et al.*, 2013).

Baker, Xiang e Atkinson (2017) apresentam um estudo com os principais desafios e oportunidades em ambientes de *Smart Healthcare*. A Figura 3 mostra a parcela de mercado projetada em aplicações *IoT* em diferentes ambientes, destacando o seu uso em aplicações de *Smart Healthcare*.

2.2.1 Arquitetura *IoT*

Na literatura não há um consenso por parte dos autores em relação às camadas da arquitetura *IoT*. Grande parte dos autores abordam uma arquitetura de três camadas, conforme a Figura 4 apresentando a camada de percepção, rede e aplicação. No entanto, alguns pesquisadores adicionaram duas camadas novas, sendo a camada de *middleware* e de negócio, tal como apresentada na Figura 4. A camada de mais baixo nível é a camada de percepção, similar a camada física do modelo de referência *OSI* do inglês (*Open System Interconnection (OSI)*). Essa camada é responsável por apresentar os dispositivos utilizados para coleta de dados, como por exemplo: sensores, rótulos de códigos de barras, dispositivos de radio-frequência, sistemas de

Figura 3 – Impacto do Crescimento da *IoT* na economia (Adaptado de (MANYIKA *et al.*, 2013)).

posicionamento global do, inglês (*Global Positioning System (GPS)*), câmeras de segurança e muitos outros (KHAN *et al.*, 2012) (AAZAM; HUH, 2014b) (FERSI, 2015) (HINAI; SINGH, 2017).

A camada de rede é responsável pelo transporte dos dados entre a camada de percepção para as camadas adjacentes. Muitas tecnologias de transmissão de dados são utilizadas nesta camada como redes cabeadas e redes sem fio tais como: *3G/4G*, *Wifi*, *WiMax*, *Bluetooth*, *ZigBee*, *UMTS*, *Bluetooth*, *Infrared* e outras. A camada de *middleware* possui como função receber os dados da camada de rede e oferecer um serviço de armazenamento dos dados recebidos. Os serviços ainda devem prover o processamento de informações e tomar decisões automáticas segundo os dados coletados (KHAN *et al.*, 2012), (AAZAM; HUH, 2014b). A camada de aplicação possui a responsabilidade de apresentar os dados provenientes da camada de *middleware* ou rede. No entanto, é importante que os dados a serem apresentados estejam de acordo com as especificações do usuário, o que implica que esteja de acordo com as especificações definidas pelo Acordo de Nível de Serviço (*Service Level Agreement (SLA)*) adotado.

A camada de negócio é responsável pelo gerenciamento do sistema o que inclui todas as aplicações e serviços existentes. Ela define os modelos de negócio, os gráficos e os fluxos de execução conforme os dados recebidos da camada de aplicação. É a partir de uma análise de todos estes dados e fluxos gerados que se determinam a tomada de decisões futuras, bem como, quais estratégias serão adotadas (KHAN *et al.*, 2012). É importante destacar, que independente do modelo de arquitetura adotado, o processo de comunicação em rede está presente em todas as camadas existentes.

Figura 4 – Arquitetura 3 e 5 Camadas da *IoT* (BARROS, 2019) e (NUNES, 2018)

2.2.2 Computação em Nuvem

O ambiente de processamento dos dados provenientes de aplicações *IoT* pode variar bastante conforme a quantidade de dados gerados e o tempo de resposta desejados. Grandes volumes de dados são frequentemente processados utilizando Computação em Nuvem, podendo ser pública, privada ou mesmo híbrida. O conceito de Computação em Nuvem parte de um modelo de compartilhamento de recursos computacionais pela Internet em forma de serviço (ARMBRUST *et al.*, 2010). Entre estes recursos estão: Servidores, Redes, Armazenamento e outros. A virtualização é o processo utilizado para fazer todo o gerenciamento destes recursos computacionais. Zhang, Yan e Chen (2012) explicam que a virtualização permite que as configurações sejam facilmente alteradas de acordo com as necessidades de seus usuários. De acordo com Keshavarzi, Haghighat e Bohlouli (2013), muitas empresas fornecem serviços baseados em nuvem e muitas universidades estão usando estes serviços para suportar suas necessidades de infraestrutura mesclando em muitos casos nuvens públicas e privadas.

A computação realizada pela nuvem possui quatro modelos de implantação tais como: nuvens públicas, privadas, híbridas e comunitárias. Em nuvens públicas como *Azure*, *Amazon Web Services (AWS)*, *Google App Engine (GAE)* e *Salesforce.com* os recursos computacionais são disponibilizados como serviço para o público geral. Neste modelo de nuvem, a localização dos provedores podem estar em qualquer lugar do planeta. Entretanto, em nuvens privadas ou internas somente os usuários autorizados de uma organização podem usufruir dos serviços oferecidos pela organização, sendo, os recursos computacionais de uso exclusivo da companhia e os provedores gerenciados pela própria organização. *EMC*, *Eucalyptus* e *OpenStack* são algumas das ferramentas comuns utilizadas para a configuração de nuvens privadas (AHMED *et al.*, 2012), (BARRY, 2003), (KESHAVARZI; HAGHIGHAT; BOHLOULI, 2013).

As nuvens híbridas são uma mistura de nuvens privadas e públicas, em que, as organizações podem compartilhar partes de seus próprios recursos com fornecedores de nuvem, controlar seus próprios dados e protegê-los ao contrário do ambiente de nuvem pública. Alguns exemplos deste modelo estão a *Intel Hybrid Cloud*, o grupo imobiliário e o *InterContinental Hotels Group* (KESHAVARZI; HAGHIGHAT; BOHLOULI, 2013), (AHMED *et al.*, 2012). Por fim, uma nuvem comunitária é aquela onde os recursos computacionais são providos para um grupo de usuários que compartilham a mesma necessidade. Normalmente usuários de uma mesma comunidade. Os provedores deste modelo de nuvem podem pertencer a um membro da comunidade ou a terceiros (BARRY, 2003), (ATLAM *et al.*, 2017).

Para que os consumidores possam escolher o serviço que melhor lhes convém, a computação em nuvem oferece seus serviços em três níveis diferentes, a saber: O modelo de **Software como Serviço** (*Software as a Service (SaaS)*), em que o *software* é entregue pela Internet para os clientes como por exemplo, o *Google Apps* (ATLAM *et al.*, 2017). O modelo de **Plataforma como Serviço** (*Platform as a Service (PaaS)*), que oferece um ambiente onde os desenvolvedores podem escrever suas aplicações, testar e implantar *software* específico de acordo com as especificações da plataforma sem preocupações com os atributos de qualidade de serviço providos pela infraestrutura de nuvem. Um exemplo de aplicação de Plataforma como Serviço é o *Microsoft Azure* (PATIDAR; RANE; JAIN, 2012), (KESHAVARZI; HAGHIGHAT; BOHLOULI, 2013).

Por fim, o modelo de **Infraestrutura como Serviço** (*Infrastructure as a Service (IaaS)*), neste modelo que oferece serviços como recursos computacionais para armazenamento ou processamento com *hardware* e servidores como serviço. Neste modelo os usuários têm o controle e a configuração do Sistema Operacional entre outros acessos da infraestrutura como por exemplo, os *Serviços Web* da *Amazon* (MARINESCU, 2017).

A Internet das Coisas e a Computação em Nuvem são tecnologias que têm se desenvolvido rapidamente e embora possam trabalhar em conjunto, cada uma possui características próprias. A *IoT* é caracterizada por dispositivos amplamente distribuídos com recursos de processamento e armazenamento limitados, os quais normalmente enfrentam problemas relacionados ao desempenho, confiabilidade, privacidade e segurança (BOTTA *et al.*, 2016). Por outro lado, a Computação em Nuvem compreende uma rede com recursos de armazenamento e poder computacional teoricamente ilimitados. Além de permitir um ambiente flexível e robusto que permite a integração de dados diversos a partir de fontes de dados heterogêneas (ZHOU *et al.*, 2013).

Deste modo, a Computação em Nuvem têm sido utilizada para prover os recursos de processamento e armazenamento que grande parcela dos dispositivos *IoT* não são capazes de processar e armazenar. Babu, Lakshmi e Rao (2015) afirmam que a *IoT* e *Cloud* são duas tecnologias comparativamente desafiadoras e estão sendo combinadas para mudar o ambiente atual e futuro dos serviços de interligação de redes (*Internetworking*). Essa combinação de Computação em Nuvem e Internet das Coisas originou o termo conhecido como Nuvem de

Coisas do, inglês *Cloud Of Things*. Essa união tem permitido que novos modelos de serviços de nuvem sejam criados, fornecendo novos serviços para bilhões de dispositivos em diferentes ramos de aplicação (BABU; LAKSHMI; RAO, 2015), (BOTTA *et al.*, 2016). Contudo, os desafios existentes na Computação em Nuvem também são considerados para o ambiente de Nuvens de Coisas. Alguns exemplos dos modelos derivado da união das duas áreas são destacados a seguir:

- ***Sensing as a Service (S2aaS)***, permite o acesso aos dados do sensor;
- ***Ethernet as a Service (EaaS)***, o papel principal é fornecer conectividade onipresente ao controle de dispositivos remotos;
- ***Sensing and Actuation as a Service (SAaaS)***, fornece lógicas de controle automaticamente;
- ***Identity and Policy Management as a Service (IPMAaaS)***, fornece acesso às políticas e gestão de identidade;
- ***Database as a Service (DBaaS)***, fornece gerenciamento de banco de dados onipresente;
- ***Sensor Event as a Service (SEaaS)***, envia serviços de mensagens gerados por eventos de sensores;
- ***Sensor as a Service (Senaas)***, fornece gerenciamento de sensores remotos;
- ***Data as a Service (DaaS)***, fornece acesso onipresente a qualquer tipo de dados.

2.3 Desafios da Internet das Coisas

A revolução de dispositivos inteligentes na sociedade ao mesmo tempo que agregam valor às pessoas, indústrias e na economia, também enfrentam vários desafios que precisam ser mitigados. A fim de garantir a adequada difusão e adoção da *IoT*, é necessário abordar essas questões com cuidado. Muitos sensores ainda são caros o que limita o acesso a muitas pessoas, a vida útil da bateria continua sendo um problema e grandes fluxos de dados ainda não podem ser analisados em tempo real (RABAIEI; HAROUS, 2016). Al-Fuqaha *et al.* (2015) e Choudhary e Jain (2016) destacam os principais desafios de *IoT* como: **Confiabilidade, Mobilidade, Disponibilidade, Escalabilidade, Desempenho, Problemas de Interoperabilidade, Segurança e Gerenciamento dos Dados**. A seguir é realizado uma explanação envolvendo cada um destes desafios para um melhor esclarecimento.

1) Disponibilidade

A maioria das aplicações de Internet das Coisas requer disponibilidade em níveis de *hardware* e *software* para fornecer serviços que atendam aos consumidores (CHOUDHARY; JAIN, 2016). Disponibilidade implica que o serviço esteja sempre acessível em qualquer lugar

e a qualquer hora para os clientes. Deste modo, ela se torna um grande desafio principalmente quando alinhada à confiabilidade, uma vez que as falhas do sistema podem levar a uma falha geral do sistema *IoT*, e isso pode resultar em perdas financeiras e danos diversos. A disponibilidade de *hardware* refere-se à existência de dispositivos o tempo todo compatíveis com as funcionalidades e protocolos da *IoT*, ou seja, os protocolos devem ser incorporados nos dispositivos restritos que oferecem funcionalidades *IoT* (AL-FUQAHA *et al.*, 2015). Outras formas de prover disponibilidade é por meio de redundância para dispositivos e serviços críticos e o uso de ferramentas adequadas para maximizar a disponibilidade do sistema de modo que minimize os efeitos causados por falhas dos dispositivos de rede (SILVA *et al.*, 2013) e (MACEDO; GUEDES; SILVA, 2014).

2) Confiabilidade

A confiabilidade refere-se à capacidade do sistema de executar de forma consistente, conforme sua especificação (MACEDO; GUEDES; SILVA, 2014). De acordo com Choudhary e Jain (2016), ela visa aumentar a resistência e a taxa de sucesso do serviço *IoT*, como por exemplo, a resistência de problemas de segurança, usabilidade a longo prazo, robustez da aplicação em caso de informações incertas e etc. A confiabilidade se torna ainda mais crítica com requisitos mais rigorosos quando se trata do campo de aplicações que carecem de respostas de emergência (MAALEL *et al.*, 2013). Al-Fuqaha *et al.* (2015) afirmam que a confiabilidade deve ser implementada em *software* e *hardware* em todas as camadas da *IoT*. Para uma aplicação eficiente de *IoT*, a comunicação precisa ser confiável, pois caso contrário, a coleta, o processamento e a transmissão dos dados podem levar a longos atrasos, perda de dados e eventualmente a decisões erradas, o que pode acarretar a cenários desastrosos. Alguns trabalhos envolvendo confiabilidade em *IoT* são discutidos em (LI *et al.*, 2012), (GANZ *et al.*, 2012), (MAALEL *et al.*, 2013).

3) Mobilidade

A mobilidade é outro desafio para as implementações de *IoT*, visto que, de acordo com Choudhary e Jain (2016), a maioria dos serviços são entregues aos usuários móveis. Al-Fuqaha *et al.* (2015) explicam que um dos grandes desafios em Internet das Coisas é suportar a mobilidade de *IP* em conectar usuários com seus serviços desejados, quando eles estão se movendo para diferentes locais e alternando entre redes de acesso, pois, a interrupção do serviço pode ocorrer quando estes dispositivos movem de um *gateway* para outro. Balte, Kashid e Patil (2015a) adicionam outros problemas que envolvem a segurança dos dispositivos móveis como por exemplo, a identificação e autenticação de dispositivos, armazenamento e troca de chaves e credenciais. Muitos trabalhos na literatura visam solucionar os desafios envolvendo a mobilidade como por exemplo, os apresentados em (ELSALEH; GLUHAK; MOESSNER, 2011), (ZHU; ZHANG; WAKIKAWA, 2011), (GANZ *et al.*, 2012), (MISRA; AGARWAL, 2012) e (FU *et al.*, 2014).

4) Interoperabilidade

Este termo está relacionado ao crescimento do número de dispositivos inteligentes. Em *IoT* este desafio é devido à necessidade de lidar com uma grande quantidade de coisas heterogêneas que pertencem as diferentes plataformas conforme explicam [Al-Fuqaha et al. \(2015\)](#). A Interoperabilidade é definida como a capacidade dos objetos para conectar e comunicar informações de forma que possam ser rapidamente processados e facilmente acessíveis por outras entidades. ([CHOUDHARY; JAIN, 2016](#)) afirmam que a interoperabilidade já deve ser pensada pelos desenvolvedores de aplicações e pelos próprios fabricantes dos dispositivos *IoT* para garantir a entrega dos serviços para todos os clientes, independentemente das especificações da plataforma de *hardware* usada. Além disso, as aplicações devem ser construídas de modo que permita a adoção de novas funções sem causar problemas em outras, mantendo a integração com diferentes tecnologias de comunicação. Alguns dos trabalhos envolvendo à interoperabilidade são encontrados em ([DUNKELS; ERIKSSON; TSIFTES, 2011](#)) e ([DUNKELS et al., 2011](#)).

5) Escalabilidade

Conforme a tecnologia cresce, os clientes desejam novos dispositivos e serviços com alguma funcionalidade diferente. A escalabilidade em Internet das Coisas refere-se à capacidade de adicionar novos dispositivos, serviços e funções ou mesmo cargas de trabalho de modo que não afete negativamente a qualidade dos serviços prestados ([GOMES; RIGHI; COSTA, 2014](#)). Segundo [Al-Fuqaha et al. \(2015\)](#), adicionar novos dispositivos nas aplicações não é uma tarefa fácil, especialmente devido as diversas plataformas de *hardware* e protocolos de comunicação existentes. Considerando tais características, as aplicações de *IoT* devem ser projetadas desde o início para permitir serviços e operações extensíveis. Algumas soluções envolvendo escalabilidade em Internet das Coisas são tratadas em ([UCKELMANN et al., 2010](#)), ([SARKAR et al., 2014](#)), ([LI et al., 2013](#)) e ([SINGH, 2013](#)).

6) Desempenho

Avaliar o desempenho dos serviços e dispositivos de *IoT* é um grande desafio, devido a relação de dependência de muitos componentes bem como do desempenho de tecnologias subjacentes ([AL-FUQAHA et al., 2015](#)). A avaliação de todos os serviços não é uma tarefa trivial, é necessário testar e monitorar com precisão toda a computação interna entre dispositivos e infraestrutura de rede. Entre os recursos monitorados, estão: perda de pacotes, largura de banda, velocidade de processamento, custo de energia, entre outros para fornecer o melhor desempenho possível a um preço acessível para os clientes ([CHOUDHARY; JAIN, 2016](#)). Ainda de acordo com [Choudhary e Jain \(2016\)](#), para o avanço da indústria de Internet das Coisas é importante que se avalie continuamente os serviços prestados. Muitas soluções foram investigadas na literatura, dentre elas estão os trabalhos de: ([SIEKKINEN et al., 2012](#)), ([CARO et al., 2013](#)), ([FERNANDES et al., 2013](#)), ([RAZA et al., 2014](#)) e ([THANGAVEL et al., 2014](#)).

7) Segurança

[Al-Fuqaha et al. \(2015\)](#) afirmam que problemas relacionados a segurança desempe-

nham um papel significativo em todos os mercados devido à sensibilidade da privacidade dos consumidores. Além disso, avaliar o desempenho dos serviços *IoT* é um desafio fundamental (UCKELMANN, 2012) pois está diretamente ligado à segurança, onde, um aumento de segurança causa uma diminuição do desempenho. Em redes heterogêneas, como no caso de Internet das Coisas não é fácil garantir a segurança e a privacidade dos usuários, Ishaq *et al.* (2013) afirmam que um dos problemas abertos na segurança da *IoT* ainda não abordado, se refere a distribuição das chaves entre os dispositivos, e Skarmeta, Hernandez-Ramos e Moreno (2014) afirmam que os problemas relacionados à segurança têm se destacado em relação aos demais. Diante disso, Pawar e Ghumbre (2016) enumera os principais desafios de segurança como: **Confidencialidade, Integridade, Autenticação, Privacidade, Confiança, Aplicação de Políticas e Controle de Acesso**. Estes desafios são detalhados a seguir em uma seção específica de desafios envolvendo a Segurança em Internet das Coisas.

2.4 Segurança em Internet das Coisas

Nesta Seção, detalhamos os principais desafios que envolvem a segurança de dados na Internet das Coisas. Destacamos estes desafios por serem considerados por muitos autores como os mais importantes (SKARMETA; HERNANDEZ-RAMOS; MORENO, 2014). Em seguida, destacamos as soluções relacionadas com o Controle de Acesso, apresentando os seus principais modelos e protocolos. Essa escolha é devido as características necessárias para prover um controle de acesso eficiente o que envolve indiretamente os demais desafios de segurança. Outro fator importante a se considerar é que soluções de controle de acesso diferentes podem ser aplicadas em cenários distintos. Nem sempre uma solução única, pode ser eficiente para todo ambiente de aplicação. Considerando estes fatores, discutimos as principais soluções de controle de acesso aplicada no contexto de Internet das Coisas.

1) Confidencialidade

Conforme Mendez, Papapanagiotou e Yang (2017) a confidencialidade garante que os dados estão disponíveis apenas para usuários autorizados e não podem ser espionados ou interferidos por usuários não autorizados. No ambiente de *IoT*, a confidencialidade é um importante princípio de segurança, porque um grande número de dispositivos de medição como sensores podem conter dados sigilosos. Deste modo, é fundamental garantir que os dados coletados por um dispositivo de medição não revelem informações secretas aos dispositivos vizinhos. Miorandi *et al.* (2012) definem a confidencialidade dos dados como uma questão fundamental para soluções em *IoT*. Os autores mencionam ainda o gerenciamento adequado de identidade um fator chave para garantir a confidencialidade dos dados. Por outro lado, (ORACEVIC; DILEK; OZDEMIR, 2017b) complementa que confidencialidade é uma das propriedades mais solicitadas em aplicativos de redes de sensores sem fio (*wireless sensor network (WSN)*).

2) Integridade

A Integridade em Internet das Coisas lida com falhas e danos físicos. Em uma primeira visão, ela inclui a preservação contra sabotagem e o uso de unidades ou componentes falsificados (SADEGHI; WACHSMANN; WAIDNER, 2015). Outros fatores críticos são relacionados a robustez e as capacidades de tolerância a falhas (MIORANDI *et al.*, 2012). Entretanto, a integridade dos dados também é assegurada por soluções baseadas em senhas, porém, é possível vulnerabilidades relacionadas ao comprimento da senha e aleatoriedade. Além disso, muitos sistemas *IoT* não suportam soluções criptográficas, devido a limitação dos recursos computacionais disponíveis. Algumas abordagens para prover integridade em ambientes *IoT* envolvem valores de *hash* que são armazenados externamente para evitar compromissos e soluções baseadas em *hardware*, como por exemplo, soluções baseadas em uso de chaves simétricas ou assimétricas, conhecidas também como *Trusted Platform Module (TPM)* (FONGEN, 2012), (MENDEZ; PAPAPANAGIOTOU; YANG, 2017).

3) Autenticidade

De acordo com Oracevic, Dilek e Ozdemir (2017b) autenticação é o processo de determinar se uma mensagem é, de fato, de onde afirma ser, ou que seja capaz de comprovar sua própria identidade. Além disso, por meio do processo de autenticação é possível determinar se um usuário ou mensagem é realmente verdadeiro. Uma forma de fazer este processo é através de mensagens (*Message Authentication Code (MAC)*), que por sua vez é uma informação utilizada para autenticar uma mensagem e fornecer garantias de integridade e autenticidade. Sain, Kang e Lee (2017) afirmam que, com a ajuda da autenticação de dois fatores, como por exemplo a autenticação de dois passos do *Google*, o processo de prover autenticidade e autorização se torna mais simples principalmente com a ajuda do celular que sempre permanece com os usuários. Ghorbani e Ahmadzadegan (2017) explicam que, embora a criptografia de chave pública tenha criado vantagens para esquemas de autorização e autenticação, a falta de uma referência à certificação global será um grande desafio para aferição da autenticidade.

4) Privacidade

Os dispositivos *IoT* estão espalhados em nosso cotidiano em *Smart TV's*, *Smartwatches*, câmeras de segurança entre outros. Suresh *et al.* (2014) afirmam que toda essa quantidade de dispositivos inteligentes podem gerar informações valiosas que podem ser coletadas pelos fabricantes, fornecedores ou outras partes interessadas. Muitos destes dados se referem as informações de localização, dados relacionados a condições de saúde, áudios ou mesmo informações confidenciais adquiridas por acesso não autorizado em *RFID* e *Electronic Product Code (EPC)* (FAROOQ *et al.*, 2015).

Com mais dispositivos *IoT* conectados à Internet, mais da privacidade dos usuários serão expostas ao público. Portanto, invasores mal-intencionados podem surgir para tirar proveito dessas falhas e terem vantagens sobre os outros (REN *et al.*, 2017). Muitos destes dispositivos como sensores, coletam os dados e em seguida armazenam dados na nuvem ou em outros tipos de infraestruturas computacionais. É importante esclarecer que mesmo que o processo de

armazenamento esteja na nuvem, ainda assim, o proprietário do dado poderá perder o controle dos seus dados e devido a isso, há os que afirmam que a *IoT* traz mais preocupações de privacidade do que provê benefícios (LEE; LEE, 2015).

5) Aplicação de Políticas

Aplicação de políticas implica na utilização de regras que desejam ser atuadas com o objetivo de prover ordem, segurança e consistência dos dados. Os trabalhos que envolvem aplicação de políticas em um ambiente de Internet das Coisas que apresentam soluções definitivas não são muitos. Apenas poucos trabalhos descrevem como controlar a aplicação de políticas de maneira definitiva. Um exemplo deste tipo de trabalho é descrito do artigo de Neisse, Steri e Baldini (2014), sendo portanto necessários mais esforços nesta área para que novos trabalhos apareçam. Balte, Kashid e Patil (2015b), afirmam que muitos trabalhos foram realizados para definir as linguagens convencionais para a especificação de políticas de privacidade, embora uma versão aprovada da linguagem que possa ser aplicada ao paradigma *IoT* ainda seja insuficiente.

6) Gerenciamento de Confiança

Em Sicari *et al.* (2015), os autores explicam que o conceito de confiança é usado em vários contextos e com diferentes explicações. Isso se dá pelo fato de não haver um consenso definitivo na literatura científica embora sua importância seja amplamente reconhecida. Um problema central com muitas aplicações de confiança é que elas não contribuem para a demonstração de métricas e metodologias de avaliação. Balte, Kashid e Patil (2015b) complementam alguns desafios abertos no gerenciamento de confiança como por exemplo: a introdução de uma linguagem de negociação de confiança bem definida, um mecanismo de negociação de confiança para o controle de acesso ao fluxo de dados e a definição de um bom sistema de gerenciamento de identidade de objetos.

As soluções disponíveis exploram diferentes técnicas para lidar com o problema de confiança no cenário *IoT*. Entre tais propostas destacam-se: o modelo hierárquico, mecanismos de reputação, abordagens derivadas de redes sociais, técnicas difusas, mecanismos baseados em comportamentos passados de nós ou estratégias de roteamento (SICARI *et al.*, 2015). Além disso, um estudo extenso em gerenciamento de confiança pode ser encontrado em Yan, Zhang e Vasilakos (2014).

2.5 Controle de Acesso

As tecnologias *IoT* podem ser aplicadas em uma variedade de ambientes, não sendo possível prever todos os seus domínios de aplicação. Cada domínio de aplicação possui particularidades e requisitos próprios de segurança relevantes, que muitas vezes não são aplicáveis em outro domínio. Ouaddah *et al.* (2017) caracterizam estes domínios em 3 classificações principais:

1. Pessoal e Doméstico: na escala individual, domiciliar e de saúde;

2. Governo e Serviços Públicos: na escala da nação e região da comunidade;

3. Empresa e Indústria: na escala de indústrias e grandes empresas.

Em cada um destes domínios os desafios antes mencionados são levantados, podendo ser mais ou menos importantes conforme o contexto inserido. Segundo [Ouaddah et al. \(2017\)](#), é improvável que uma abordagem *one-size-fits-all* funcione para todo domínio de aplicação em *IoT*. Ao contrário, deve-se elaborar muitas abordagens diversas. Deste modo, é necessário que se analise as vulnerabilidades específicas e o nível de segurança exigido para cada dispositivo em determinada aplicação. Assim, cada ambiente pode necessitar de um modelo de controle de acesso diferente que melhor se adapte ao seu próprio ambiente. Partindo dessa premissa, as principais soluções de controle de acesso em Internet das Coisas são apresentadas a seguir.

1) Controle de Acesso Baseado em Usuário (UBAC)

[Amir, Pillai e Hu \(2014\)](#) afirmam que o *User-Based Access Control (UBAC)* é o modelo de controle de acesso mais básico, que constitui a base para padrões de segurança como o (*Security Assertion Markup Language (SAML)*). Nesta abordagem o usuário controla o acesso aos recursos. Assim, quando existe a necessidade de compartilhamento público, este modelo se torna bastante problemático. As políticas de acesso estão associadas as contas de usuários, ou seja, se uma conta de usuário for revogada ou excluída, as políticas relacionadas também desaparecem e precisam ser geradas novamente para usuários sucessivos que possam assumir o mesmo papel. Embora seja considerado um dos modelos mais seguros, visto que cada usuário só pode acessar seus próprios recursos, em um ambiente de Internet das Coisas ele se torna inviável, devido as suas numerosas restrições. É adequado para aplicações em que os usuários e suas funções dentro do domínio do aplicativo permanecem constantes durante um longo período de tempo, como por exemplo, em um ambiente de negócios ([AMIR; PILLAI; HU, 2014](#)).

2) Controle de Acesso Baseado em Regras (RBAC)

O controle de acesso baseado em regras ou funções *Role-based Access Control (RBAC)*, é um modelo de controle de acesso que foi introduzido para suprir algumas desvantagens do modelo de controle de acesso baseado em usuário ([GUSMEROLI; PICCIONE; ROTONDI, 2013a](#)). Este modelo baseia-se na premissa de funções que pertencem aos usuários e possui privilégios de acesso ([CHEN; DAHANAYAKE, 2007](#)). Este modelo é composto por quatro diferentes componentes e cada um deles atribui uma série de funcionalidades ao *RBAC*. Estas funcionalidades são: o *RBAC* principal, o *RBAC* hierárquico, a separação estática das relações de dever e a separação dinâmica das relações de dever ([OUADDAH et al., 2017](#)). O esquema baseia-se na hipótese de que os papéis e as responsabilidades permanecem em grande parte constantes dentro de uma organização e são os usuários que mudam, ou seja, as funções são atribuídas aos usuários e as permissões são atribuídas às funções.

Portanto, a modelagem de políticas de acesso por meio de funções de usuário em vez de identidades de usuários fornece uma solução mais conveniente e sustentável ([AEDO; DÍAZ;](#)

SANZ, 2006). Amir, Pillai e Hu (2014) afirmam que o *RBAC* é baseado em funções, que por sua vez são vinculadas aos usuários e, portanto, promove um esquema centrado no usuário que proíbe em grande parte o acesso anônimo de recursos por agentes não registrados. De acordo com Ouaddah *et al.* (2017), o modelo *RBAC* puro não é adequado para modelar políticas de segurança que representem cenários *IoT* complexos e ambíguos. Portanto, ele deve ser estendido para enfrentar os desafios existentes no ambiente de Internet das Coisas. Algumas soluções que estendem este modelo podem ser encontradas em (ZHANG; TIAN, 2010), (JINDOU; XIAOFENG; CHENG, 2012), (LIU; XIAO; CHEN, 2012) e (BARKA; MATHEW; ATIF, 2015).

3) Controle de Acesso Baseado em Atributo (ABAC)

Este modelo é bastante semelhante ao *UBAC*, na forma que ainda é baseado na existência de usuários e a diferença está no fato de que o *Attribute-based Access Control (ABAC)*, em vez de usar a identidade de um usuário, ele se concentra em fazer uso de dados de autorização reais, ou seja, aquilo que o usuário pode fazer (AMIR; PILLAI; HU, 2014). Desta maneira, elimina os problemas relacionados ao gerenciamento de identidade distribuída, uma vez que cada domínio de serviço participante possui informações sobre seus próprios serviços e as autorizações relevantes necessárias para a realização desses serviços.

Ouaddah *et al.* (2017) explicam que o *ABAC* é composto por dois aspectos: o modelo de política às vezes referido como *Policy Based Access Control (PBAC)* e o modelo de arquitetura que aplica a política. Em sua aplicação básica, afirma que o acesso pode ser determinado com base em vários atributos apresentados por um sujeito, já as regras de políticas especificam as condições em que o acesso é concedido ou negado. Mesmo este modelo promovendo melhorias em relação ao *UBAC*, ele continua sendo centrado no usuário e, portanto, não é adequado para grandes aplicações de *Web of Things (WoT)*, em que o acesso aos recursos por agentes anônimos é de extrema importância. Uma solução envolvendo este modelo de controle de acesso pode ser encontrada em (YE *et al.*, 2014).

4) Controle de Acesso Baseado no Controle de Uso (UCON)

O modelo de controle de acesso baseado no controle de uso (*Access control solutions based on usage control (UCON)*), foi proposto em Park (2003), Zhang *et al.* (2005) e por sua vez é considerado como a próxima geração de modelos de controle de acesso. Várias novidades aparecem quando comparadas aos modelos tradicionais como *RBAC* e *ABAC*. Ouaddah *et al.* (2017) explicam que o grande diferencial do modelo de controle de uso está no fato de lidar com o problema da autorização de forma contínua antes da execução do acesso, no momento da execução e após a execução. Outra importante característica está na mutabilidade do atributo, o que implica dizer, se caso o atributo de acesso for alterado enquanto o acesso estiver em andamento e essa alteração leva a uma insatisfação da política de segurança, o acesso concedido é revogado e cancelado.

Mais detalhes com relação a este modelo podem ser encontrados no trabalho de (LA-

ZOUSKI; MARTINELLI; MORI, 2010). Vale ressaltar, que o único trabalho encontrado que enfatizou as particularidades do modelo *UCON* aplicado no cenário de Internet das Coisas foi o de (GUOPING; WENTAO, 2011). As particularidades residem no conceito de continuidade da decisão e mutabilidade dos atributos introduzidos pelo modelo *UCON*. (OUADDAH *et al.*, 2017) complementam que as soluções baseadas neste modelo possuem alto nível de granularidade e consciência do contexto, graças à alta expressividade, por outro lado, possui baixo nível de distribuição, usabilidade e leveza.

5) Controle de Acesso Baseado no Modelo Organizacional (OrBAC)

O modelo de controle de acesso organizacional (*organization-based access control* (*OrBAC*)) (KALAM *et al.*, 2003), foi proposto para abordar problemas existentes nos modelos *RBAC* estendidos. O conceito de organização é introduzido como uma entidade que realiza a separação do nível concreto, físico como usuário, objeto e ação do nível abstrato como as funções, atividades e visualizações. Ouaddah *et al.* (2017) explicam que o modelo *OrBAC* inclui diferentes dados de contexto que podem ser históricos, espaciais e temporais ou mesmo dados que podem ser declarados pelo usuário. O controle de acesso organizacional é baseado no modelo de entidade-relacionamento e desta forma, as entidades e as relações podem estar associadas aos atributos. Este modelo foi idealizado para estruturas centralizadas e não abrange as necessidades de interoperabilidade, colaboração, heterogeneidade e distribuição.

Alguns modelos foram criados a partir do controle de acesso organizacional como o *Multi-OrBAC* (*MultiOrganization-Based Access Control* (*Multi-OrBAC*)) (BAINA; LAAROUCHI, 2012) projetado para multi-organizações, o *Poly-OrBAC* (*access control framework for Inter-Organizational Web Services* (*Poly-OrBAC*)), que integra o *OrBAC* com políticas internas de cada organização e tecnologias *Web* para garantir a interoperabilidade entre organizações (KALAM *et al.*, 2009), o *Integrity-OrBAC* (KALAM; HASSANI; OUAHMAN, 2013) que visa responder de forma mais realista às necessidades das infraestruturas críticas, em particular a integridade.

Uma aplicação mais recente deste modelo é encontrada no trabalho de (ASCHOFF *et al.*, 2017) que combina o *OrBAC* com redes definidas por *software* (*Software Defined Networking* (*SDN*)). Por fim, outra solução conhecida como *SmartOrBAC*, visa estender o modelo de controle de acesso organizacional tradicional para ambientes de Internet das Coisas (OUADDAH *et al.*, 2015) e (BOUIJ-PASQUIER *et al.*, 2015) no entanto, ele é complexo para ser suportado por dispositivos restritos.

6) Controle de Acesso Baseado em Capacidade (CAPBAC)

O conceito de capacidade não é atual, sendo introduzido inicialmente em Dennis e Horn (1966) como um *token*, *ticket* ou chave que dá ao possuidor permissão para acessar uma entidade ou objeto em um sistema informático. Deste modo, o CAPBAC é baseado no conceito de capacidade que contém direitos concedidos à entidade que o possui. Este modelo têm sido adotado em muitos projetos de larga escala e têm sido amplamente utilizado em aplicações

de Internet das Coisas (OUADDAH *et al.*, 2017). Ouaddah *et al.* (2017) salientam que aplicar o modelo original em *IoT* trouxe várias desvantagens, como por exemplo a propagação e a revogação da capacidade que foi apontado em Gong (1989). A ideia proposta por Gong é chamada de Sistema de Capacidade de Identidade Segura (*Secure Identity-based Capability System (ICAP)*) que visa basicamente estender o conceito de capacidade em qualquer usuário ou assunto que deseja acessar um determinado dispositivo ou recurso.

A particularidade introduzida consiste em incluir a identidade de sujeito ou usuário em sua operação (OUADDAH *et al.*, 2017). Várias soluções levando em consideração estes dois modelos foram propostas, tais como em: (MAHALLE *et al.*, 2013a), (ANGGOROJATI *et al.*, 2012), (ANGGOROJATI *et al.*, 2013), (GUSMEROLI; PICCIONE; ROTONDI, 2013b), (HERNÁNDEZ-RAMOS *et al.*, 2013) e (HERNÁNDEZ-RAMOS *et al.*, 2016). Ouaddah *et al.* (2017) explicam que levando em consideração sob a perspectiva de *IoT*, o modelo *CAPBAC* apresenta vantagens de grande escalabilidade, flexibilidade, simplicidade e usabilidade. A possibilidade de utilizar *tokens* de acesso pode ser adaptado em diversos domínios de aplicação, bastando apenas limitar ou estender o tempo de vida dos *tokens*, além de, possibilitar a delegação e revogação de permissões conforme necessário.

Um modelo similar ao *CAPBAC* é conhecido como controle de acesso baseado em *token* (*Token-Based Access Control (TBAC)*), que permite acesso a um conjunto ou grupo de recursos para um determinado usuário que possui o *token* (KHALED *et al.*, 2010). Os *tokens* podem ser configurados para expor apenas os recursos relevantes sem que a identidade do proprietário do recurso seja revelada, o que é vantajoso se comparado ao *UBAC* que exige que a identidade do usuário seja transmitida com um pedido. De acordo com Amir, Pillai e Hu (2014), este modelo fornece um mecanismo de controle de acesso centrado em recursos desacoplados que é capaz de dimensionar de forma eficiente o ambiente dinâmico da *Web of Things*. Segundo os mesmos autores, o *TBAC* possui uma segurança menor, pois, oferece autenticação de única etapa, ou seja, apenas um *token* é suficiente para ter acesso ao recurso, no entanto, os outros modelos tendem a possuir autenticação de duas etapas como *login* e *password* o que pode melhorar a segurança.

7) Controle de Acesso Baseado em Outros Modelos

Tsiatsis *et al.* (2010) é um projeto de grande escala, que foi desenvolvido como parte do projeto *SENSEI ICT-FP7* europeu. Este modelo desenvolveu uma estrutura geral de autenticação, autorização e contabilidade. Sua arquitetura baseia-se em uma série de conceitos avançados como gerenciamento de confiança, escalabilidade para gerenciamento de identidade, análise de privacidade e preservação de privacidade e por fim, estabelecimento de reputação. Duas abordagens são oferecidas, o controle de acesso e a confiança que são noções intimamente relacionadas. Mahalle *et al.* (2013b) propõem usar o gerenciamento de confiança como uma ferramenta na tomada de decisão do controle de acesso com base em uma abordagem *Fuzzy*.

Um modelo de confiança baseado em lógica difusa chamado de *TACIoT (Trust-Aware Access Control System (TACIoT))*, é apresentado em Bernabe, Ramos e Gomez (2016). Este

modelo segue uma abordagem multidimensional para permitir uma computação de valor de confiança que seja precisa para dispositivos *IoT*. Ele leva em consideração quatro dimensões: reputação, qualidade de serviço, relacionamento social entre dispositivos *IoT* e aspectos de segurança.

Além disso, o modelo proposto foi integrado em um mecanismo de controle de acesso leve e flexível baseado no trabalho de [Hernández-Ramos et al. \(2016\)](#) com o objetivo de tornar os dispositivos *IoT* conscientes dos resultados de confiança de outros dispositivos e controlar suas decisões de controle de acesso. De acordo com [Ouaddah et al. \(2017\)](#) *TACIoT* fornece um mecanismo de segurança de ponta a ponta e confiável para dispositivos *IoT* com base em um mecanismo de autorização leve e um novo modelo de confiança que foi especialmente projetado para ambientes de Internet das Coisas.

[Neisse et al. \(2014\)](#) propõem um *kit* de ferramentas de segurança chamado de *Seckit*. Ele fornece a especificação e aplicação da política de controle de uso, incluindo autorizações e obrigações para permitir a proteção de dados do usuário. A especificação das regras de política de autorização e obrigação é feita usando um modelo de regra baseado em eventos, contendo modelos de regras. [Ouaddah et al. \(2017\)](#) explicam que a base de dados do *SecKit* é uma coleção de meta-modelos que descrevem dados, tempo, identidade, papel, estrutura, comportamento, confiança, regra e risco.

2.6 Protocolos e Frameworks

Do mesmo modo que muitas soluções de controle de acesso tradicionais têm sido adaptadas para ambientes de *IoT*, alguns protocolos e *frameworks* que auxiliam na aplicação de controle de acesso têm sido utilizados com este mesmo objetivo. Entre os protocolos e *frameworks* existentes, dois deles se destacam. O protocolo *OAuth* que posteriormente se tornou um *framework* aberto para autorização na Internet e o *XACLM* que são apresentados a seguir.

2.6.1 Linguagem de Marcação de Controle de Acesso Extensível (XACML)

XACML do inglês (*Xtensible Access Control Markup Language (XACML)*) ([RISSANEN et al., 2013](#)) é uma linguagem baseada em *XML* para controle de acesso que foi padronizada pelo *OASIS (Organization for the Advancement of Structured Information Standards (OASIS))*. Um estudo completo foi realizado em [El-Aziz e Kannan \(2013\)](#), mas, basicamente ela descreve um idioma de política e decisões de controle de acesso. Este modelo não foi idealizado especificamente para uso em dispositivos restritos, embora haja muitos trabalhos utilizando este modelo como o de [Seitz, Selander e Gehrmann \(2013\)](#) que propõe uma estrutura genérica para autorização em *IoT*. Ele suporta controle de acesso fino e flexível para qualquer objeto com recursos de baixa potência e memória.

[Ouaddah et al. \(2017\)](#) afirmam que com base nos padrões atuais de Internet e controle de acesso como *XACML* e *SAML* ([MALER; MISHRA; PHILPOTT, 2003](#)) a decisão de controle de acesso é aplicada localmente pelo objeto (PEP local) levando em consideração seus parâmetros locais. [Ouaddah et al. \(2017\)](#) complementam que devido à complexidade da avaliação da política *XACML* para objetos com altas restrições de recursos, uma grande parte do processo de tomada de decisão para a concessão de acesso é externalizada (PDP externalizado). Em [Marra et al. \(2017\)](#) é proposto um controle de acesso baseado no modelo de controle de uso para *IoT* chamado *UCIoT (Usage Control in IoT (UCIoT))*. Este modelo traz as funcionalidades de uma estrutura baseada em *U-XACML* em uma arquitetura descentralizada, distribuída e *P2P (Peer-to-Peer (P2P))*.

2.6.2 OAuth 2.0

OAuth, é uma estrutura de controle de acesso que possibilita autorização padronizada para clientes que acessam recursos em servidores *Web*. O *OAuth 1.0 Protocol* ([HAMMER-LAHAV, 2010](#)) e o seu sucessor, o *OAuth 2.0 Framework* ([HARDT, 2012](#)) foram projetados para abordar questões de privacidade e prover controle de acesso em aplicações de grande escala conectadas à Internet. [Emerson et al. \(2015\)](#) definem o *OAuth2* como um protocolo de autorização aberto com uma maneira simples e segura de autenticar usuários e permitir o acesso aos seus dados protegidos, dando ao serviço um *token* de acesso. Ele foi projetado para permitir que um aplicativo de terceiro tenha acesso aos recursos restritos sob o controle de usuário, sem requerer compartilhamento de credenciais do usuário.

[Ouaddah et al. \(2017\)](#) afirmam existir mais de um bilhão de contas de usuários baseadas em *OAuth* fornecidas pelos principais provedores de serviços, como *Facebook*, *Google* e *Microsoft*. [Sciancalepore et al. \(2017\)](#) explicam que a arquitetura do *OAuth 2.0* engloba quatro papéis para prover autorização, que são:

1. O Proprietário do recurso capaz de conceder ou negar o acesso a um recurso protegido. É o usuário que autoriza uma aplicação a acessar sua conta. O acesso da aplicação à conta do usuário é limitado ao “escopo” da autorização concedida (por exemplo acesso para leitura ou escrita);
2. Um aplicativo de terceiro disposto a acessar recursos protegidos ou seja uma aplicação Cliente. O cliente é a aplicação que quer acessar a conta do usuário. Antes de fazer isso, ela deve ser autorizada pelo usuário e a autorização deve ser validada pela API;
3. O Servidor de autorização que controla o processo de autorização. Ele verifica a identidade do usuário e então emite *tokens* de acesso para aplicação;
4. O Servidor de recursos que expõe recursos protegidos, ou seja, ele é quem hospeda as contas de usuários protegidas e somente irá liberar o acesso a partir de um *token* validado.

A Figura 5 apresenta um diagrama abstrato de como estes papéis se interagem.

Figura 5 – Fluxo abstrato do Protocolo OAuth 2.0

1. Aplicação cliente solicita ao usuário uma autorização para acessar seus recursos protegidos;
2. Se o usuário autorizar a solicitação, a aplicação cliente recebe uma concessão de autorização;
3. A aplicação cliente solicita um *token* de acesso ao servidor de autorização (API) através da autenticação de sua própria identidade e da concessão de autorização fornecida anteriormente pelo usuário;
4. Se a identidade da aplicação cliente está autenticada e a concessão de autorização fornecida pelo usuário for válida, o servidor de autorização (API) emitirá um *token* de acesso para a aplicação cliente;
5. De posse do *token*, a aplicação cliente solicita o recurso desejado do usuário ao servidor de recursos (API) e apresenta o *token* de acesso para autenticação que recebeu do servidor de autorização;
6. Se o *token* de acesso for válido, o servidor de recurso (API) fornece o recurso solicitado para a aplicação cliente que exibirá ao usuário.

O fluxo real desse processo é variável nos quatro primeiros passos dependendo do tipo de concessão de autorização que se utiliza. Esse é apenas um modelo geral e abstrato. O OAuth 2 define vários tipos de concessões, cada uma das quais é útil e aplicável para diferentes casos.

Entre os principais tipos de concessões estão: Código de Autorização, Implícito, Credenciais de Senha do Proprietário do Recurso, Credenciais do Cliente e *tokens* de atualização. A escolha do tipo de concessão a ser adotada pode ser elegida de acordo com a Figura 6.

Figura 6 – Escolhendo o fluxo de concessão, adaptado de (RICHER; SANSO; GLAZER, 2017)

Uma pesquisa abrangente sobre *OAuth* pode ser encontrada no trabalho de (GOUD; REDDY, 2013) e informações de como cada fluxo do *OAuth 2* funciona podem ser consultadas em ¹. Embora todo o processo esteja baseado na especificação do *OAuth 2.0*, as concessões de autorização podem ser tratadas com vários formatos de *tokens*, tais como: **Tokens de Suporte**, **Tokens JSON** e **Tokens de Prova de Posse**. Estas representações estão definidas pelo grupo de trabalho *Authentication and Authorization for Restrastrash Entries (ACE)* da *Internet Engineering Task Force (IETF)*. Dentre os formatos de *tokens* destacados, os *tokens JSON* têm sido escolhidos por muitos pesquisadores, provavelmente devido ao seu tamanho reduzido.

O *IETF ACE WG* propôs uma arquitetura *Delegated CoAP Authentication and Authorization Framework (DCAF)* (GERDES; BERGMANN; BORMANN, 2014) que apresenta servidores de autorização que são usados para autenticação e autorização, sendo esta arquitetura centrada principalmente em ambientes restritos. Este *framework* têm crescido bastante e muitos projetos de pesquisa estão em andamento para implementar *OAuth* com protocolos *IoT*, como (FREMANTLE *et al.*, 2014), (CIRANI *et al.*, 2015), (HERNANDEZ-RAMOS *et al.*, 2015), (SEITZ *et al.*, 2016), (HEMDI; DETERS, 2016), (KINIKAR; TERDAL, 2016), (TRNKA; CERNY, 2016), (BELTRÁN; CALVO; GONZÁLEZ, 2017), (SCIANCELEPORE *et al.*, 2017), (FERNÁNDEZ *et al.*, 2017) e (SCIANCELEPORE *et al.*, 2017).

¹ <https://oauth.net/2/>

2.6.3 JSON Web Tokens

JSON Web Token (JWT) é um padrão aberto (*RFC 7519*)², que define um formato compacto de representação de reivindicações destinado para ambientes com espaço limitado. O padrão possibilita de uma maneira simples e autônoma, transmitir de forma segura informações entre as partes por um objeto *JSON*. O conceito de compacto é devido ao seu tamanho menor que possibilita ser enviados por meio de uma *URL Uniform Resource Locator (URL)*, pelo parâmetro *POST* ou mesmo dentro de um cabeçalho *HTTP (Hypertext Transfer Protocol (HTTP))*. Além disso, o seu tamanho reduzido torna mais rápida a transmissão. O conceito de autônomo refere-se a dizer que a carga útil, corpo ou *payload* contém todas as informações necessárias sobre o usuário evitando a necessidade de consultar o banco de dados várias vezes. Os *JWT's* podem ser assinados usando uma chave com o algoritmo *HMAC (Hash-based Message Authentication Code (HMAC))* ou um par de chaves públicas/privada usando *RSA* por exemplo. Estas características fazem dos *JSON Web Token* excelentes para realização de autenticação e intercâmbio de informações (JONES; BRADLEY; SAKIMURA, 2015).

A estrutura do *JSON Web Token* consiste em três partes separadas por (*.*), que são: **cabeçalho, carga útil e assinatura**. O cabeçalho normalmente consiste em duas partes: o tipo de *token*, que é *JWT* e o algoritmo utilizado, como *HMAC SHA256* ou outro. Logo, esta informação se torna confiável devido a possibilidade de assinatura digital provido pelo uso de criptografia. O *Payload*, é a parte que contém as reivindicações, as quais, são declarações sobre uma entidade (normalmente, o usuário) e metadados adicionais. Há três classificações de reivindicações: reivindicações registradas, públicas e privadas. As reivindicações registradas são predefinidas mas não são obrigatórias. No entanto, são recomendadas para fornecer um conjunto de reivindicações úteis e interoperáveis. Alguns exemplos são: *iss* (emissor), *exp* (tempo de expiração), *sub* (assunto), *aud* (audiência) e outras (JONES; BRADLEY; SAKIMURA, 2015).

As reivindicações públicas são definidas à vontade por aqueles que usam *JWT's*. Entretanto, é importante que elas sejam definidas no *IANA JSON Web Token Registry* ou definidas como *URI* que contenha um espaço de nome que seja resistente às colisões. Por fim, as reivindicações privadas são aquelas que podem ser personalizadas, criadas para compartilhar informações entre partes que concordam em usá-las e que não sejam reivindicações registradas ou públicas. A última parte do *token* refere-se à assinatura, e é utilizada para verificar se o remetente do *JWT* é quem diz ser e para prover integridade, ou seja, garantir que a mensagem não foi adulterada ao longo do caminho. Para ser criada, deve-se codificar o cabeçalho, o corpo, um segredo, o algoritmo especificado no cabeçalho e assinar tudo. A saída são três repartições em *Base64* separadas por pontos que por sua vez pode ser passada em ambientes *HTML (HyperText Markup Language (HTML))* e serviços *HTTP*, sendo muito mais compactos quando comparados aos padrões baseados em *XML*, como o *SAML*. Um exemplo de *token JWT* pode ser visualizado pela Figura 7.

² <https://tools.ietf.org/html/rfc7519>

Figura 7 – Modelo de um *Token JSON*

Os *JSON Web Tokens* possuem alguns benefícios quando comparado aos *tokens* simples da *Web* (*Simple Web Token (SWT)*) e aos *tokens* de *SAML*. O grande diferencial dos *JWT's* está no seu tamanho. Como ele é menos detalhado que um *token XML*, quando codificado seu tamanho também é drasticamente menor fazendo-o mais compacto que o *SAML*. Quando consideramos a segurança, um *token SWT* só pode ser assinado simetricamente por um segredo compartilhado utilizando para isso o algoritmo *HMAC*. Em contrapartida, os *tokens JWT* e *SAML* possibilitam usar um par de chaves pública/privada sob a forma de um certificado *X.509* para assinatura. Contudo, assinar *XML* com *XML Digital Signature* possui maior complexidade quando comparado à simplicidade de assinar utilizando o formato *JSON*. Vale ressaltar, que os analisadores *JSON* são comuns na maior parte das linguagens de programação e isso se dá pelo fato de realizar o mapeamento diretamente para objetos. Já o *XML* por sua vez, não possui um mapeamento natural de documento para objeto. Isso contribui para que o trabalho com *JWT* seja mais simples que as afirmações *SAML*.

1) Cabeçalho JOSE

Esta primeira parte que é separada pelos períodos (.) do *JWT* é conhecida como cabeçalho *JOSE*, que significa (*JavaScript Object Signing and Encryption (JOSE)*). Este é o nome dado ao grupo de trabalho do *IETF* que trabalha na padronização da representação da estrutura *JSON*. Neste cabeçalho é definido se a mensagem será assinada ou não. Na prática, um *JWT* não existe e representa uma abstração para um *JSON Web Signature (JWS)* (JONES; BRADLEY; SAKIMURA, 2015) ou um *JSON Web Encryption (JWE)* (JONES; HILDEBRAND, 2015). Basta imaginar uma classe abstrata, onde o *JWS* e *JWE* são as implementações concretas do *JWT* quando se utiliza serialização compacta. Alguns *JWT's* podem ser não seguros e isso acontece quando no cabeçalho *JOSE* o valor do elemento **alg** está definido para "nenhum". Em outras palavras, um *JWT* não seguro é um *JWS* que não possui assinatura. Quando utilizamos um *JWT*

sem assinatura, espera-se que o transporte subjacente forneça uma garantia da integridade e confidencialidade do *token*, como por exemplo utilizando *TLS* (*Transport Layer Security (TLS)*).

2) Serialização de Dados

Uma mensagem assinada ou codificada pode ser serializada³ de duas maneiras, seguindo a especificação *JWS* ou *JWE*. De forma geral, elas podem ser serialização compacta *JWS/JWE* ou serialização *JSON JWS/JWE*. Cada modelo de *token*, seja ele de assinatura *JWS* ou encriptação *JWE* utilizará um destes dois modelos de serialização. No entanto, somente será definido como instância de um *JWT* se a serialização utilizada for compacta. A serialização compacta em um *token JWS* representa um *token JWT* assinado como uma *string* compacta de uma *URL* segura. Esta *string* compacta possui três elementos principais separados por períodos (.): O cabeçalho (*header*) *JOSE*, a carga útil (*Payload*) e a assinatura *JWS*. Quando o modelo de serialização compacta é utilizado na carga útil *JSON*, que por sua vez também pode ser *XML* ou outro formato, teremos apenas uma única assinatura que é calculada sobre o cabeçalho *JOSE* completo e sobre a carga útil *JWS* (JONES; HILDEBRAND, 2015), (JONES; BRADLEY; SAKIMURA, 2015), (JONES, 2015a), (JONES, 2015b).

Em contrapartida, a serialização *JWS JSON* pode produzir assinaturas múltiplas sobre a carga útil *JWS*, juntamente com vários cabeçalhos *JOSE*. O formulário serializado final com a serialização *JWS JSON* envolve a carga útil assinada em um objeto *JSON*, com todos os metadados relacionados. Este objeto *JSON* inclui dois elementos de níveis superior: carga útil e assinatura (que é uma matriz *JSON*) e três sub-elementos sob cada entrada da matriz de assinaturas: protegido, cabeçalho e assinatura. O processo de serialização em *JWE* segue o mesmo princípio. No *JWS* a mensagem a ser criptografada usando o padrão *JWE* não precisa possuir uma carga útil *JSON* exclusiva, ou seja, ela pode ter qualquer outro conteúdo. Tanto o processo de serialização compacta ou *JSON* são similares quando comparadas ao modelo de serialização *JWS*. No entanto, novos campos surgem, e a notação de cada uma pode ser consultada em (JONES, 2015a) e (JONES, 2015b).

3) Algoritmos JSON Web Tokens

Quando se utilizam *tokens JSON* é possível não aplicar criptografia nos dados, deixando assim à cargo de outras camadas da aplicação prover segurança. Porém, é recomendado que se aplique alguma criptografia para proteção dos dados logo após a definição das reivindicações do *payload*. O algoritmo selecionado para aplicar a criptografia dependerá de qual tipo de *token JSON* que será utilizado. A Tabela 1 apresenta a relação de algoritmos permitidos para se usar os *tokens* para prover **Integridade** utilizando o processo de assinatura. Na Tabela 2 são apresentados os algoritmos permitidos para prover a **Confidencialidade**. Em ambas as tabelas, os algoritmos

³ **Serialização:** Em Ciência da Computação, no contexto de armazenamento e transmissão de dados, a serialização é referida ao processo de salvar um objeto em um meio de armazenamento como por exemplo um *buffer* de memória, ou ainda transmitir-lo pela rede em um formato de representação seja em forma binária, *JSON*, *XML* e etc.

são definidos pelo *IETF* e suas especificações estão disponíveis pela: RFC 7515 ⁴, RFC 7516 ⁵, RFC 7517 ⁶, RFC 7518 ⁷ e RFC 7519 ⁸.

Tabela 1 – Algoritmos *JWS* para Assinatura (JONES, 2015a)

"alg"Param Value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended+
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

Tabela 2 – Algoritmos *JWE* para Encriptação (JONES, 2015a)

"alg"Param Value	Key Management Algorithm	More Header Param	Implementation Requirements
RSA1_5	RSASSA-PKCS1-v1_5	(none)	Recommended-
RSA-OAEP	RSASSA OAEP using default parameters	(none)	Recommended+
RSA-OAEP-256	RSASSA OAEP using SHA-256 and MGF1 with SHA-256	(none)	Optional
A128KW	AES Key Wrap with default initial value using 128-bit key	(none)	Recommended
A192KW	AES Key Wrap with default initial value using 192-bit key	(none)	Optional
A256KW	AES Key Wrap with default initial value using 256-bit key	(none)	Recommended
dir	Direct use of a shared symmetric key as the CEK	(none)	Recommended
ECDH-ES	Elliptic Curve Diffie-Hellman Ephemeral Static key agreement using Concat KDF	"epk", "apu", "apv"	Recommended+
ECDH-ES+A128KW	ECDH-ES using Concat KDF and CEK wrapped with "A128KW"	"epk", "apu", "apv"	Recommended
ECDH-ES+A192KW	ECDH-ES using Concat KDF and CEK wrapped with "A192KW"	"epk", "apu", "apv"	Optional
ECDH-ES+A256KW	ECDH-ES using Concat KDF and CEK wrapped with "A256KW"	"epk", "apu", "apv"	Recommended
A128GCMKW	Key wrapping with AES GCM using 128-bit key	"iv", "tag"	Optional
A192GCMKW	Key wrapping with AES GCM using 192-bit key	"iv", "tag"	Optional
A256GCMKW	Key wrapping with AES GCM using 256-bit key	"iv", "tag"	Optional
PBES2-HS256+A128KW	PBES2 with HMAC SHA-256 and "A128KW" wrapping	"p2s", "p2c"	Optional
PBES2-HS384+A192KW	PBES2 with HMAC SHA-384 and "A192KW" wrapping	"p2s", "p2c"	Optional
PBES2-HS512+A256KW	PBES2 with HMAC SHA-512 and "A256KW" wrapping	"p2s", "p2c"	Optional

2.7 Avaliação de Desempenho de Sistemas Computacionais

Avaliação de desempenho é um processo aplicado em um sistema computacional com o intuito de avaliar resultados de experimentos. Um dos grandes desafios que há na computação, refere-se em manter um equilíbrio aceitável entre confiabilidade e desempenho pois são de forma geral considerados antagônicos, ou seja, a medida que o desempenho de um sistema cresce a sua confiabilidade se torna menor. Muitos projetistas, desenvolvedores de *software*, administradores

⁴ **JWS**: <https://tools.ietf.org/html/rfc7515>

⁵ **JWE**: <https://tools.ietf.org/html/rfc7516>

⁶ **JWK**: <https://tools.ietf.org/html/rfc7517>

⁷ **JWA**: <https://tools.ietf.org/html/rfc7518>

⁸ **JWT**: <https://tools.ietf.org/html/rfc7519>

de rede ou mesmo usuários de computadores se interessam pela avaliação de desempenho. Isso ocorre já que o objetivo primordial deste processo por meio de um planejamento de experimento é obter resultados que possibilitam atingir ou prover maior desempenho possível com o menor custo. Com os resultados obtidos destas técnicas, é possível determinar a melhor configuração de um determinado cenário de *software* e/ou *hardware* associados. O que permite, aferir qual deles ou qual combinação é mais eficiente na solução de um problema e quão essa solução é melhor ou pior que às demais opções existentes.

Deste modo, o processo de avaliação de desempenho se mostra importante para a avaliação de um sistema computacional ou mesmo para uma aplicação de *software* construída a partir de uma infraestrutura de computação. Ela pode ser aplicada inclusive para avaliar sistemas restritos que normalmente possuem baixo poder computacional como é o caso da maioria dos dispositivos *IoT*. Recursos como memória **RAM**, consumo de energia e largura de banda podem ser analisados para escolher um melhor algoritmo de acordo com o cenário disponível. De acordo com Jain (1990) os passos necessários para se realizar uma avaliação de desempenho em um sistema computacional são:

1. **Definir os objetivos e o sistema:** O passo inicial de uma avaliação de desempenho é definir os objetivos do estudo e a arquitetura do sistema, ou seja, em que constitui o sistema e quais são suas fronteiras;
2. **Listar serviços e saídas:** Quando um usuário realiza uma ação no sistema, ele pode apresentar uma série de saídas possíveis. Parte destas saídas podem ser previsíveis e desejáveis, no entanto, outras não. Portanto, é importante estabelecer quais variáveis de resposta serão monitoradas;
3. **Selecionar métricas:** As métricas são critérios existentes utilizados para comparar o desempenho do sistema. Em geral, elas são relacionadas com velocidade, precisão e disponibilidade dos serviços;
4. **Listar parâmetros:** Refere-se aos parâmetros que afetam o desempenho;
5. **Selecionar fatores para estudo:** A lista de parâmetros pode ser dividida em duas partes: aqueles que sofrem variações durante uma avaliação que são denominados fatores e aqueles que permanecem constantes em uma avaliação que são definidos como níveis;
6. **Selecionar a técnica de avaliação:** Estas técnicas variam conforme os recursos disponíveis, o nível desejado de precisão e o tempo gasto para resolver o problema em questão;
7. **Seleção da carga de trabalho:** A carga de trabalho consiste em um conjunto de requisições de serviços realizadas em um sistema, que visa analisar seu comportamento com diferentes conjuntos de requisições;

8. **Planejamento de experimentos:** É uma sequência de experimentos com combinações variadas que retornem o maior número de informações com o mínimo de execuções possíveis;
9. **Análise e interpretação dos dados:** É uma das fases mais importantes de todo o processo de avaliação, pois é necessário conhecer bem o sistema avaliado para que se possa identificar possíveis discrepâncias nos testes, de modo, que as conclusões não sejam comprometidas;
10. **Apresentação de resultados:** Esta é a fase final de todo o processo de avaliação em que os resultados são apresentados normalmente por meio de gráficos e tabelas para que seja de fácil entendimento.

Uma carga de trabalho pode ser real ou sintética. A carga real é aquela observada em um sistema sendo usado para operações normais. Este modelo de carga não pode ser replicado, logo, seu uso não é recomendado para casos de testes. Uma carga de trabalho sintética possui características semelhantes à carga real, porém, ela pode ser replicada diversas vezes de forma controlada, por isso é desenvolvida e utilizadas para estudos (JAIN, 1990).

Jain (1990) explica que o planejamento de experimentos possui como objetivo, obter o máximo de informação com o mínimo de experimentos possíveis. Segundo o mesmo autor, a análise adequada de experimentos também ajuda a separar os efeitos de vários fatores que podem afetar o desempenho. Além de determinar se um fator terá um efeito significativo no experimento ou se a diferença observada é simplesmente devido a variações aleatórias causadas por erros de medição e parâmetros que não foram controlados. Jain (1990) discorre existir alguns termos que fazem parte do processo de análise de experimentos, tais como:

1. **Variável de resposta:** Refere-se a saída de um experimento e geralmente é considerado a métrica utilizada na avaliação do sistema;
2. **Fatores:** Variável que afeta as variáveis de resposta e que podem assumir diversas alternativas. Podem ser classificados como Fatores Primários quando causam um grande impacto em uma variável de resposta e que devem ser considerados e Fatores Secundários, quando o impacto na variável de resposta não é significativo ou não se têm interesse em quantificar;
3. **Níveis:** Os valores que um determinado fator pode assumir;
4. **Interação:** Indica a dependência entre os fatores avaliados;
5. **Replicação:** Repetição de todo ou de parte de um experimento;
6. **Projeto:** Determina o número de experimentos a serem considerados, incluindo o número de fatores e níveis, a combinação entre os níveis e o número de replicações para cada experimento.

Há vários modelos de planejamentos de experimentos, no entanto, os mais utilizados são: fatorial simples, fatorial completo e fatorial parcial. Suas características são definidas a seguir:

1. **Fatorial Simples:** É o modelo mais fácil de se aplicar, contudo, não é estatisticamente eficiente, pois, não é possível analisar a influência em todos os fatores. No fatorial simples, começa-se com uma configuração inicial típica e varia-se um fator a cada vez. O intuito é perceber como esse fator afeta o desempenho do sistema.
2. **Fatorial Completo:** Este modelo consiste em utilizar todas as combinações possíveis em todos os níveis de todos os fatores. A grande vantagem está na possibilidade de examinar todas as combinações possíveis, bem como, as cargas de trabalho, além de permitir analisar as influências entre os fatores. Sua desvantagem refere-se em alguns casos ao elevado número de experimentos.
3. **Fatorial Parcial:** Consiste em utilizar uma fração do planejamento fatorial completo, em casos que a quantidade de experimentos necessários é muito grande.

Algumas medidas estatísticas são necessárias para avaliar os resultados coletados do comportamento do sistema. Utilizar somente a média pode não ser suficiente em determinadas situações. Isso acontece devido à média não apresentar nenhuma informação acerca do espalhamento dos dados, isto é, o quão distante as amostras estão do valor médio encontrado. A fim, de tratar estas questões utiliza-se o desvio padrão e a variância, que são parâmetros utilizados para medir a dispersão de um conjunto de dados em relação à média.

Para encontrar uma estimativa perfeita para a média seria necessário um número infinito de amostras. No entanto, isso não é possível. Assim, adota-se a obtenção de limites probabilísticos denominados como intervalos de confiança, de modo que a média exata esteja entre este intervalo com uma certa probabilidade de acerto. De modo geral, o intervalo de confiança é explicitado em percentuais próximos a 100%, 90% ou 95% por exemplo (SILVA *et al.*, 2016).

2.7.1 Técnicas de Avaliação de Desempenho

Para avaliação de desempenho de um sistema, o avaliador deve coletar informações referentes aos parâmetros significativos à avaliação. Para coleta dessas informações surgem os métodos de avaliação de desempenho. Muitos métodos são aplicados conforme a natureza da avaliação e recursos disponíveis. De forma geral, as técnicas existentes se classificam em: técnicas de aferição e técnicas de modelagem (SANTANA *et al.*, 1994). A Figura 8 apresenta as classificações existentes.

1) Técnicas de Aferição

Esta classe de técnicas exige que o sistema a ser avaliado já exista, esteja disponível ou em fase final de desenvolvimento. O estudo por meio de aferição pode ser feito através

Figura 8 – Técnicas de Avaliação de Desempenho (Adaptado de (SANTANA *et al.*, 1994))

da experimentação direta. Isso implica em utilizar o sistema e coletar dados a seu respeito diretamente. Em relação a modelagem, as técnicas de aferição apresentam vantagens observadas por meio de resultados mais precisos. No entanto, a desvantagem está na necessidade de se ter um sistema disponível. Independente da técnica utilizada é necessário garantir que a própria técnica não interfira no comportamento do sistema e comprometa os resultados obtidos. Algumas das principais técnicas de aferição são: protótipos, *benchmarks* e coleta de dados (ORLANDI, 1995).

A construção de protótipos representa uma simplificação de um sistema computacional, desde que se mantenha a mesma funcionalidade. Os protótipos consideram apenas características relevantes para a análise do sistema, podendo ser considerado um meio termo entre o sistema final e as expectativas que se tem dele quando ele ainda não existe. Esta técnica apresenta um custo elevado em relação as demais técnicas de aferição, todavia, é menor se comparada a construção do sistema real. A grande dificuldade desta técnica, é determinar quais características são essenciais ao sistema (SANTANA *et al.*, 1994).

Benchmarks são programas usados para o teste de *software*, *hardware* ou sistemas computacionais completos. Em suma, qualquer aspecto de desempenho de um sistema computacional que interesse aos avaliadores, pode ser objeto de medição por meio de um *benchmarks*. É importante tomar alguns cuidados ao se utilizar esta técnica, primeiro, por se tratar de uma técnica de aferição, deve-se verificar se o próprio *benchmark* não influenciará no comportamento do sistema (SANTANA *et al.*, 1994).

A coleta de dados é a técnica de aferição mais precisa e utiliza de ferramentas chamadas monitores para observar as atividades de um sistema, coletando as características importantes para a análise do sistema. Os resultados obtidos por meio dessa técnica podem ser usados para comparações com os resultados obtidos a partir de modelos do mesmo sistema. A coleta de

dados, muitas vezes referida como monitoração pode ser realizada por meio de dois recursos: monitores de *hardware* e monitores de *software*.

Os monitores de *hardware*, são *hardwares* específicos empregados para coletar e analisar alguns dados referente ao objeto de estudo (ORLANDI, 1995). Estes monitores não podem ser intrusivos, ou seja, devem limitar-se a obter os sinais e os dados sem que haja alteração, mantendo fidelidade aos valores originais. Por outro lado, os monitores de *software* são utilizados em casos que se deseja observar características específicas do *software*, como por exemplo, verificar se existe ou não uma fila de espera associada a algum recurso do sistema.

2) Técnicas de Modelagem

Não é viável para as indústrias, centros de pesquisas ou qualquer outra instituição, ter que construir um sistema para só então avaliar seu desempenho. O processo de verificação e teste de um sistema, antes mesmo de sua existência representa uma redução enorme de gastos para os desenvolvedores. Em situações como esta, em que se deseja avaliar um sistema antes de sua existência ou que sua experimentação prática seja bastante complexa é que se propõe o uso da técnica de modelagem. Um modelo nada mais é que a representação de um sistema com o objetivo de evidenciar as características mais importantes. Há várias técnicas para modelar um sistema computacional, contudo, três técnicas têm sido amplamente utilizadas tais como: Redes de Filas, Redes de Petri e *Statecharts*. Cada técnica possui vantagens e desvantagens conforme o domínio de aplicação considerado (SANTANA *et al.*, 1994).

A técnica de Redes de Filas foi criada baseada na teoria de filas (um ramo das probabilidades). Ela é utilizada em situações em que o surgimento de filas em um sistema computacional é inevitável. Como por exemplo o surgimento de "disputas" por algum recurso como acesso à disco, utilização da CPU e acesso à rede. Uma rede de filas consiste de duas entidades: centros de serviços e usuários, de forma geral os centros de serviços são entidades que oferecem serviços dos quais os usuários usufruem (JUNIOR; PIMENTA, 2004). Já as Redes de Petri são técnicas que permitem a representação de sistemas utilizando uma forte base matemática. Essa técnica possui a particularidade de permitir modelar sistemas paralelos, concorrentes, assíncronos e não determinísticos. *Statecharts* são considerados uma extensão das máquinas de estados finito, ou seja, uma linguagem visual bem conhecida para modelar o comportamento executável de sistemas complexos reativos baseados em eventos. A definição de *statecharts* é fundamental em conjunto de estados, eventos e condições primitivas, transições e variáveis. A ideia principal é seu uso na representação de sistemas complexos onde os diagramas de estado apresentam deficiências (JUNIOR; PIMENTA, 2004).

Depois que se define a técnica de análise para a construção do modelo do sistema em estudo, o próximo passo é escolher a solução a ser dada. Nesta fase, há basicamente duas técnicas para a resolução de modelos: Solução Analítica e Simulação. Cada solução também possui suas vantagens e desvantagens conforme o tipo de modelo a ser resolvido (SANTANA *et al.*, 1994). A solução analítica é o método mais rápido, mas nem sempre pode ser aplicável, pois, são necessá-

rias simplificações no modelo para que este possa ser resolvido. Simplificações, de forma geral, são restrições impostas pela solução analítica, que geralmente não correspondem ao verificado em sistemas reais (JUNIOR; PIMENTA, 2004). Por outro lado, a possibilidade de simular o comportamento de um sistema pode ser necessária em muitas áreas do conhecimento. Algumas das vantagens de se aplicar simulação refere-se ao fato de executar experimentos repetidas vezes para que os resultados coletados atendam aos requisitos estipulados no planejamento de experimentos e ajustar os gargalos do sistema antes da implantação entre outros (CENTURION, 2015).

2.8 Considerações Finais

Este Capítulo apresentou o referencial teórico que serve de base para este projeto de mestrado. Foram abordados os conceitos de Internet das Coisas e Computação em Nuvem. Posteriormente, foram apresentados os principais desafios de *IoT*, destacando aqueles relacionados à segurança. Foram discutidas as principais soluções envolvendo controle de acesso, com destaque para o modelo baseado em capacidades e o protocolo *OAuth* que utilizam *tokens* para prover autenticação e autorização. Em seguida, foram considerados os conceitos de *tokens JSON* e uma tabela com os seus algoritmos para assinatura e encriptação de dados. Por fim, apresentamos uma base para avaliação de desempenho e planejamento de experimentos.

TRABALHOS RELACIONADOS

3.1 Considerações Iniciais

Este Capítulo tem por objetivo apresentar o mapeamento sistemático realizado para o levantamento dos trabalhos relacionados, o qual são apresentados na Sessão 1. Na Sessão 2, apresentamos uma análise crítica dos principais trabalhos selecionados com um resumo de cada trabalho identificando as lacunas nas soluções propostas. Na Sessão 3 apresentamos a identificação do problema e o domínio da aplicação e por fim, concluímos com as considerações finais atribuídas a este Capítulo.

3.2 Mapeamento Sistemático

Segundo (KITCHENHAM *et al.*, 2009a) Revisão Sistemática da Literatura (Systematic Literature Review - SLR - em inglês) é definido como: “um estudo secundário que usa uma metodologia bem definida para identificar, analisar e interpretar todas as evidências relacionadas a uma questão de pesquisa de forma que seja repetível e livre de qualquer viés”. No entanto, é comum encontrar na literatura o conceito de Mapeamento Sistemático, que segundo os mesmos autores, possui o objetivo de aumentar a qualidade de pesquisa em relação à revisão de literatura tradicional, também conhecida como pesquisa *Ad-Hoc*.

De forma geral, podemos definir um mapeamento sistemático como um estudo secundário realizado, cujo objetivo é agregar evidências disponíveis sobre um determinado tema. Assim, ele é constituído por uma metodologia rigorosa que busca fornecer uma visão geral de uma área de pesquisa, visando identificar a quantidade, qualidade, tipo, diminuição de erros e outras variáveis desejadas para que se identifique os trabalhos disponíveis mais relevantes de uma área.

(PETERSEN *et al.*, 2008) argumentam que a técnica de mapeamento sistemático é recomendada quando existem falta de estudos primários relevantes e de alta qualidade. Normal-

mente o mapeamento sistemático (SM) é recomendado no lugar de uma SLR, quando o tópico é muito abrangente ou ainda quando existem poucas evidências a serem coletadas. Usamos uma combinação de estratégias de busca para garantir a cobertura completa da literatura relevante, combinando pesquisas automatizadas de banco de dados com um processo de *snowballing* (WOHLIN, 2014), isso nos dá uma melhor cobertura da área mapeada.

3.2.1 Método de Pesquisa

Seguimos um método de levantamento de trabalhos que se assemelha a um mapeamento sistemático que objetiva identificar quais são os formatos de *tokens* mais amplamente utilizados nas aplicações de controle de acesso para Internet das Coisas, quais algoritmos de criptografia e protocolos estão sendo aplicados para a proteção e sigilo dos dados contidos nos *tokens*. Em muitas situações, a capacidade deve ser protegida para que não conceda acesso às pessoas não autorizadas e a maneira usual de proteger a capacidade é por meio de criptografia simétrica e assimétrica. No entanto, em aplicações envolvendo Internet das Coisas é importante que o custo computacional gerado pelo algoritmo criptográfico e pelo *token* seja o menor possível para que a durabilidade dos recursos seja maior. Com tais objetivos fixados, definimos três questões de pesquisa:

1. **Questão de Pesquisa 1:** Quais são os formatos de *tokens* utilizados para o controle de acesso em Internet das Coisas?
2. **Questão de Pesquisa 2:** Quais protocolos e *frameworks* utilizados para o controle de acesso baseado em capacidades?
3. **Questão de Pesquisa 3:** Quais são os algoritmos utilizados para proteção da capacidade?

3.2.2 Bases de pesquisa

Existem várias bases de dados bibliográficas que são usadas em Ciência da Computação (KITCHENHAM *et al.*, 2009b) (KITCHENHAM *et al.*, 2010), como pode ser visto na Tabela 3. As características mais úteis para este levantamento foram a habilidade de usar pesquisas especializadas com uma variedade de operações booleanas que concentre a pesquisa nos campos de Título, Resumo e Palavras-chave (TAK), que retornam resultados mais relevantes em comparação à pesquisa de todos os campos. Com base nestes requisitos, utilizamos as bases *IEEE Digital Library*, *Science@Direct*, *Scopus*, *ISI Web of Science* e *ACM Digital Library* para realização do nosso levantamento de trabalhos da literatura.

Para definição da *string* de busca, utilizamos o operador lógico “**OR**” para selecionar de forma alternada as palavras chaves e seus sinônimos, e o operador lógico “**AND**” para selecionar os termos para população, intervenção e resultados. Com base na relação obtida, definimos uma *string* de busca para seleção dos trabalhos. Em cada base de pesquisa, houve a necessidade de

Tabela 3 – Bases de Pesquisa

Fonte	URL
IEE Digital Library	https://ieeexplore.ieee.org
ACM Digital Library	https://dl.acm.org/
Science@Direct	https://www.sciencedirect.com/
ISI Web of Science	http://apps.webofknowledge.com
Scopus	https://www.scopus.com

fazer uma adaptação da *string* de busca, de forma que se adaptasse ao modelo de seleção de cada base. Um exemplo da *string* adotada é apresentado pela Figura 9 considerando a base da *Scopus*.

((*"Internet of Things"* OR *"IoT"* OR *"Web of Things"* OR *"Cloud of Things"* OR *"Internet of Services"* OR *"Industry 4.0"*) AND (*"Access Controls"* OR *Cryptography*) AND (*"Capability Based Access Control"* OR *"CAPBAC"* OR *"Tokens"*))

Figura 9 – String de Busca formatada na *Scopus*

3.2.3 Critérios de Inclusão e Exclusão

Para a seleção e avaliação dos resultados retornados, definimos critérios de inclusão (IC) e exclusão (EC). Estes critérios são importantes por apoiar a seleção dos estudos apropriados (primários), bem como, por reduzir o número de trabalhos na literatura cinza¹ que retornam das *engines* de busca. Se um determinado trabalho é classificado em um ou mais *Critério de Inclusão (IC)*, o mesmo será incluído como estudo primário. No entanto, se o trabalho possui classificação de *Critério de Exclusão (EC)*, este será excluído. Portanto, esta é uma das principais atividades durante o planejamento do levantamentos dos trabalhos. Aplicamos os seguintes critérios de inclusão e exclusão neste trabalho:

- **IC 1.** Estudos primários relacionados aos tópicos de pesquisa;
- **IC 2.** Estudos primários que possuam as palavras reservadas da *string* de busca no título, resumo ou palavra-chave;
- **IC 3.** Estudos primários publicados em periódicos e conferências;
- **IC 4.** Estudos primários publicados entre 2010 e 01/02/2020.
- **EC 1.** Estudos classificados como literatura cinza;
- **EC 2.** Estudos não focados nas áreas de Ciência da Computação e Engenharia;

¹ **Literatura cinza:** Informação ou resultados de investigação produzida pelas organizações, fora de canais de publicação e distribuições comerciais ou acadêmicas como, por exemplo, relatórios de pesquisa, documentos do governo e outros

- **EC 3.** Estudos que não estejam acessíveis livremente;
- **EC 4.** Estudos que não estejam escritos em Inglês.

3.2.4 Processo de Seleção de Leitura

Depois de recuperar os resultados usando o procedimento descrito na subseção anterior, aplicamos critérios de seleção para filtrar os trabalhos mais relevantes seguindo o modelo de leitura proposto por (KESHAV, 2007) em três passos. Os resultados removidos em cada estágio é ilustrado pela Figura 10.

Figura 10 – Método de Seleção de trabalhos (Adaptado de (WHITE; NALLUR; CLARKE, 2017))

1. **Remoção de duplicados:** usando a ferramenta online *parsif.al*² conseguimos localizar e remover automaticamente trabalhos duplicados com base nos nomes dos autores, ano e título do trabalho;
2. **Seleção por título:** estágio utilizado para remover rapidamente os artigos retornados pelos resultados da pesquisa cujo escopo não se relaciona com as palavras chaves utilizadas;
3. **Seleção por resumo:** esta etapa remove trabalhos que não apresentam uma abordagem de qualidade como contribuição do trabalho considerando o resumo;
4. **Seleção por trabalho completo:** Esta etapa foi utilizada para remoção de trabalhos que não apresentam uma abordagem de qualidade como uma das contribuições do trabalho.

3.2.5 Extração de dados

Os trabalhos selecionados no filtro final foram submetidos ao processo de extração de dados, etapa esta que consiste em obter as informações necessárias que nos permitiu responder as questões de pesquisa. Para que a extração fosse realizada de maneira mais objetiva possível, definimos um formulário de extração pelo *Google Forms*³ para registrar as informações necessárias relativas a cada trabalho, reduzindo a oportunidade de inclusão de viés do pesquisador. Neste sentido, os campos descritos na Tabela 4 apresenta como o nosso formulário de extração de dados foi definido. Destacamos os três campos principais (Algoritmo utilizado, Protocolo/Framework e formato de *Token*), pois estes respondem diretamente as questões de pesquisa propostas.

² <https://parsif.al/>

³ <https://goo.gl/forms/YMIQOJIyPJZ3HOk72>

Tabela 4 – Dados do formulário de extração

Campo	Descrição
Título	Qual o título do trabalho
Ano	Ano de publicação
DOI	Registro DOI
Evento	Evento publicado
País de publicação	Onde foi publicado
Autores	Quais são os autores
Conferências/Revista	Trabalho de conferência ou Revista
Algoritmo utilizado	Quais algoritmos utilizados
Protocolo/Framework	Utiliza? quais?
Formato do <i>token</i>	<i>JSON, XML, CSV</i> e etc
Avaliação de desempenho	Possui uma avaliação?
Ambiente de aplicação	Qual o cenário
Limitações	Limitações do trabalho

No total 34 trabalhos analisados, apresentamos uma análise crítica dos 10 trabalhos que mais se aproximam e que serviram de base para o desenvolvimento desta dissertação. As características principais destes trabalhos estão evidenciadas pela Tabela 5.

Tabela 5 – Principais trabalhos selecionados

Trabalho	Token	Algoritmo	Avaliação de Desempenho	Ambiente	OBS
\Skarmeta, Hernandez-Ramos e Moreno (2014)	JSON	ECDSA	SIM	-	CoAP
\Emerson <i>et al.</i> (2015)	-	-	NÃO	-	OAuth
\Solapurkar (2016)	JSON	RSA	SIM	Saude	OAuth
\Trnka e Cerny (2016)	JSON	-	NÃO	-	OAuth, OPEN ID CONNECT
\Park <i>et al.</i> (2016)	-	-	NÃO	Saude	-
\Beltrán, Calvo e González (2017)	JSON	-	SIM	Saude	PUF/Token
\Fernández <i>et al.</i> (2017)	-	-	NÃO	-	OAuth
\Sciancalepore <i>et al.</i> (2017)	JSON	AES, SHA	SIM	-	OAuth
\Jung e Jung (2017)	JSON	HMAC	SIM	Saude	OAuth
\Lee, Huang e Yang (2017)	JSON	Hash	SIM	Saude	OAuth
\Martins Júlio Cezar Estrella (2020)	JSON	RSA e HMAC	SIM	Saude	OAuth

3.3 Análise Crítica dos Trabalhos Relacionados

Skarmeta, Hernandez-Ramos e Moreno (2014) propõem um mecanismo de controle de acesso baseado em capacidade (*CAPBAC*), utilizando criptografia de chave pública com o protocolo CoAP (*Constrained Application Protocol (CoAP)*). A solução é baseada no design de um *token* considerado leve com uma implementação otimizada do algoritmo de assinatura digital de Curva Elíptica (*ECDSA*) dentro do objeto inteligente. Essa implementação otimizada garante autenticidade, integridade e não repúdio de ponta a ponta sem a intervenção de qualquer entidade intermediária. Para implementar sua abordagem de controle de acesso baseada em capacidades distribuídas, os autores escolheram o formato *JSON Web Token* para representar o *token* de capacidade devido à sua adequação em ambientes restritos, como os sugeridos pelos cenários de *IoT*. A estrutura de cada campo do *token* é apresentada juntamente com uma avaliação de desempenho com 100 testes de acordo com o cenário. O tempo necessário para a avaliação do cenário foi de 480,96ms, tornando-o viável para um ambiente real. Embora os autores utilizem

o algoritmo de Curva Elíptica (*ECDSA*) (LAMBA; SHARMA, 2013), eles não detalham ou justificam a razão da escolha deste algoritmo com profundidade. Uma variação de algoritmo, poderia aumentar ou diminuir o desempenho do mecanismo bem como a segurança, mas para esta análise se necessita de uma avaliação mais completa. Cada cenário pode apresentar configurações próprias, logo, é interessante que o mecanismo possa ser flexível para alterar-se conforme o ambiente de teste. Os mesmos autores identificam ainda outras possibilidades para emitir o *token*, como por exemplo, por geolocalização ou mesmo nas relações de confiança entre usuários e objetos inteligentes envolvidos no estágio. Outro fator a ser explorado é o uso de criptografia baseada em atributos (*Attribute-Based Encryption (ABE)*) como alternativa ou em combinação com *ECC (Elliptic Curve Cryptography (ECC))* para comunicações dentro de grupos.

Emerson *et al.* (2015) utilizam o protocolo *OAuth 2.0* (HARDT, 2012) em uma rede *IoT*, introduzindo um mecanismo de autenticação que também ajuda a evitar os ataques de segurança mais comuns deste protocolo. O modo padrão de autenticação requer que a rede *IoT* armazene e gerencie todas as informações do usuário por conta própria e a autenticação é feita comparando a *ID* do usuário e a senha armazenada no banco de dados. Este esquema é vulnerável à ataques de segurança e requer um banco de dados de armazenamento para manter as informações pessoais do usuário com mais segurança, o que é uma tarefa tediosa. A abordagem proposta evita que todos os usuários dos provedores de serviços acessem a rede *IoT* e permitam que apenas os usuários autenticados acessem a rede. O processo de autenticação é feito pelo gerente de segurança. O gerente de segurança compara o *ID* do usuário obtido do provedor de serviços usando o *token* de acesso com seu banco de dados local. Somente em caso de autenticação bem-sucedida o usuário pode acessar a rede *IoT*. Uma limitação deste trabalho refere-se ao fato dos autores não apresentarem a estrutura e o processo de assinatura dos *tokens*. Caso um atacante consiga capturar um *token*, é importante garantir que ele não consiga decodificar e utilizar seu conteúdo. Para tal, a criptografia utilizada deve ser promissora segundo o dado transmitido e sua necessidade de privacidade. Informações sigilosas necessitam de uma cifragem mais robusta que implicará em um desempenho mais precário. É importante que seja feita uma avaliação de desempenho dos algoritmos de cifragem e do modelo proposto de forma a contribuir no mecanismo de construção de *tokens* mais apropriados, isso possibilita verificar quando é necessário um desempenho maior ou uma segurança maior.

(SOLAPURKAR, 2016) apresenta um esquema baseado em *OAuth 2.0* e *JSON Web Token* para garantir serviços de saúde em um cenário de nuvem *IoT*. Este esquema pressupõe um aplicativo clínico (cliente) executado em um navegador da *Web* ou em um *Smartphone*, diversos pacientes que procuram assistência médica em tempo real, vários dispositivos *IoT* para monitoramento remoto da saúde do paciente e vários bancos patrocinadores com várias contas do paciente. O aplicativo clínico e o banco on-line possui um relacionamento confiável e o servidor de autorização de banco emite o *token* de acesso somente após a validação do *JWT* que está assinado e criptografado digitalmente com o algoritmo de assinatura *RSA SHA-256*. Neste modelo, o médico clínico usa os dados analisados dos dispositivos inteligentes para fazer

uma nova receita eletrônica sempre que necessário. Isso pode ser conseguido porquê o paciente que recebe assistência médica em tempo real consegue autorizar pagamentos automáticos para cada receita eletrônica. Por utilizar o método de assinatura assimétrico, pode haver um consumo maior no tempo de resposta para aquisição do *token*. Em casos críticos, pode haver um impacto negativo para aplicação e seria interessante analisar o custo da assinatura assimétrica em relação a um modelo simétrico. Em casos em que se necessita maior velocidade, o uso da assinatura simétrica pode ser uma melhor opção.

Trnka e Cerny (2016) propõem uma solução para gerenciamento de dispositivos em Internet das Coisas baseada em uma loja central de identidade, em que cada dispositivo possui uma conta associada na loja com funções correspondentes. Para qualquer comunicação, o dispositivo recupera um *JSON Web Token* e o usa para certificar-se em todas as conexões de rede. A estrutura proposta cria um ambiente confiável e permite uma resposta rápida para eventos de segurança. A metodologia consiste em uma estrutura de identidade central na qual realiza o controle de cada dispositivo conectado a rede por meio de um identificador exclusivo. Além disso, armazena as funções internamente por meio de um controle de acesso baseado em funções e todas as máquinas e aplicativos podem usar estas funções para suas regras de autorização. A comunicação é composta por dois participantes, o primeiro chamado de provedor onde expõe serviços a outros, e um segundo denominado consumidor, que usa a funcionalidade dos provedores e inicia a comunicação. Todo o processo de comunicação, autenticação e autorização são realizadas pelo protocolo *OAuth 2.0* com *OpenID Connect* e *JSON Web Tokens*. Os autores apresentam um protótipo de aplicação, em que desenvolveram um aplicativo que reúne dados de dois sensores (temperatura e movimento) e utiliza um *Raspberry Pi* para transmitir os dados pela Internet. O aplicativo criado expõe os dados aos usuários por meio de uma interface *RESTful* com *Java EE* no *backend* e *Web Java Script* no *front-end*. Um fator interessante a ser analisado para este trabalho está no fato dos autores não realizarem uma avaliação de desempenho do modelo proposto. A estruturação dos *JSON Web Tokens* não é mencionada sendo esta a única forma de comunicação entre os dispositivos. O tamanho do *token* e o método de assinatura impacta na segurança e no desempenho da aplicação, logo, uma análise com esta perspectiva é importante principalmente para dispositivos com baixo poder computacional, bem como o monitoramento dos recursos consumidos (CPU, Memória, Disco e etc) pelo modelo adotado.

Park *et al.* (2016) criaram um mecanismo de controle de acesso que possibilita a transferência de informações de pacientes hospitalares utilizando delegação. Neste modelo, o usuário é a autoridade responsável pelo processo de autorização dos dados e não um servidor específico. O ponto forte do trabalho consiste em não permitir que membros da equipe hospitalar colem informações sem o consentimento do paciente e isso reduz o risco de vazamento de informações sensíveis. A metodologia adotada se resume em seis papéis: O proprietário do recurso (paciente), um usuário autorizado com relação de confiança com o proprietário do recurso, um estabelecimento de saúde primário que contem as informações do paciente, um estabelecimento de saúde secundário que não possui os dados do paciente, mas que poderá

precisar para uma eventual emergência futura, um servidor de armazenamento no qual armazena os *tokens/chaves* para comunicação e por fim, um provedor de serviço que realiza a autenticação. Neste modelo, um usuário é responsável por fazer a solicitação dos dados do paciente que esta armazenado em um servidor e encaminha-los até o serviço de saúde secundário de modo que somente através deste usuário autorizado as informações serão enviadas. O envio dos dados do paciente é realizado por um *token* que realiza o controle de acesso com as informações requeridas. Algumas limitações deste trabalho envolve o não detalhamento do uso de *tokens* no processo de autorização e autenticação. Os autores não apresentam como a estrutura do *token* é formada, quais algoritmos e chaves utilizados e não apresentam uma avaliação de desempenho do modelo proposto. Como são informações sensíveis, tanto a segurança destas informações como o desempenho da aplicação é importante e carecem de uma avaliação mais aprofundada.

Beltrán, Calvo e González (2017) propõem um novo mecanismo de autenticação e autorização com base na combinação de uma *Função Unclonable Física (PUB)* e dois *tokens* (um dedicado à autenticação e outro dedicado à autorização), capaz de trabalhar com *HTTP* ou *CoAP* confiando em esquemas federados e adaptados aos requisitos específicos destes tipos de ambientes. O novo mecanismo é validado e sua eficiência e segurança são avaliadas usando um estudo de caso de saúde real. Uma pesquisa detalhada a respeito de autenticação por *PUF's* pode ser encontrada em (DELVAUX *et al.*, 2015). Entre as soluções existentes, os trabalhos de Maes (2012) e Herreweghe *et al.* (2012) serviram como base para o mecanismo proposto. A autenticação e autorização por *tokens* é realizada por dois *tokens* diferentes, um *token* de identificação (em relação à identificação e autenticação) e um *token* de acesso (em relação à autorização e ao controle de acesso). Cada *token JWT* é assinado digitalmente para prover integridade e sua estrutura é apresentada. Uma análise de desempenho é realizada partindo de um estudo de caso real que envolve cuidados com a saúde. Nesta avaliação, foi projetado uma caixa de pílula inteligente baseada em *Field Programmable Gate Array (FPGA)*, que auxilia o paciente com informações de controle de medicamento e tratamento de incidentes caso o paciente esqueça de realizar o procedimento esperado. Testes diferentes foram realizados em dois cenários para validar e avaliar a solução proposta. O cenário **A**, onde o serviço é executado em um centro de dados de nuvem remota de recursos completos, e no cenário **B**, onde o serviço é executado em um servidor de borda restrita perto da casa do paciente com base no paradigma de Computação em Nuvem.

Os autores apontam que o processo de validação de *token*, quando implementado para fornecer os níveis de segurança desejados não é apenas uma tarefa que consome energia, mas, também uma tarefa demorada. Especificamente, a verificação da assinatura dos *tokens* é uma das etapas envolvidas mais complexas e o tempo gasto para a validação do *token* não foi incluída nas avaliações realizadas. Embora os autores especifiquem os campos existentes na estruturação do *token JWT*, eles não mencionam qual o algoritmo utilizado para assinatura digital e nem mesmo, qual a chave adotada. Logo, uma análise profunda dos algoritmos para validação de *tokens JSON* faz-se necessária para melhorar o desempenho do modelo proposto. Em ambientes de saúde,

onde os dados são sensíveis, o fator segurança e desempenho deve ser combinado de modo que possibilite a melhor configuração para o usuário.

Fernández *et al.* (2017) propõem um modelo de autorização baseado no protocolo *OAuth 2.0* oferecendo controle de acesso como um serviço para ambientes de Internet das Coisas. Do ponto de vista do provedor de identidade, este modelo permite gerenciar funções e permissões para uma autorização de escopo de aplicativos, para permitir cenários mais flexíveis e de fácil delegação. A arquitetura proposta consiste nas seguintes entidades: Um dispositivo *IoT*, um aplicativo (Um serviço cujos recursos estão dispostos a ser acessado pelo dispositivo) um administrador de serviço, um provedor de identidade (*Identity Provider (IdP)*) e os Pontos de Administração (*Policy Administration Point (PAP)*), Decisão (*Policy Decision Point (PDP)*) e Execução de Políticas (*Policy Enforcement Point (PEP)*), estes três últimos são os componentes que fazem a amplamente conhecida arquitetura de controle de acesso. Além disso, incluem um *Policy DB* onde as políticas são armazenadas pelo PAP e verificado pelo PDP. Os autores argumentam que as políticas de controle de acesso são definidas por meio de permissões e funções que o administrador do serviço gerencia no *PAP*. No entanto, políticas mais complexas também podem ser definidas por meio de uma linguagem de descrição de políticas como *XACML*. Porém, os autores não implementam ou validam o modelo proposto. As especificações relacionadas aos *tokens* como estruturação, algoritmos e chaves não são mencionadas, bem como a definição das políticas de controle.

Sciancalepore *et al.* (2017) apresentam uma estrutura de autenticação e autorização flexível para Internet das Coisas com base no protocolo *OAuth*, chamada de *OAuth-IoT*. *OAuth 2.0* é uma estrutura de autorização padronizada em (HARDT, 2012). Ele foi projetado para permitir que aplicativos de terceiros tenham acesso aos recursos restritos sob o controle de um usuário sem requerer o compartilhamento de suas credenciais. A arquitetura engloba quatro atores: O cliente que são aplicações de terceiros dispostos a acessar recursos protegidos, o proprietário do recurso capaz de conceder ou negar o acesso ao recurso protegido, o servidor de recursos que expõe recursos protegidos e o servidor de autorização que controla o processo de autorização. Os autores adotam o *JSON Web Token* para transportar as concessões de acesso e trabalham com o algoritmo de criptografia *AES* e o algoritmo de *hashing* do algoritmo *Hash* seguro (*SHA*), usado para o gerenciamento de *tokens* Prova de Posse (*Proof of Possession (PoP)*). As funcionalidades e os benefícios oferecidos pelo *OAuth-IoT* são pragmaticamente demonstrados por meio de um teste experimental e demonstrado com uma avaliação de desempenho muito preliminar. Os resultados relatados demonstram que as funcionalidades de segurança introduzidas pelo *OAuth-IoT* aumentam a quantidade de tempo necessário para alcançar com sucesso o recurso solicitado. Neste caso, percebemos que quanto maior a segurança a ser considerada haverá um impacto no desempenho da aplicação. Uma análise do ponto de vista de performance e o quanto a segurança impacta na performance se mostra importante, sobretudo para ambientes de Internet das Coisas.

(JUNG; JUNG, 2017) propõem uma alteração do protocolo *OAuth* no processo de autorização e autenticação em ambientes de Internet das Coisas que trabalham com dados sensíveis. O modelo tradicional possui uma entidade terceira declarada confiável como por exemplo Facebook, Google e etc que auxilia no processo de autenticação e autorização. Assim, um serviço utiliza as credenciais destas entidades para comprovar a autenticidade do usuário. Este modelo tem sido amplamente usado em vários ambientes de controle de acesso em Computação em Nuvem e vários autores também tem proposto este modelo para ambientes de *IoT* como (CIRANI *et al.*, 2015) e (EMERSON *et al.*, 2015). Em ambientes que trabalham com dados sensíveis como hospitais, pode não ser trivial encontrar uma terceira entidade que seja confiável, partindo desta premissa, os autores propõem um novo modelo onde um servidor de autorização *OAuth* utilizando *smartphones* como serviço de autorização. Deste modo, é muito mais confiável do que usar um servidor de autorização de terceiros para se autenticar, porque os usuários controlam diretamente o acesso às informações geradas pelos dispositivos. Os autores utilizam *Json Web Token* para prover assinatura do *token* juntamente com o algoritmo *HMAC SHA256*. Uma contribuição para este trabalho poderia ser a possibilidade de geração de *token* dinâmica, atribuindo níveis de segurança ao tipo de dado transmitido. Dados mais sensíveis que requerem um maior nível de privacidade, poderia utilizar uma criptografia mais pesada no processo de assinatura do *token* enquanto que dados com menor preocupação com segurança poderia utilizar uma criptografia mais leve. Para isso, é importante que haja um estudo que avalie a segurança e o desempenho dos algoritmos de *tokens JWT* para uma análise mais refinada.

(LEE; HUANG; YANG, 2017) apresentam uma arquitetura de autorização e autenticação para controle de acesso que utiliza o protocolo *OAuth*, *OpenID*, e o uso de *Json Web Token*. A aplicação envolve um cenário de serviço médico com dispositivos portáteis. Os experimentos envolvem mulheres grávidas e animais (porcos), cooperando com o Centro de Gerenciamento Pré-natal e o Centro Nacional de Animais de Laboratório. O dispositivo médico portátil é um dispositivo *wearable* que pode capturar batimentos cardíacos da mulher grávida e do animal que utilizam o dispositivo. Os dados são enviados para um *smartphone* e posteriormente retransmitidos para uma plataforma de processamento. Uma cópia dos dados é mantida pelo Centro de Gerenciamento Pre-natal e depois enviados para o centro de pesquisa médica de terceiros por uma plataforma de gerenciamento de nuvem. A comunicação é feita por meio de *tokens JWT*, onde cada *token* é emitido por um centro de verificação de terceiros e tem limites de tempo e contagem em que cada acesso aos serviços da nuvem será contado uma vez. Quando a contagem de referência do *token* excede o valor ajustado do sistema, o *token* torna-se inválido e recebe o aviso de solicitar um novo *token* emitido pelo sistema. Os autores propõem um algoritmo de *hash* para assinatura do *token* e fazem uma avaliação do modelo desenvolvido.

3.4 Identificação do Problema

O desenvolvimento de um trabalho científico não é um processo simples e rápido. Muitas vezes a busca por encontrar uma lacuna na literatura pode levar vários meses e em alguns casos até anos conforme a área de pesquisa. A Figura 11, apresenta o método utilizado até encontrar a lacuna que este trabalho explora. As cores em laranja, representam o caminho percorrido até a definição do projeto de pesquisa. Primeiramente, definimos a grande área de pesquisa como "Internet das Coisas" e investigamos por trabalhos do tipo *surveys* em busca dos desafios e problemas em aberto em *IoT*. Uma série de desafios foram encontrados como ilustra a Figura 11. Um dos principais desafios refere-se à segurança das aplicações. Portanto, enfatizamos este desafio e procuramos novos *surveys* nesta área. Surgiram um leque de novos desafios, o que nos levou a optar por levantar as soluções relacionados ao controle de acesso. A escolha desse tema, foi devido às características próprias de controle de acesso que, indiretamente necessitam trabalhar com os demais desafios levantados.

A partir de então, realizamos uma nova busca *survey* objetivando encontrar as principais soluções de controle de acesso no contexto de *IoT*. Encontramos uma variedade de modelos e protocolos de controle de acesso adaptados para o contexto de "coisas". Dentre as soluções encontradas, a literatura destaca às soluções baseadas em capacidades (CAPBAC) que fazem uso de *tokens* juntamente com o protocolo *OAuth* para aplicar o controle de acesso em *IoT*. Entre os tipos de *tokens* existentes, o formato *JSON* tem sido preferido pelos desenvolvedores devido ao seu tamanho reduzido em relação ao formato *XML* além disso, eles possibilitam utilizar criptografia para prover integridade e confidencialidade aos dados. A partir de então, aplicamos uma técnica científica para seleção de trabalhos similar a um mapeamento sistemático com a definição de uma *string* de busca com palavras chaves aplicada em diferentes motores de busca na Internet na qual detalhamos no Capítulo 3 em (**Técnica de Seleção de Trabalhos**). É importante observarmos que o uso de *tokens* pode ser aplicado independente do ambiente de processamento da informação. Contudo, o tipo de algoritmo adotado na construção dos *tokens* se difere caso o ambiente de controle de acesso seja construído na *Cloud*, *Fog* ou mais na borda da rede.

Após a análise crítica dos trabalhos selecionados, identificamos que grande parte dos autores não mencionam qual algoritmo utilizaram no processo de criação dos *tokens* ou mesmo a estrutura do *token*. As avaliações de desempenho são simples e não seguem uma metodologia de testes. Autores tais como Martin e Wang (2015) e Skarmeta, Hernandez-Ramos e Moreno (2014) utilizaram o algoritmo *ECC* em suas aplicações com *tokens JSON*. Já Jung e Jung (2017) utilizaram *HMAC* e Solapurkar (2016) utilizaram *RSA*. Porém, a escolha por estes algoritmos em detrimento dos demais existentes na *RFC 7518* não são abordados nos trabalhos. Beltrán, Calvo e González (2017) afirmam que o processo de validação dos *tokens* na aplicação que eles propuseram é uma das tarefas mais demoradas e que consomem muita energia, sendo o processo de verificação da assinatura uma das etapas mais complexas.

Figura 11 – Identificação da Lacuna de Pesquisa

Com o intuito de contribuir com a literatura, para seleção de algoritmos criptográficos mais apropriados na construção de *tokens JSON*, este trabalho busca, avaliar o impacto que o uso de *tokens JSON* podem causar no desempenho de um sistema voltado a ambientes de aplicações críticas possibilitando um *trade-off* entre desempenho e segurança. Propomos um protótipo de arquitetura para autenticação e autorização de usuários em ambientes de saúde que avalia inicialmente dois métodos de criptografia para assinatura com *JSON Web Token*. Um modelo simétrico com o algoritmo *HMAC* e um modelo assimétrico com o algoritmo *RSA*. Seguindo uma metodologia de avaliação de desempenho, buscamos identificar quais variáveis mais afetam o tempo de resposta final. Nossa avaliação, adota o processo de avaliação de desempenho definido por (JAIN, 1990) e melhor explanada no Capítulo 2. Para alcançar este objetivo, definimos planejamentos de experimentos visando mensurar informações estatísticas que nos mostrem os gargalos da nossa arquitetura. Além disso, métricas de monitoramento são adotadas para mensurar o consumo de memória RAM, CPU e disco de modo que contribuam para uma melhor seleção de algoritmo visando ambientes com baixo poder computacional.

3.4.1 Domínio da Aplicação

Atualmente e possivelmente nos próximos anos, os dispositivos inteligentes serão tão comuns em nossas vidas que estarão por todas as partes. O campo de aplicação *IoT* é extremamente amplo, com aplicações em diferentes áreas como já explicado em capítulos anteriores. De acordo com Manyika *et al.* (2013) o setor de *Healthcare* é um dos domínios com mais utilização de dispositivos *IoT* com cerca de 41% de utilização, enquanto que o setor de manufatura possui 33%, aplicações envolvendo eletricidade 7% e o restante cerca de 11% divididos entre outros domínios.

Estes dados comprovam a importância da *IoT*, principalmente em aplicações envolvendo cuidados com a saúde. Porém, esse domínio de aplicação também possui os desafios da *IoT* já mencionados anteriormente. Logo, por geralmente ser considerado um domínio de aplicação

crítico e com informações sensíveis e confidenciais de pacientes, o uso de segurança adicionada ao desempenho se torna mais importante. Dentre os trabalhos levantados e discutidos no Capítulo 3, a maior parte deles envolvem estudos neste domínio de aplicação reafirmando a importância deste campo. Partindo destas premissas, decidimos escolher este domínio de aplicação para construção e avaliação dos testes desta pesquisa de mestrado por meio de nossa arquitetura de controle de acesso IoT-SHASEC.

3.5 Considerações Finais

Este Capítulo apresentou os Trabalhos Relacionados, descrevendo uma análise crítica sobre os estudos primários relacionados a esta dissertação de mestrado. Apresentou uma metodologia para a definição da *string* de busca e dos critérios de seleção dos trabalhos, que foram fundamentais para o levantamento e análise crítica dos mesmos vinculados à esta pesquisa. Após a seleção dos trabalhos, foram extraídas as principais características encontradas nos estudos selecionados. Além disso, para auxiliar o levantamento destas características, utilizou-se um formulário do *Google forms* para extração dos dados principais.

Notamos que o uso de *Json Web Tokens* está presente na maior parte dos trabalhos selecionados, porém, as discussões envolvendo a relação segurança x desempenho é pouco explorada. Embora, exista uma avaliação de desempenho na maior parte dos trabalhos, estas são avaliações superficiais e que não levam em consideração informações importantes como o algoritmo de assinatura e encriptação, estrutura do *token*, carga de trabalho, comunicação, entre outras métricas que influenciam no tempo de resposta final. Tais métricas são importantes principalmente para ambientes de dados sensíveis e críticos como por exemplo dados de saúde e informações hospitalares. Tais ambientes, necessitam um tempo de resposta baixo e uma segurança considerável para uma tomada de decisão assertiva. Diante desta lacuna identificada, nosso trabalho apresenta uma arquitetura de autorização para dados sensíveis, tendo como base para um estudo de caso, dados de saúde providos por dispositivos inteligentes em que nosso objetivo visa um estudo de *trade-off* entre desempenho e segurança analisando diferentes modelos criptográficos.

Por fim, este Capítulo apresentou o caminho percorrido para encontrar a identificação do problema e a proposta do protótipo IoT-SHASEC.

ARQUITETURA IOT-SHASEC

4.1 Considerações Iniciais

Este Capítulo tem por objetivo apresentar as informações relacionadas ao desenvolvimento desta dissertação de mestrado. Na Seção 2, apresentamos o protótipo da arquitetura IoT-SHASEC desenvolvido com todos seus componentes e o relacionamento entre eles. Na Seção 3, apresentamos a construção do tempo médio de resposta e todos os seus componentes e por fim na Seção 4 encerramos este capítulo com as considerações finais.

4.2 A Arquitetura IOT-SHASEC

A arquitetura IoT-SHASEC é um modelo arquitetural para aplicações de controle de acesso que se utilizam de *tokens*. Ela foi projetada para oferecer um serviço de autorização para usuários com enfoque em ambientes que trabalham com dados sensíveis de saúde pública gerados por dispositivos inteligentes. Esta arquitetura requer cinco componentes para o seu funcionamento e um modelo de alto nível da IOT-SHASEC pode ser observado por meio da Figura 12. Inicialmente é necessária a existência do usuário que possui algum recurso a ser protegido, armazenado e consumido. Esses recursos são basicamente as informações do usuário e em nosso modelo o recurso equivale as informações de dados clínicos de pacientes com diabetes. Este usuário é conhecido como *Resource Owner* ou proprietário do recurso. Como exemplo, podemos citar um paciente que deseja acessar seus dados clínicos que estão salvos em uma base de dados. Em seguida é necessária uma aplicação *Cliente* que interaja com o usuário e se responsabiliza por solicitar o recurso desejado. Por exemplo, um aplicativo *mobile*, *desktop* ou *Web* que possua uma interface para solicitação e visualização dos dados clínicos do usuário. Em nossa arquitetura estamos utilizando o Apache JMeter como aplicação Cliente para geração da carga de trabalho o qual descrevemos melhor nas próximas seções.

Outro componente é o *broker* que possui a finalidade de atuar como um agente otimizador

Figura 12 – IOT-SHASEC - Modelo Abstrato

para fornecer uma melhor qualidade de serviço para nossa aplicação cliente. Adotamos o Nginx para atuar como nosso broker e que funciona como *gateway* para o provedor de identidade. No entanto, além de ponte para o provedor de identidade ele foi utilizado como um balanceador de carga. Deste modo, quando ativada essa funcionalidade o *broker* Nginx divide as requisições para as replicações do nosso provedor de identidade. Por fim, o último componente é o *provedor de identidade* que irá gerar e validar a autorização para o usuário por meio de um *token* de acesso. O provedor de identidade foi construído adotando o protocolo *OAuth 2* que é considerado o padrão no setor de autorização fornecendo fluxos de autorização específicos para aplicativos da *Web*, aplicativos de *desktop*, telefones celulares e dispositivos inteligentes. De forma resumida os componentes são definidos como:

1. **Resource Owner:** Usuário que autoriza um aplicativo a acessar sua conta. O acesso do aplicativo à conta do usuário é limitado ao "escopo" da autorização concedida (por exemplo: acesso de leitura e/ou gravação).
2. **Cliente:** É o aplicativo que deseja acessar a conta do usuário. Antes de fazer isso, ele deve ser autorizado pelo usuário e a autorização deve ser validada pelo Servidor de autorização (OAuthServer). Em nossa aplicação, utilizamos o Apache JMeter como aplicação Cliente para a geração da carga de trabalho (requisições).
3. **Broker Nginx:** Servidor Nginx que recebe a requisição do Cliente e encaminha para o provedor *OAuth 2*. Em nossa arquitetura ele funciona como um **Proxy** e como um **Balanceador de carga**.
4. **OAuthServer:** Responsável inicialmente pela geração do *token* de acesso (Servidor de autorização) para aplicação Cliente e posteriormente pela validação e liberação do recurso (Servidor de Recursos).
5. **Base de dados:** Nossa arquitetura possui duas bases de dados construídas em um modelo relacional com MySQL. A base (bdautentica) com dados utilizados para autenticação e autorização de acesso e a base (bdsaude) referente as informações de dados clínicos de

saúde pública disponibilizada para pesquisa referente aos prontuários de setenta pacientes portadores de diabetes.

A Figura 13 ilustra uma instância de protótipo construído utilizando o *broker* Nginx como balanceador de carga e o provedor de identidade replicado em um segundo servidor.

Figura 13 – Instância IoT-SHASEC

4.2.1 Fluxo de execução IOT-SHASEC

A Figura 14 apresenta o passo a passo da nossa arquitetura para consumir uma informação na base de dados. Este modelo leva em consideração que estamos utilizando um balanceador de carga no *broker* Nginx. Inicialmente, é necessário adquirir um *token* de acesso válido (Servidor de autorização) e em seguida passar este *token* em uma nova requisição para acessar a informação. Esse processo é validado pelo provedor de autenticação (Servidor de Recurso) que valida ou rejeita a solicitação da aplicação Cliente. O processo completo é exemplificado a seguir:

1. *Resource Owner* insere um *username* e um *password* na aplicação Cliente;
2. Aplicação Cliente realiza uma solicitação HTTP Basic passando as credenciais inseridas pelo *Resource Owner* no *endpoint* */oauth/token* para nosso servidor Nginx;
3. O servidor Nginx recebe a solicitação do Cliente e encaminha para o OAuthServer no mesmo *endpoint* */oauth/token*. Caso o recurso de *Load balance* esteja ativado o *broker* Nginx escalonará as solicitações para os OAuthServers especificados.
4. O AuthServer possui dois papéis principais: *Authorization Server* e *Resource Server*. Inicialmente as requisições chegam no *endpoint* *oauth/token* do *Authorizarion Server*. O

Figura 14 – Sequência de execução da IoT-SHASEC

Authorization Server acessa o banco de dados de autenticação por meio de uma conexão JDBC em uma máquina remota e verifica se as informações enviadas inicialmente pelo *Resource Owner* são válidas.

5. Se as informações forem válidas é gerado um *token JWT* juntamente com um *token* de atualização que será assinado simetricamente com *HMAC* ou por assinatura assimétrica com o algoritmo *RSA*.
6. Depois que os *tokens* são gerados, eles são retornados para o *broker* Nginx que em seguida retorna para a aplicação Cliente. Assim que a aplicação Cliente recebe o *token* de acesso, ela pode fazer uma nova solicitação para o *endpoint* */dados_saude:12f5* que possui o recurso desejado passando o *token* recebido anteriormente.
7. O *broker* Nginx então recebe a nova solicitação e encaminha para o servidor OAuthServer que funcionará agora como o provedor do recurso. Quando a requisição chegar no *Resource Server*, o *token* existente na requisição será validado. Em caso de sucesso, o *Resource server* disponibiliza a informação solicitada acessando a base de dados de saúde e retornando a

informação que passará novamente ao *broker* Nginx até chegar na aplicação Cliente para o usuário.

8. Enquanto o *token* for válido, isto é, não tiver sido expirado, o mesmo *token* poderá ser utilizado para novas requisições sem que o *Resource owner*/usuário precise inserir novamente as credenciais de acesso.
9. Depois de expirado o *token* de acesso, podemos utilizar o *token* de atualização criado inicialmente junto com o *token* de acesso para solicitarmos um novo *token* de acesso.

Assim, o usuário não precisa ficar inserindo continuamente as credenciais de acesso sempre que o *token* for expirado. O *token* de atualização fica armazenado em um *Cookie HTTP* na aplicação Cliente e sempre que o *token* de acesso expirar, o *token* de atualização pode ser utilizado para conseguir um novo *Access token*. Por questões de segurança o *token* de acesso precisa ter um tempo de vida baixo, pois caso um atacante consiga interceptar este *token* o tempo de duração dele será curto e não dará tempo do atacante utiliza-lo. Por outro lado, quando solicitamos um *Access token* em nosso modelo retornamos também um *Refresh token* que está anexado em um *Cookie HTTP*. Este *Refresh token* serve unicamente para conseguir um novo *token* de acesso e pode ter um tempo de duração muito maior que o *Access token*. Toda vez que o *Access token* expirar utilizamos na aplicação Cliente o *Refresh token* para adquirir um novo *token*. Neste caso quando utilizamos o *Refresh token*, o usuário não necessita inserir suas credenciais de acesso novamente. Deste modo, somente depois que o *Refresh token* expirar o usuário terá que inserir novamente as credenciais para conseguir um novo *token* juntamente com o *Refresh token* e então todo o ciclo se repete.

4.2.2 Componentes da IOT-SHASEC

Tão importante quanto destacar os componentes da arquitetura é descrever suas funcionalidades e como eles trocam informações, as quais serão relevantes para entender os resultados dos experimentos. Assim, descrevemos com mais detalhes o funcionamento de cada componente isoladamente. As interações entre os componentes são ilustradas pela Figura 15 onde percebemos o fluxo da informação através da sequência dos números. Primeiramente, o usuário proprietário do recurso interage com a aplicação Cliente. Esta por sua vez se comunica com o broker e este interage com os provedores de acesso. Os provedores de acesso então consultam as bases de dados para validação e liberação dos recursos do usuário e retorna estes dados até o usuário do recurso novamente.

1 - Usuário/Resource Owner

Em nossa arquitetura o usuário também referenciado como *Resource Owner* possui uma função importante, pois ele é o proprietário do recurso e quem inicialmente envia as credenciais de acesso para geração da chave de capacidade por meio de um login e senha inserido na

Figura 15 – IOT-SHASEC - Interações dos componentes

aplicação Cliente. Além disso é o usuário que terá a visualização dos recursos e é importante destacar que a aplicação Cliente seja confiável do usuário conforme apresenta a Figura 16.

Figura 16 – Resource Owner

Em nosso modelo, o usuário precisa inserir a senha apenas na primeira solicitação para aquisição do *token*, nas próximas solicitações a busca do *token* de acesso será efetuada pelo uso de um *token* de atualização que foi armazenado no *Cookie* da aplicação Cliente no retorno da primeira requisição. O usuário apenas colocará novamente suas credenciais de acesso quando o *token* de atualização presente no cliente for expirado ou removido. Deste modo, o risco de alguém mal intencionado capturar as credenciais do usuário se torna menor.

2 - Aplicação Cliente

Para simular o papel da aplicação Cliente, adotamos o Apache JMeter. Este é um *software* de código aberto escrito em linguagem Java puro e desenvolvido pela primeira vez por Stefano Mazzocchi da Apache Software Foundation. Foi inicialmente projetado para testar aplicações da *Web* e posteriormente expandido para diferentes aplicações de teste. A escolha do JMeter como aplicação Cliente se deve primordialmente pelo fato de ser um *software* voltado para testes de desempenho, sendo possível simular uma carga pesada em um servidor, grupo de servidores, rede, banco de dados entre outros para análise de desempenho geral com diferentes tipos de carga.

Nossa arquitetura visa analisar um *trade-off* entre desempenho e segurança para aplicações de controle de acesso em Internet das Coisas. Por meio do JMeter podemos simular

diferentes comportamentos e definir cargas de trabalho para nossa API seguindo um intervalo de tempo definido de forma constante ou seguindo uma distribuição estatística de maneira muito mais eficiente do que o desenvolvimento de uma aplicação de interface *Frontend*. De forma geral, o JMeter simula um grupo de usuários enviando solicitações para um servidor de destino e retorna informações estatísticas por meio de diagramas gráficos.

A Figura 17 ilustra um exemplo do nosso plano de teste utilizando o Apache Jmeter. Neste exemplo utilizamos o temporizador *Uniform Random Timer* que segue uma distribuição uniforme com os parâmetros de *Offset* de atraso constante = 3000ms e Atraso aleatório máximo = 4000ms isso significa que o JMeter enviará um conjunto de requisições a um servidor destino com um intervalo variando de 3 à 7 segundos entre cada requisição.

Figura 17 – Plano de Teste para carga de trabalho utilizando o JMeter

Nosso Cliente JMeter necessita das credenciais do usuário e uma credencial válida da aplicação Cliente para seguir a notação do fluxo de concessão de autorização que adotamos. A Figura 18 (a) apresenta como uma requisição é construída pelo JMeter para adquirir um *Acess token*. Preenchemos os campos de protocolo com o HTTP e inserimos o endereço do *host* destino com o número da porta. Nosso *host* destino é o *broker* Nginx que está escutando na porta 8081, toda solicitação de *token* será por meio do método POST e no *endpoint* */oauth/token*. É necessário passar uma série de parâmetros para que o JMeter possa construir a requisição e fazer a solicitação ao *broker*. Os parâmetros obrigatórios são:

- **client**: um identificador da aplicação Cliente, em nosso caso chamado de **angular**;
- **username**: é o *username* do usuário, em nosso exemplo é considerado o endereço de *email* **admin@usp.br**;
- **password**: é a senha do usuário, em nosso exemplo a senha é **admin**;
- **grant_type**: é o tipo de concessão utilizado pelo *OAuth 2* para aquisição de um *Acess Token*. Nossa arquitetura implementa o fluxo de *Resource Owner Password Credentials Grant* o qual denominamos apenas **password**.

Em seguida necessitamos adicionar no *Header* do JMeter dois parâmetros:

- **Authorization:** Este campo é o que será validado em nosso servidor de autorização *OAuth 2*. Identifica que é uma requisição HTTP do tipo *Basic* passando o *username* e a senha do usuário codificado em base64, em nosso exemplo **YW5ndWxhcjphbmd1bGFy**;
- **Content-Type:** Especifica como os dados do formulário devem ser codificados ao serem enviados para o servidor.

Assim, nossa requisição para solicitar um *Access Token* está completa. A Figura 19 apresenta como podemos capturar o *token* que foi recebido para utiliza-lo na requisição seguinte para acessar o recurso do usuário. Utilizamos o pós-processador *JSON Extractor* e definimos a variável **access_token** que armazenará o *token* recebido. Fazemos uma nova solicitação para nosso *host* destino, no entanto, agora solicitamos o recurso do usuário. Para isso, mudamos o método HTTP para GET e passamos o *endpoint* **/dados_saude** de onde está o recurso solicitado. Devemos ainda passar o *token* que recebemos na primeira solicitação no *Header* desta nova requisição. A Figura 19 apresenta o campo **Authorization** com o novo parâmetro para requisição. Note que agora, nossa requisição não é mais do tipo *Basic* e sim do tipo *Bearer* que denota que agora estamos passando um *token* do tipo portador assinado criptograficamente que possui todas as regras de acesso para o nosso usuário. Se o *token* for validado com sucesso, as informações dos dados do usuário serão retornadas para nossa aplicação Cliente.

3 - Broker Nginx

Nosso *broker* Nginx possui duas funcionalidades. A primeira ele atua como um servidor de *Proxy* que pode bloquear ou filtrar as requisições que chegam até nosso servidor *OAuth 2*. A segunda funcionalidade refere-se a um balanceador de carga capaz de dividir as requisições em instâncias de nosso servidor *OAuth 2* seguindo um algoritmo de escalonamento. Por padrão o Nginx Open Source oferece suporte a quatro métodos de balanceamento de carga (Round Robin, Least Connections, IP Hash e Generic Hash) e o Nginx Plus adiciona mais dois métodos (Least Time e Random). Em nossa arquitetura, optamos por utilizar o escalonamento com Round Robin que fará o balanceamento dividindo as cargas de forma homogênea entre os provedores de autenticação e autorização existentes que objetiva a redução do tempo final de resposta. A escolha de adotarmos o Round Robin se deve ao fato das máquinas possuírem configurações de hardware similares, neste caso, as requisições são divididas de formas iguais entre os servidores. Além disso, configuramos um arquivo de *log* com variáveis de tempos internas para cada requisição proveniente do(s) cliente(s). Esse arquivo de *log* é importante porque guarda conhecimento da origem das requisições, *status code* e variáveis de tempo. As informações de tempo são gravadas em segundos com resolução de milissegundos e estão ilustradas pela Figura 20. Estas variáveis são essenciais pois auxiliam na composição do tempo total de resposta e contribuem para descoberta de possíveis gargalos.

Figura 18 – Construção da requisição no Apache JMeter

Cada variável interna é descrita como:

- **log_format main [time_local]:** Indica o nome do nosso arquivo de *log* e o formato de data/hora;
- **remote_addr:** Indica o endereço IP da máquina origem que gerou a requisição;
- **request:** Indica qual o tipo de requisição recebida (GET, POST, DELETE e etc);
- **status:** Indica o *status code* da requisição (200/sucesso, 201/criado, 500/erro interno e etc);

(a)

(b)

Figura 19 – Construção da requisição no Apache JMeter

```

w@w:~$ nano /etc/nginx/conf.d
GNU nano 2.5.3 File: log_format.conf

log_format main '[${time_local}] '
#log_format main '[${time_iso8601}] '
    '$remote_addr '
    '"$request" '
    '$status '
    '$upstream_connect_time '
    '$upstream_header_time '
    '$upstream_response_time '
    '$request_time ';
access_log /var/log/nginx/nginx.log main;

```

Figura 20 – Configuração do arquivo de *log format*

- **upstream_connect_time:** Tempo gasto estabelecendo uma conexão com um servidor *upstream*;

- **upstream_header_time**: Tempo entre o estabelecimento de uma conexão com um servidor *upstream* e o recebimento do primeiro *byte* do cabeçalho de resposta;
- **upstream_response_time**: Tempo entre o estabelecimento de uma conexão com um servidor *upstream* e o recebimento do último *byte* do corpo da resposta;
- **request_time**: Tempo de solicitação completo, começando quando o Nginx lê o primeiro *byte* do Cliente e terminando quando o Nginx envia o último *byte* do corpo da resposta;
- **access_log/var/log/nginx/nginx.log main**: Caminho de onde será gravado nosso arquivo de *log*.

A segunda configuração que realizamos refere-se ao destino das requisições, ou seja, para onde e para quantas instâncias do nosso provedor *OAuth 2* serão responsáveis pelo atendimento das requisições. A Figura 21 apresenta este arquivo de configuração, onde definimos os IP's que serão direcionados as requisições e a porta que nosso *broker* está ouvindo.

```
welington@and08: /etc/nginx/sites-available 80x17
GNU nano 2.5.3      File: default      Modified
upstream location {
    server 10.1.1.90:8080;
#    server 10.1.1.110:8080;
}
server {
    listen 8081;
    location / {
        proxy_pass http://location;
    }
}
```

Figura 21 – Configuração do arquivo de *log Proxy/Load Balance*

4 - Servidor OAuth 2

Nosso provedor de identidade é dividido em duas identidades: o servidor de autorização (*Authorization Server*) e o servidor de recursos (*Resource Server*). Cada identidade é representada e gerenciada por uma classe no código de nossa API. O Servidor de autorização como o próprio nome diz é responsável pela geração dos *tokens* para a aplicação Cliente. Neste caso, é necessário definir nesta classe as informações da aplicação Cliente, o fluxo de concessão para requisitar o *token*, o modelo de *token* (em nosso caso JSON), *scopo*, validade do *token*, estrutura e por fim, o método de assinatura. Já a classe responsável pelo nosso *Resource Server* possui a finalidade de validar se o *token* gerado pelo Servidor de Autorização é válido e permitir o acesso ao recurso do usuário conforme as regras que estão definidas no *Access token*. Por exemplo, se no *Access token* estiver definido que o Cliente tem apenas autorização de leitura e o mesmo estiver tentando inserir um registro na base o *Resource Server* não permitirá a escrita.

De forma geral, o *OAuth 2* utiliza-se essencialmente de *tokens*: *CLIENT_ID*, *CLIENT_SECRET*, *GRANT_CODE*, *ACCESS_TOKEN* e *REFRESH_TOKEN*. Alguns desses *tokens* são

emitidos na criação do aplicativo e outros são emitidos a partir de um dos fluxos do *OAuth 2* listados a seguir: *Client Credentials*, *Resource Owner Password Credentials*, *Authorization Code*, *Implicit* e *Refresh Token*. Cada fluxo do *OAuth 2* possui características próprias e cenários de utilização, e a escolha de qual fluxo a ser adotado pode ser feita conforme o fluxo da Figura 6. Devido a confiabilidade da nossa aplicação Cliente, adotamos o fluxo *Resource Owner Password Credentials*, este fluxo é utilizado por aplicativos confiáveis onde é necessário que seja feita a autenticação do aplicativo e do usuário do aplicativo. Ele tem por objetivo obter um *ACCESS_TOKEN* por meio do uso do *CLIENT_ID* e *CLIENT_SECRET* do aplicativo e dos dados de autenticação do usuário (exemplo: login e senha). Por se tratar de um aplicativo confiável, o login e a senha do usuário pode ser digitado no próprio aplicativo. Juntamente com o *Resource Owner Password Credentials* utilizamos o fluxo de *REFRESH_TOKEN* que tem como objetivo obter um novo *ACCESS_TOKEN* quando o mesmo estiver expirado por meio do uso de um *token* de atualização *REFRESH_TOKEN*.

O *REFRESH_TOKEN* é criado juntamente com o *ACCESS_TOKEN* como resultado do processo de autenticação. Embora o *OAuth 2* trabalhe essencialmente com *tokens*, ele não define um formato de *token* padrão para utilização. O *JSON Web Token (JWT)* surge então como complemento para nosso provedor de identidade *OAuth 2*. Os *tokens JWT* definem uma maneira compacta e independente para transferência de informações entre partes com segurança por meio de um objeto JSON, estas informações são assinadas digitalmente utilizando um segredo (com o algoritmo *HMAC*) ou por um par de chaves pública/privada usando **RSA** ou **ECDSA**. Os *tokens JWT* são o cerne da nossa arquitetura, pois são onde se encontram as informações sensíveis e necessárias para fornecer o acesso aos recursos do usuário. É possível inserir várias informações no *token* conforme a necessidade da aplicação, contudo, quanto mais informações são inseridas e o *token* cresce, maior será o tempo para processar e trafegar-lo pela rede. Assim, definimos apenas as informações essenciais para a nossa aplicação de estudo. A Figura 22 exemplifica o *token* construído, o qual é descrito pelas reivindicações:

1. **Alg**: Algoritmo utilizado para assinar o *token*;
2. **JWT**: Identifica que é um *token JSON*;
3. **Username**: Credencial que deverá ser informada inicialmente pelo *Resource Owner*, neste caso o *email*;
4. **Scope**: Escopo que o *Resource Owner* está inserido. Em nosso exemplo o escopo é de leitura e escrita;
5. **Nome**: Nome do usuário;
6. **Exp**: Tempo em milissegundos que o *token* estará válido para uso;
7. **Authorities**: São as regras definidas para cada usuário. Em nosso exemplo utilizamos um usuário administrador que possui todas as permissões no sistema;

Figura 23 – Bases utilizadas

possui trabalhos científicos publicados. A base é composta de informações de pacientes com diabetes e está estruturada da seguinte forma:

Informações do conjunto de dados:

Os prontuários de pacientes com diabetes foram obtidos de duas fontes: um dispositivo de gravação eletrônica automática e registros em papel. O dispositivo automático tinha um relógio interno para registrar os registros de data e hora, enquanto os registros em papel forneciam apenas intervalos de "tempo lógico" (café da manhã, almoço, jantar, hora de dormir). Para registros em papel, os horários fixos foram atribuídos ao café da manhã (08:00), almoço (12:00), jantar (18:00) e hora de dormir (22:00). Assim, os registros em papel têm tempos de gravação uniformes fictícios, enquanto os registros eletrônicos têm registros de hora mais realistas.

Os arquivos de diabetes consistem em quatro campos por registro. Cada campo é separado por uma guia e cada registro é separado por uma nova linha. Os campos são: Data no formato MM-DD-AAAA, Hora no formato XX:AA, Código e Valor. O campo Código é decifrado conforme Tabela 6. A base é constituída por informações de setenta pacientes com um total de 29270 registros. No entanto, houve a necessidade de adicionarmos dois novos campos para a construção da nossa base onde um campo identifica o paciente e outro campo identifica a medição realizada. Deste modo, cada solicitação de dado é feita levando em consideração apenas um registro de medição. Estas e demais informações podem ser consultadas em ¹.

¹ <https://archive.ics.uci.edu/ml/datasets/Diabetes>

Tabela 6 – Tabela de composição do código chave/valor

Código	Valor
33	Dose regular de insulina
34	Dose de insulina NPH
35	Dose de insulina UltraLente
48	Medição de glicemia não especificada
57	Medição de glicemia não especificada
58	Medição de glicemia antes do café da manhã
59	Glicemia no pós-café da manhã medição
60	Medição da glicemia antes do almoço
61	Medição da glicemia pós-almoço
62	Medição da glicemia antes da ceia
63	Medição da glicemia pós-ceia
64	Medição da glicemia antes do lanche
65	Sintomas hipoglicêmicos
66	Ingestão típica de refeições
67	Ingestão de refeições mais que o normal
68	Ingestão de refeições que não é habitual
69	Atividade típica do exercício
70	Atividade acima do usual
71	Atividade abaixo do normal
72	Evento especial não especificado

4.3 Composição do tempo de resposta

O tempo de resposta é a principal variável de análise em nossa arquitetura IoT-SHASEC. Por meio dele podemos selecionar um determinado algoritmo para assinar ou criptografar nosso *token JSON*. Vários fatores podem contribuir positivamente ou negativamente para a construção do tempo de resposta, com o objetivo de verificar estes fatores nós segmentamos o tempo de resposta por cada módulo da aplicação. Assim, temos maior visibilidade de quais componentes tem causado um maior ou menor gargalo na arquitetura e prover mecanismos que melhorem os pontos identificados.

O tempo médio final de resposta é dado pela nossa aplicação Cliente (Apache JMeter). No JMeter temos o tempo desde que é iniciado a solicitação até o retorno da mesma. Deste modo, o tempo do Cliente é construído por todos os demais sub-tempos da arquitetura. Os tempos são coletados seguindo formatos próprios de cada componente que processa as requisições. No Cliente JMeter os tempos são armazenados em arquivos de *log* com extensão *.jtl* que por sua vez estão estruturados em um formato JSON. Os tempos são separados por cada tipo de requisição (*Get Access Token e Get Data*). Em nossa arquitetura e planos de testes trabalhamos com uma requisição inicial para solicitar um *access token* e uma requisição final para solicitar/insere um recurso do usuário. Quando as duas requisições (*Get Access token e Get Data*) chegam ao nosso

broker Nginx, utilizamos o arquivo de *log* (**log_format**) explicado em seções anteriores para guardar as informações do tempo de resposta no *broker* entre outras informações importantes. Por fim, quando as requisições chegam em nosso servidor *OAuth 2*, coletamos o tempo de processamento pela API no código fonte com ajuda do *Log4j* e direcionamos o tempo de processamento para um ultimo arquivo de *log*. De forma geral, todo o tempo de resposta é armazenado em três arquivos de *logs* independentes e depois fazemos o processamento destes arquivos para o cálculo e somatório de todos os sub-tempos existentes.

A Figura 24 ilustra de uma perspectiva temporal a composição das partes que integram o tempo médio final de resposta para aquisição de um *token JWT* e para aquisição de um recurso do usuário.

Figura 24 – Composição do tempo de resposta

A composição do tempo médio de resposta engloba métricas da aplicação Cliente, do *broker* Nginx e de nossa API de autenticação e autorização o qual estão descritos pela Tabela 7 e decompostos pela Tabela 8. Deste modo, temos o tempo total de resposta definido pela equação

$$\gamma = A + G + H + I + F \quad (4.1)$$

onde temos um somatório das métricas definidas na Tabela 8 e que resultam no tempo final na aplicação Cliente.

4.4 Considerações Finais

Este Capítulo apresentou o desenvolvimento do projeto de mestrado descrevendo detalhadamente a construção do projeto de pesquisa, o protótipo da arquitetura com seus módulos e como eles se relacionam e por fim, apresentamos como é formado o tempo de resposta médio final com todos os seus componentes. No próximo Capítulo, como forma de validar o funcionamento da arquitetura, descreveremos um processo formal de planejamento de experimentos e a condução de uma avaliação de desempenho para análise do protótipo desenvolvido.

Tabela 7 – Componentes do tempo de resposta

Aplicação	Métrica	Descrição
Cliente	<i>Elapsed</i>	Tempo decorrido imediatamente antes de enviar a solicitação e logo após o recebimento da última parte da resposta
Cliente	<i>Connect</i>	Tempo necessário para estabelecer a conexão, incluindo o <i>handshake SSL</i>
Broker	<i>Connect Time</i>	Tempo gasto estabelecendo uma conexão com um servidor <i>upstream</i>
Broker	<i>Header Time</i>	Tempo entre o estabelecimento de uma conexão e o recebimento do primeiro <i>byte</i> do cabeçalho de resposta
Broker	<i>Response Time</i>	Tempo entre o estabelecimento de uma conexão e o recebimento do último <i>byte</i> do corpo da resposta
Broker	<i>Request Time</i>	Tempo de solicitação completo começando quando o Nginx lê o primeiro <i>byte</i> do cliente e terminando quando o Nginx envia o último <i>byte</i> do corpo da resposta
API	<i>Processor Filter</i>	Tempo gasto quando API recebe a solicitação e realiza o processamento completo
Rede	Time CB	Tempo gasto na comunicação entre o Cliente e o <i>Broker</i>
Broker	Time Broker	Tempo gasto de processamento do <i>Broker</i>
Rede	Time BA	Tempo gasto na comunicação entre <i>Broker</i> e API

Tabela 8 – Decomposição do tempo de resposta

Métrica	Variável	Decomposição
<i>Elapsed</i>	X	X
<i>Connect</i>	A	A
<i>Connect Time</i>	B	B
<i>Header Time</i>	C	C
<i>Response Time</i>	D	D
<i>Request Time</i>	E	E
<i>Processor Filter</i>	F	F
Time CB	G	X-A-E
Time Broker	H	E-D-B
Time BA	I	E-F-H

AVALIAÇÕES EXPERIMENTAIS

5.1 Considerações Iniciais

Este Capítulo apresenta os resultados obtidos desta dissertação de mestrado e as contribuições alcançadas por meio das avaliações realizadas do protótipo IOT-SHASEC. Na seção 1, apresentamos um breve resumo do conceito de avaliação de desempenho. Na seção 2 definimos o planejamento de experimento com os fatores fixos e variáveis para cada experimento e os resultados oriundos de cada experimento. Por fim, na seção 3 finalizamos o Capítulo com as considerações finais.

5.2 Avaliação de desempenho

Avaliação de desempenho é um processo aplicado em um sistema computacional com o intuito de avaliar resultados de experimentos. Um dos grandes desafios que há na computação, refere-se em manter um equilíbrio aceitável entre confiabilidade e desempenho. Pois são, de forma geral considerados antagônicos, ou seja, a medida que o desempenho de um sistema cresce a sua confiabilidade se torna menor. O objetivo primordial de uma avaliação de desempenho é obter resultados que possibilitam atingir ou prover o maior desempenho possível com o menor custo. Com os resultados obtidos da avaliação, é possível determinar a melhor configuração de um determinado cenário de *software* e/ou *hardware* associados. O que permite, aferir qual deles ou qual combinação é mais eficiente na solução de um problema, e quão essa solução é melhor ou pior que às demais opções existentes. Para uma avaliação bem construída, uma série de fatores precisam ser analisados partindo inicialmente da formulação do desenho de experimentos.

(JAIN, 1990) explica que o planejamento de experimentos objetiva obter o máximo de informação com o mínimo de experimentos possíveis. Segundo o mesmo autor, a análise adequada de experimentos também ajuda a separar os efeitos de vários fatores que podem afetar

o desempenho, além de determinar se um fator terá um efeito significativo no experimento ou se a diferença observada é simplesmente devido a variações aleatórias causadas por erros de medição e parâmetros que não foram controlados. (JAIN, 1990) discorre existir alguns termos que fazem parte do processo de análise de experimentos, tais como:

- **Variável de resposta:** Refere-se a saída de um experimento e geralmente é considerada a métrica utilizada na avaliação do sistema;
- **Fatores:** Variável que afeta as variáveis de resposta e que podem assumir diversas alternativas. Podem ser classificados como Fatores Primários quando causam um grande impacto em uma variável de resposta e que devem ser considerados e Fatores Secundários, quando o impacto na variável de resposta não é significativo ou não se tem interesse em quantificar;
- **Níveis:** Os valores que um determinado fator pode assumir;
- **Interação:** Indica a dependência entre os fatores avaliados;
- **Replicação:** Repetição de todo ou de parte de um experimento;
- **Projeto:** Determina o número de experimentos a serem considerados, incluindo o número de fatores e níveis, a combinação entre os níveis e o número de replicações para cada experimento.

Há vários modelos de planejamentos de experimentos na literatura, no entanto, este trabalho adotou o modelo Fatorial Completo 2^k que consiste em utilizar todas as combinações possíveis em todos os níveis de todos os fatores. A grande vantagem deste modelo está na possibilidade de examinar todas as combinações possíveis, bem como, as cargas de trabalho, além de permitir analisar as influências entre os fatores. Sua desvantagem refere-se em alguns casos ao elevado número de experimentos.

5.3 Monitoramento

Uma tarefa importante para a análise de desempenho se refere ao monitoramento de recursos computacionais pois é uma das formas de entendermos melhor o consumo do sistema, banco de dados, *Cluster* ou de uma arquitetura como um todo. Durante um teste de carga, é importante conhecer a integridade dos servidores carregados e também se a carga de trabalho foi despachada corretamente, isso pode ser feito por meio da técnica de aferição por coleta de dados já explanada no Capítulo 2. Contudo, é importante que a coleta das métricas selecionadas não influencie na avaliação realizada. Com o intuito de auxiliar em nossa avaliação, inserimos agentes monitores em *hosts* de nossa arquitetura coletando métricas de desempenho. Para isso, utilizamos o monitor *Perfmon* que é um *plugin* adicionado ao *Apache JMeter* que permite o monitoramento e coleta de mais de 75 métricas de desempenho. Para nossa análise consideramos

o monitoramento de CPU, Memória RAM e Disco. A Figura 25 apresenta um exemplo de configuração no *Perfmon* no *Apache JMeter* onde é necessário inserir o endereço IP ou nome do servidor a ser monitorado, a porta de escuta do agente e as métricas que se deseja coletar. Em seguida, em cada servidor é necessário iniciar o agente que coletará as métricas no período de teste do *JMeter*. A Figura 26 ilustra os *hosts* monitorados em nosso cenário de experimento, maiores informações sobre o *Perfmon* podem ser consultadas em sua página oficial ¹.

Figura 25 – Configuração do Agente Perfmon

5.4 Ambiente de Configuração

Definimos inicialmente dois planejamento de experimentos para avaliação de nossa arquitetura. O experimento A e B que são descritos em seguida. No entanto, antes da execução dos experimentos é necessário definir a configuração do ambiente de teste e o cenário do ambiente.

Nosso ambiente de teste é composto por 14 máquinas *hosts*, sendo: 10 *hosts* representando os Clientes responsáveis pela geração da carga de trabalho. Uma máquina *host* que é utilizada como *broker* onde está configurado o Servidor Nginx, esta máquina é utilizada como um *Proxy* que encaminha as requisições dos Clientes e como um balanceador de carga que faz a divisão igualitária da carga de trabalho entre os provedores de identidade (OAuthServers). Uma máquina *host* é definida para o provedor de identidade que está replicada em outra máquina. Por fim, 2 máquinas foram utilizadas para o armazenamento das bases de dados. A base de dados de

¹ **Perfmon:** <https://jmeter-plugins.org/wiki/PerfMon/>

Figura 26 – Servidores monitorados

autenticação e a base de dados de saúde estão em *hosts* diferentes com o intuito de aumentar a segurança da arquitetura e deixa-la mais distribuída. As máquinas utilizadas são homogêneas, ou seja, possuem a mesma configuração de *hardware* apresentada na Tabela 9. O cenário de teste está representado pela Figura 27.

Tabela 9 – Configuração *clusters* do *LaSDPC*

Cluster	Nodes	CPU	Memória	Disco
Andromeda	13	AMD Vishera 4.2 Ghz	32 GB	2TB
Halley	13	Intel Core I7 3.60 Ghz	32 GB	2 TB
Cosmos	19	Intel Core 2 Quad 2.66 GHz	8 GB	160 GB

5.4.1 Desenho do experimento A

Este experimento teve por objetivo, avaliar o tempo médio de resposta de dois métodos de criptografia para assinatura de *tokens JWT*. O experimento ainda visou verificar a influência que a adição de um balanceador de carga exerce no tempo de resposta final. Definimos o tempo médio de resposta como nossa variável de resposta e adotamos o modelo de fatorial completo 2k com quatro fatores e dois níveis para cada fator. A parametrização destes fatores pode ser observada pela Tabela 10 e na Tabela 11 apresentamos todas as combinações geradas a partir dos fatores e níveis definidos. Outro ponto-chave analisado em nosso cenário de experimento é a definição de fatores fixos que devem ser considerados na parametrização para aplicação do teste. Em nosso cenário experimental esses fatores estão destacados na Tabela 12.

Figura 27 – Cenário de teste

Tabela 10 – Parametrização dos fatores e níveis do experimento

Parâmetros Variáveis	
Fatores	Níveis
Arquitetura	Proxy e Load Balance
Total de Requisição	1000 e 10000
Tipo de Requisição	Get Access Token e Get Data
Criptografia	Simétrico e Assimétrico

Tabela 11 – Configuração experimental

Configuração experimental					
Exp	D Criptografia	C Tipo de Req	B Total de Req	A Arquitetura	Variável de Resposta Tempo Médio
1	Simétrico	Get Access Token	1000	Proxy	-
2	Simétrico	Get Access Token	1000	Load Balance	-
3	Simétrico	Get Access Token	10000	Proxy	-
4	Simétrico	Get Access Token	10000	Load Balance	-
5	Simétrico	Get Data	1000	Proxy	-
6	Simétrico	Get Data	1000	Load Balance	-
7	Simétrico	Get Data	10000	Proxy	-
8	Simétrico	Get Data	10000	Load Balance	-
9	Assimétrico	Get Access Token	1000	Proxy	-
10	Assimétrico	Get Access Token	1000	Load Balance	-
11	Assimétrico	Get Access Token	10000	Proxy	-
12	Assimétrico	Get Access Token	10000	Load Balance	-
13	Assimétrico	Get Data	1000	Proxy	-
14	Assimétrico	Get Data	1000	Load Balance	-
15	Assimétrico	Get Data	10000	Proxy	-
16	Assimétrico	Get Data	10000	Load Balance	-

Tabela 12 – Valores fixos associados aos parâmetros do experimento

Parâmetros fixos	
Distribuição estatística	Uniforme
Algoritmo escalonador	Round Robin
Refresh Token	Sim
Operação	Leitura
Número de Clientes	10
Ramp-Up Period	10 segundos
Número de Iterações Por Thread	5
Número de Replicações	10

Definimos um plano de teste no Apache JMeter que pode ser visualizado pela Figura 28. Utilizamos o temporizador *Uniform Random Timer* que segue uma distribuição uniforme com os parâmetros de *Offset* de atraso constante = 3000ms e Atraso aleatório máximo = 4000ms. Portanto, ele definirá um atraso aleatório de 3 à 7 segundos para cada requisição enviada ao *broker* Nginx. Em casos em que a configuração experimental levou em consideração o uso do *broker* como balanceador de carga, fixamos o *Round Robin* como algoritmo escalonador. Para toda requisição *Get Access Token*, estamos retornando juntamente o *Refresh Token* anexado a um *Cookie HTTP*. Também fixamos a operação de leitura, onde apenas serão retornados informações do banco de dados. Utilizamos como Cliente o Apache JMeter configurado para uma arquitetura mestre/escravo para testes distribuídos. Ao todo utilizamos 10 nós geradores de carga com 9 escravos e 1 nó mestre. Definimos um total de 5 iterações para cada *thread* nas máquinas clientes e um tempo de espera de 10 segundos para inicialização do teste. Para o propósito deste estudo replicamos o experimento 10 vezes. Isso foi feito para fornecer melhor confiabilidade para a análise dos resultados.

Figura 28 – Plano de teste para carga de trabalho utilizando o JMeter

Análise do Experimento A

A análise dos resultados leva em consideração primordialmente o tempo médio de resposta. Buscamos analisar este tempo separando-o em sub-tempos menores de cada ação dentro de nossa arquitetura. Deste modo, é possível identificar onde se concentra o maior gargalo e construir melhores mecanismos futuros para otimização. Junto ao tempo médio total, calculamos o desvio padrão, a variância, o tempo máximo e mínimo de resposta e o intervalo de confiança, com 95% do grau de confiança com base nos valores da tabela T-Student. Os resultados iniciais de nosso experimento ilustra na Figura 29 e na Figura 30 o tempo médio de resposta com os dois métodos de assinatura (*HMAC e RSA*) para os dois tipos de requisições (*Get Access Token e Get Data*). Percebemos pela Figura (a) que quando utilizamos o algoritmo *RSA* com um total de 10 mil requisições e sem o balanceador de carga, o tempo médio de resposta se torna aproximadamente 3.3 vezes maior que o tempo gasto utilizando o modelo simétrico. A medida que o número de requisições se torna maior e demanda um processamento elevado pelo provedor de identidade, a criptografia assimétrica diminui drasticamente o desempenho, ocasionando em um aumento do tempo de resposta final para adquirir um *Access Token*. Deste modo, o uso de um balanceador de carga se mostra fundamental para aumentar o desempenho da aplicação ao mesmo tempo que auxilia na diminuição do tempo de resposta.

Figura 29 – Custo com Proxy

A Figura 31 apresenta o tempo de resposta total para as requisições utilizando assinatura simétrica com *HMAC*. Os resultados variam de 1000 à 10000 requisições adotando o *broker* com a configuração de *Proxy* e Balanceador de carga. Percebemos pela Figura 31 (a) e (b), que as requisições possuem um tempo médio de resposta próximo uma da outra (90ms para *Get access token* e 10ms para *Get data*). Neste caso, para 1000 requisições com uma taxa de chegada variando de 3 à 7 segundos, nosso provedor *OAuthServer* consegue suprir as solicitações de forma similar a configuração com o *load balance*. Assim, podemos eliminar o uso do balanceador de

Figura 30 – Custo com Load Balance

carga e diminuir o custo de manutenção e gasto de energia da arquitetura. Por outro lado, quando aumentamos a quantidade de requisições para 10 vezes o valor inicial (10000), percebemos um grande aumento do tempo de resposta quando não utilizamos o *load balance*. Neste caso, um único provedor *OAuthServer* demanda um alto tempo para processamento da informação causando um enfileiramento no nosso servidor. Quando adicionamos mais um provedor de identidade, percebemos que o tempo volta a ser normalizado. Podemos concluir desta análise que para um volume baixo de solicitações enviada ao nosso provedor, não é necessário o uso de um balanceador de carga pois ele pouco afeta na diminuição do tempo de resposta final. No entanto, para uma carga alta de solicitações é necessário fazer o balanceamento entre dois ou mais provedores. É importante destacar, que mesmo com uma carga alta e utilizando apenas um *OAuthServer* as requisições não falharam, apenas houve uma gasto maior para realizar o processamento. Os resultados ilustrados pela Figura 32 seguem o mesmo comportamento discutido, no entanto o algoritmo em avaliação é o RSA.

A influência dos fatores estão divididos em influências principais ilustrado pela Figura 33 e secundárias ilustrados por meio da Figura 34. Analisando a influência entre os fatores principais, consideramos todas as combinações de experimentos e o tempo médio de resposta foi comparado para cada combinação de nível e fator existente com o intuito de identificar qual deles possui maior influência no tempo de resposta. O primeiro bloco apresenta uma comparação entre os dois níveis (*Proxy* e *Load Balance*) em relação ao fator Arquitetura. Percebemos que quando a configuração de *load balance* está ativada, ela contribui para a diminuição do tempo de resposta se comparado a configuração de *Proxy*. O segundo bloco apresenta a comparação entre os níveis de 1 mil e 10 mil requisições para o fator Número de Requisições. Percebemos que o aumento do número de requisições contribui para o aumento do tempo de resposta. No terceiro bloco é apresentado uma comparação entre os níveis *Get Access Token* e *Get Data* para

Figura 31 – Análise usando criptografia simétrica

o fator Tipo de Requisição. É possível verificar que a requisição para solicitar o *Access token* contribui para o aumento do tempo de resposta enquanto que a requisição *Get Data* possui uma influência pequena no tempo médio total. O processo para conseguir um *token* é mais complexo e demanda o uso de criptografia para assinar o *token*, enquanto que na requisição *Get data*, o *token* é apenas passado como um autenticador. Por fim, o quarto bloco apresenta a comparação entre os níveis simétrico e assimétrico para o fator Criptografia. Percebemos que a criptografia/assinatura assimétrica influencia em um aumento no tempo de resposta. Isso é devido ao uso do algoritmo *RSA* que é mais custoso se comparado ao *HMAC* que é mais simples. O aumento da segurança do *token* para muitas requisições exerce uma queda de desempenho se comparado ao tempo de resposta.

A Tabela 13 apresenta a divisão do tempo de resposta total para 10000 requisições em todo o processo para aquisição do token e em seguida para aquisição do dado levando em consideração o algoritmo simétrico RSA com o modelo de arquitetura utilizando o *broker* como Proxy e também como balanceador de carga. Os valores representam o método, média (AVG), desvio padrão (ST), intervalo de confiança (CI), limite superior (UL), limite inferior (LL), tempo mínimo (Min), tempo máximo (Max), 90%th percentil, tempo do cliente para estabelecer uma conexão (CCT), tempo de rede entre cliente e *broker* (NCB), tempo do *broker*, tempo de rede

Figura 32 – Análise usando criptografia assimétrica

Figura 33 – Influência de fatores principais

entre *broker* e api (NBA) e o tempo da Api.

Quando usamos a configuração de proxy o *broker* apenas filtra e encaminha as requisições até o provedor de identidade, porém, ao utilizarmos o balanceador de carga o proxy também divide a carga de trabalho entre os provedores de identidade existentes. Apresentamos a média, o desvio padrão, o intervalo de confiança com 5% de significância, os limites inferiores e superiores, o tempo mínimo e máximo e o valor de 90%th pct. Percebe-se que em 90% das requisições o tempo médio total para se adquirir um token foi de até 2948.70ms sendo o mínimo

Figura 34 – Influência de fatores de segunda ordem

de 88ms e o máximo de 5011ms. Porém, para requisitar um recurso protegido do usuário o valor foi de até 395.90ms sendo o mínimo de 2ms e o máximo de 3175ms quando utilizamos o Proxy. Ao adicionarmos o balanceador de carga a variação destes tempos fica de até 106.00ms para conseguir um token de acesso e até 5.00ms para requisitar um recurso do usuário. Neste caso, fica evidente que quando há uma quantidade de requisições elevada, o tempo médio de resposta final é relativamente alto e o nosso provedor OAuthServer fica sobrecarregado. Porém, quando utilizamos o balanceador de carga o provedor de identidade fica menos estressado e consegue tratar as requisições com maior eficiência impactando diretamente na diminuição do tempo de resposta. Percebemos ainda que o maior gargalo está no processamento do nosso provedor de identidade OAuthServer, enquanto que os demais componentes possuem um tempo de processamento relativamente baixo. Além disso, o tempo médio para requisitar o token é relativamente maior se comparado ao tempo de requisitar o recurso protegido do usuário. Isso se deve ao processo criptográfico para geração do token e também do Refresh token, pois em cada solicitação de token estamos retornando o token de atualização junto. Já na requisição do Get data passamos apenas o token contendo todas as permissões necessárias. Esse comportamento é seguido por todas as variações de níveis e fatores mencionados anteriormente na configuração experimental com 10 mil requisições. Assim, não apresentamos o comportamento para cada relação Fator/Nível. As Figuras 35 e 36 ilustram estes resultados graficamente e demonstram onde está o principal gargalo do tempo de resposta.

A Figura 37 apresenta os resultados relativos ao monitoramento de CPU e RAM para nosso provedor OAuth2 instanciado em dois servidores físicos (and09 e and11). O uso do

Tabela 13 – Tempo de Resposta (RSA) - Proxy x Load Balance

Método	AVG	ST	CI	UL	LL	Min	Max	90th pct	CCT	NCB	Broker	NBA	API
RSA Proxy													
Get Token	1785.03	938.17	26.00	1759.03	1811.04	88	5011	2948.70	0.37	6.77	8.34	91.29	1678.27
Get Data	111.05	238.90	6.62	104.42	117.67	2	3175	395.90	0.36	-0.49	2.22	93.53	15.45
Total	948.04	1081.30	21.19	926.85	969.23	2	5011	2549.80	0.37	3.14	5.28	92.41	846.86
RSA Load Balance													
Get Token	98.37	11.39	0.32	98.05	98.68	87	230	106	0.33	0.43	0.2	0.52	96.83
Get Data	4.20	1.38	0.04	4.16	4.24	2	30	5	0.32	0.58	0.21	0.56	2.5
Total	51.28	47.78	0.94	50.35	52.22	2	230	101	0.33	0.51	0.21	0.54	49.67

Figura 35 – Tempo de Resposta com RSA - 10 mil Proxy

Figura 36 – Tempo de Resposta com RSA - 10 mil Load balance

monitoramento teve por objetivo mensurar o consumo de recurso computacional que determinada configuração exerce. Deste modo, é possível auxiliar na melhora de uma determinada seleção de configuração visando um menor consumo computacional. Muitos dispositivos *IoT* possuem recursos computacionais limitados o que torna necessário configurações menos custosas sem deixar de lado um nível de segurança aceitável. Em cada gráfico é ilustrado o consumo das 10 execuções do experimento onde cada intervalo da parábola equivale a um conjunto de execuções. Em todas as configurações o consumo de Memória RAM é relativamente baixo para nosso provedor *OAuth2*. O método de assinatura simétrica e assimétrica possuem consumo de RAM próximos em todas as configurações variando em média de 18% à 25% do consumo de RAM. Quando o enfoque está para o consumo de CPU o comportamento possui uma variação considerável conforme a configuração adotada. A carga de trabalho maior, com 10 mil requisições apresenta um alto consumo de CPU tanto para o *HMAC* quanto para o *RSA*. Neste caso, quando

adotamos a inclusão de um balanceador de carga podemos perceber uma queda no consumo médio de CPU entre 40% e 50%.

Figura 37 – Monitoramento de CPU e RAM

As Figuras 38 e 39 apresentam a latência gasta quando a quantidade de requisição é alta. A latência da rede refere-se ao tempo gasto para um pacote de dados ir de um ponto a outro, ou seja, a latência da nossa arquitetura é medida no momento que o pacote de dados é criado nas máquinas clientes seguindo o modelo de referência TCP/IP e é enviado para o *broker* que por sua vez encaminha ao *OAuthServer* que faz o processamento necessário e retorna o pacote de dados novamente seguindo o caminho contrário até o Cliente que enviou o pacote. Podemos perceber que o atraso é alto quando temos apenas o provedor de identidade e o *broker* trabalha como *Proxy*. Para os dois métodos de assinatura de *token* em estudo, a latência é alta para muitas requisições. Quando o balanceador de carga é inserido, percebemos pela Figura 39 que a latência diminui substancialmente, isso complementa nosso resultado anterior que comprova a importância de usar de se utilizar o *broker* e a possibilidade de replicação do provedor de identidade.

Por meio da Figura 40 percebemos como os dados estão distribuídos, para 1 mil requisições tanto utilizando o *Proxy* quanto o balanceador de carga o comportamento da distribuição dos dados tende a ser mais uniforme. Porém, ao aumentarmos a quantidade de requisições para

Figura 38 – Análise da Latência ao longo do Tempo com Proxy

Figura 39 – Análise da Latência ao longo do Tempo com o Load Balance

10 mil, a configuração utilizando o *Proxy* apresenta valores relativamente altos que fogem ao comportamento das demais configurações em estudo. Esse comportamento tem sido evidenciado em nossas discussões dos resultados, pois o provedor de identidade fica sobrecarregado e não consegue responder rapidamente às requisições que chegam. O comportamento apresenta ainda mais caótico quando o algoritmo de assinatura é assimétrico. Contudo, para o mesmo cenário de teste com a adição do balanceador de carga a distribuição dos dados se tornam mais uniforme. Assim é possível perceber que o impacto que o *broker* como balanceador contribui para a diminuição do tempo de resposta final.

5.4.1.1 Conclusões do Experimento A

Conforme as avaliações realizadas, concluímos que o algoritmo assimétrico RSA possui um impacto significativo no tempo de resposta quando temos uma quantidade expressiva de requisições (em nosso exemplo, 10 mil requisições). Quando a quantidade de requisições não é relativamente alta o impacto no tempo médio de resposta final não significativo se comparado ao algoritmo simétrico HMAC. Assim, para aplicações que possuem uma carga de trabalho baixa (em nosso exemplo 1 mil requisições) recomendamos o uso do RSA uma vez que possui resultados bem próximos ao algoritmo simétrico, porém, com uma segurança maior. Para ambientes que requerem uma carga de trabalho elevada (em nosso exemplo 10 mil requisições) é importante a adição de um balanceador de carga que possibilitará uma queda no tempo médio de

Figura 40 – Análise da distribuição dos dados

resposta final. O balanceador é importante porque alivia o provedor de autenticação e autorização de ficar sobrecarregado dividindo as requisições entre mais servidores.

O impacto de utilizar uma criptografia assimétrica aumenta a medida que a quantidade de requisições cresce, em nossas análises evidenciamos uma queda considerável no tempo final com a adição do balanceador. Já para ambientes com baixa carga de trabalho, o balanceador é ineficaz e não contribui para a diminuição do tempo final. Para aplicações que demandam uma alta carga de trabalho e não possuem condições de inserir um balanceador de carga e mais provedores de autenticação e autorização, o algoritmo que melhor se adapta é o HMAC, pois causará o impacto substancialmente menor quando comparado ao modelo assimétrico RSA. Em termos de consumo de CPU e memória, podemos perceber que ambos algoritmos demandam maior uso de CPU e baixo consumo de memória sendo que para a configuração máxima de requisições ambos algoritmos utilizaram 100% de CPU. Contudo, com o acréscimo do balanceador de carga houve uma queda em torno de 35% a 45% do consumo total de CPU.

5.4.2 Desenho do Experimento B

O experimento B teve por objetivo avaliar o custo computacional no tempo de resposta final comparando a escrita e a leitura no banco de dados bem como a taxa de transferência de cada operação. A Tabela 14 apresenta os parâmetros fixos, a Tabela 15 define os parâmetros variáveis e a Tabela 16 apresenta todas as combinações geradas para o experimento a partir dos parâmetros variáveis. Neste teste definimos uma carga de trabalho de 1000 requisições tanto utilizando o *broker* e utilizando o balanceador de carga. Testamos os dois algoritmos que compõem todo o estudo desta dissertação (*HMAC e RSA*) e buscamos avaliar o comportamento

Tabela 14 – Valores fixos associados aos parâmetros do experimento

Parâmetros Fixos	
Distribuição estatística	Uniforme
Algoritmo Escalonador	Round Robin
Refresh Token	Sim
Total de Requisições	1000
Número de Clientes	10
Ramp-Up Period	10 segundos
Número de iterações por thread	5
Número de Replicação	10

Tabela 15 – Valores variáveis associados aos parâmetros do experimento

Parâmetros Variáveis	
Fatores	Níveis
Arquitetura	Proxy e Load Balance
Tipo de Requisição	Get Access Token e Get Data
Criptografia	Simétrica e Assimétrica
Operação	Leitura e Escrita

Tabela 16 – Combinação experimental

Combinação experimental					
Exp	D	C	B	A	Variável de Resposta
	Criptografia	Tipo de Requisição	Operação	Arquitetura	Tempo médio
1	Simétrico	Get Access Token	Leitura	Proxy	-
2	Simétrico	Get Access Token	Leitura	Load Balance	-
3	Simétrico	Get Access Token	Escrita	Proxy	-
4	Simétrico	Get Access Token	Escrita	Load Balance	-
5	Simétrico	Get Data	Leitura	Proxy	-
6	Simétrico	Get Data	Leitura	Load Balance	-
7	Simétrico	Create Data	Escrita	Proxy	-
8	Simétrico	Create Data	Escrita	Load Balance	-
9	Assimétrico	Get Access Token	Leitura	Proxy	-
10	Assimétrico	Get Access Token	Leitura	Load Balance	-
11	Assimétrico	Get Access Token	Escrita	Proxy	-
12	Assimétrico	Get Access Token	Escrita	Load Balance	-
13	Assimétrico	Get Data	Leitura	Proxy	-
14	Assimétrico	Get Data	Leitura	Load Balance	-
15	Assimétrico	Create Data	Escrita	Proxy	-
16	Assimétrico	Create Data	Escrita	Load Balance	-

tanto para recuperar um dado da base quanto para inserir um dado na base. O cenário de teste é o mesmo apresentado pela Figura 27 e selecionamos um planejamento fatorial 2k com quatro fatores e dois níveis para cada fator.

Os resultados deste teste apontam uma leve melhora do tempo de resposta quando utilizamos o *load balance* para o processo de escrita na base. As Figuras 41 e 42 apresentam uma comparação do tempo de resposta para ler e escrever um dado na base utilizando o modelo

de *Proxy* e *Load balance* respectivamente para uma quantidade de 1000 requisições.

Figura 41 – Leitura x Escrita com Proxy

Figura 42 – Leitura x Escrita com Load balance

É possível notar por meio da Figura 41 que quando adotamos o *broker* como *Proxy* o processo de geração do *token JWT* para o algoritmo *RSA* tende a ser maior que para o *HMAC*, resultado este já evidenciado no experimento A. No entanto, para o processo de leitura e escrita na base, os valores são muito próximos uns dos outros não tendo diferença significativa. Quando alteramos a configuração do *broker* para o balanceador de carga, a diferença continua relativamente baixa para o processo de aquisição do *token* para leitura em ambos os algoritmos. Neste caso é possível notar que a adição do balanceador não contribui significativamente para a diminuição do tempo de resposta. Porém, quando avaliamos o processo de aquisição do *token*

para escrita na base de dados é possível perceber uma melhora no tempo de resposta final quando utilizamos o balanceador de carga. A Figura 43 ilustra de uma forma conjunta o comportamento da carga de trabalho utilizando o *Proxy* e o balanceador de carga para ambos os métodos simétrico e assimétrico.

Figura 43 – Análise usando Proxy e Load Balance para Escrita

A Figura 44 ilustra o consumo de Memória RAM e CPU utilizando o *Proxy* e o *Load balance* em nosso provedor *OAuth2* para os dois algoritmos de assinatura em estudo focando na operação de escrita. Enquanto que nossa análise anterior do monitoramento teve enfoque em operações de leitura, a Figura 44 enfoca em operações de escrita. Percebemos que o consumo de RAM e CPU é maior quando utilizamos apenas o *Proxy*, isto é, sem replicação do nosso provedor *OAuth2*. Este resultado é esperado, visto que a adição do balanceador objetiva diminuir a carga de processamento correspondente a apenas um provedor. Neste caso, o desempenho tende a melhorar com a diminuição do tempo de resposta final ao mesmo tempo que diminui o consumo de CPU e RAM.

5.4.2.1 Conclusões do Experimento B

As conclusões do Experimento B foram bastante similares ao experimento A, no entanto, no experimento B utilizamos uma carga de trabalho pequena. O método de escrita possui um

Figura 44 – Monitoramento de CPU e RAM

gargalo maior no tempo final se comparado ao método de leitura. Contudo, esse gargalo é devido as características próprias do método de escrita e não está relacionado diretamente ao algoritmo de assinatura do token, embora o algoritmo escolhido possua um pequeno impacto na inserção e no tempo médio final da requisição considerando as requisições para buscar o token e as requisições de inserção na base de dados.

5.5 Considerações Finais

Neste Capítulo apresentamos os resultados finais deste projeto de mestrado. Detalhamos os resultados dos testes preliminares na avaliação de desempenho de dois algoritmos para assinatura de *tokens JSON (RSA e HMAC)*, demonstramos um desenho de experimento e os resultados das avaliações realizadas. De forma geral, as avaliações e conclusões se expandem para qualquer ambiente de aplicação. Logo, são conclusões gerais e não apenas específicas para o nosso ambiente de teste. Deste modo, percebemos dois fatores importantes na escolha de um determinado algoritmo de assinatura. A carga de trabalho e a existência de um balanceador de carga. Cargas muito altas evidenciaram um aumento significativo no tempo de resposta final com o método de assinatura assimétrico e também simétrico. Quando fazemos uso de um balanceador de carga há uma queda significativa tanto no tempo médio final de resposta quanto no consumo de CPU. Para aplicações cuja carga de trabalho não se mostra elevada o uso do balanceador de carga é dispensável, podendo inclusive optar pela assinatura assimétrica uma vez que os tempos se mostraram próximos. Todavia, para ambientes cuja carga de trabalho é elevada o uso de balanceadores de carga é indispensável.

CONCLUSÃO

Esta dissertação apresentou um estudo envolvendo controle de acesso para Internet das Coisas levando em consideração uma análise entre o *trade-off* de segurança e desempenho aplicado no contexto de dados sensíveis de saúde e além disso apresentamos um protótipo arquitetural que serviu como base para as avaliações realizadas. O controle de acesso envolve vários aspectos que precisam ser analisados. Nossa abordagem enfatizou uma análise que considera tanto a segurança quanto o desempenho e buscamos analisar quais agentes podem afetar o controle de acesso. Ao considerar os trabalhos relacionados, identificamos que a análise que foi conduzida neste trabalho é pouco explorada pela comunidade científica. Deste modo, buscamos avaliar dois métodos de assinatura de *tokens* (*HMAC* e *RSA*) que são amplamente utilizados e qual o impacto que estes algoritmos podem causar no tempo de resposta para ambientes que demandam uma boa segurança alinhado a um rápido tempo de resposta com variações da carga de trabalho. Assim, nosso objetivo visou avaliar o impacto e o custo computacional em adotar uma abordagem mais segura e outra menos segura para assinatura do *token JWT* em situações que demandam segurança e desempenho.

6.1 Publicações e Submissões

Os resultados obtidos com o desenvolvimento deste trabalho permitiu a seguinte publicação até a presente data: **MARTINS, W. S.**; ESTRELLA, J. C.; BRUSCHI, S. M.; AZEVEDO, LEONILDO J. M.; ANDREAZI, G. T.,. Performance evaluation for signing JSON tokens in access control for the cloud of things. In: IEEE Cloud Summit 2020 Oct 21- 22, 2020 Now a virtual conference v. 3(2020): Main Conference, 2020 (Publicação)

6.2 Contribuições e Limitações

A principal contribuição deste trabalho concentra-se na avaliação do *trade-off* envolvendo a segurança e o desempenho dos algoritmos simétrico *HMAC* e assimétrico *RSA*. Nossa contribuição foi centrada na seleção e avaliação destes algoritmos para a assinatura de *tokens JSON* voltados para o controle de acesso, pois queremos auxiliar que pesquisadores e desenvolvedores possam construir e selecionar os algoritmos de assinatura mais apropriados para suas aplicações levando em consideração o custo destes algoritmos no tempo de resposta final por meio de um estudo voltado para análise de desempenho. Nossa segunda contribuição refere-se a construção do protótipo IoT-SHASEC que serviu de base para as avaliações realizadas. O protótipo construído, possui módulos que podem ser escaláveis possibilitando a construção de vários cenários para avaliação. Inicialmente a IoT-SHASEC possui implementado dois métodos de assinatura de *tokens* sendo um método simétrico com o *HMAC* e um método assimétrico com *RSA*. Além disso nossa API de autenticação e autorização foi construída de forma que possa se adaptar a qualquer cenário que seja necessário o uso de autenticação e autorização baseada em *tokens JWT* com poucas alterações de código. Construímos uma arquitetura com módulos independentes e distribuídos com possibilidade de escalabilidade do *Broker* e do Provedor de Identidade.

Os resultados dos experimentos evidenciam que o balanceador de carga se torna importante para diminuição do tempo médio final de resposta quando a quantidade de requisições é relativamente alta (10 mil requisições) além de diminuir a quantidade de CPU utilizada em cerca de 40% à 50% conforme experimento A. Para os cenários com 1000 requisições a adição do balanceador de carga não obteve uma melhora ou piora significativa no tempo resposta. Os resultados ainda mostraram que a medida que a quantidade de requisições aumenta, o custo da criptografia assimétrica se torna maior o que acarreta no aumento do tempo de resposta. No entanto, para cenários cuja a quantidade de requisições é baixa pode ser interessante a escolha de uma segurança maior dado que o tempo de resposta entre o método simétrico e assimétrico foram próximos.

Entretanto, apesar das contribuições deste trabalho algumas limitações necessitam ser evidenciadas. Este trabalho focou na avaliação de desempenho dos métodos de assinatura simétrica (*HMAC*) e assimétrica (*RSA*). Devido aos testes de desempenho, utilizamos o *Apache JMeter* como Cliente para geração da carga de trabalho por ser um *software* construído para tal finalidade. Para um cenário final mais próximo da realidade é necessário que uma aplicação Cliente *Web* ou *Mobile* seja construída para interação com o usuário. Outra limitação de nosso trabalho está na variedade de algoritmos existentes sendo atualmente dois métodos. Por fim, outra limitação do nosso trabalho está na troca estática do *Broker* no funcionamento de *Proxy* e *Load balance*. Inicialmente é necessário escolher qual modelo utilizar de forma estática pois o sistema não altera dinamicamente conforme a quantidade de requisições aumentam.

6.3 Trabalhos futuros

Como perspectiva de trabalhos futuros focaremos em mitigar as limitações mencionadas na seção anterior e:

- Construir uma interface *Web* que interaja com o usuário e funcione como aplicação Cliente. A interface refere-se a possibilidade do usuário visualizar por meio de relatórios seus dados médicos ou resultados de medições de saúde, como por exemplo medições de diabetes que foi base de estudo neste projeto, bem como informações de pressão arterial, batimentos cardíacos ou qualquer outra medição passível de ser realizada;
- Inserir novos algoritmos de assinatura de *tokens* e avaliar novos cenários com variações de tamanho de chaves. Analisamos os algoritmos *HMAC* e *RSA* por serem métodos de assinatura simétrica e assimétrica e nossos trabalhos relacionados terem abordado estes algoritmos. No entanto, novas implementações são necessárias para avaliações mais completas. Além disso, este trabalho não apresentou avaliações que variassem o tamanho da chave. Novos experimentos neste contexto serão propostos;
- Avaliar novos cenários com alterações do algoritmo de escalonamento do *Broker*. Nossa análise se baseia no uso do Round Robin como algoritmo escalonador, no entanto, conforme a quantidade de requisições aumenta e novos provedores de serviço são inseridos é importante dividir a carga de trabalho para os provedores mais livres. Nos próximos trabalhos analisaremos quais escalonadores melhor contribuem para esta perspectiva;
- Separar o servidor de autorização do servidor de recursos que atualmente faz parte na mesma API, assim, podemos construir novos cenários de testes escalando apenas o servidor com a maior necessidade;
- Utilizar otimização multi-objetivo para selecionar um algoritmo de assinatura melhor apropriado levando em consideração métricas escolhidas pelo próprio usuário na interface.

REFERÊNCIAS

AAZAM, M.; HUH, E. N. Fog computing and smart gateway based communication for cloud of things. In: **2014 International Conference on Future Internet of Things and Cloud**. [S.l.: s.n.], 2014. p. 464–470. Citado na página [2](#).

AAZAM, M.; HUH, E.-N. Fog computing and smart gateway based communication for cloud of things. In: IEEE. **Future Internet of Things and Cloud (FiCloud), 2014 International Conference on**. [S.l.], 2014. p. 464–470. Citado na página [10](#).

AAZAM, M.; ST-HILAIRE, M.; LUNG, C. H.; LAMBADARIS, I. Pre-fog: Iot trace based probabilistic resource estimation at fog. In: **2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)**. [S.l.: s.n.], 2016. p. 12–17. Citado na página [2](#).

AEDO, I.; DÍAZ, P.; SANZ, D. An rbac model-based approach to specify the access policies of web-based emergency information systems. **International Journal of Intelligent Control and Systems**, v. 11, n. 4, p. 272–283, 2006. Citado na página [20](#).

AHMED, M.; CHOWDHURY, A.; AHMED, M.; RAFEE, M. M. H. *et al.* An advanced survey on cloud computing and state-of-the-art research issues. **IJCSI International Journal of Computer Science Issues**, v. 9, n. 1, p. 1694–0814, 2012. Citado nas páginas [11](#) e [12](#).

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M.; AYYASH, M. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE Communications Surveys Tutorials**, v. 17, n. 4, p. 2347–2376, Fourthquarter 2015. ISSN 1553-877X. Citado nas páginas [2](#), [8](#), [13](#), [14](#) e [15](#).

AMIR, M.; PILLAI, P.; HU, Y. F. Cascading permissions policy model for token-based access control in the web of things. In: **2014 International Conference on Future Internet of Things and Cloud**. [S.l.: s.n.], 2014. p. 238–245. Citado nas páginas [19](#), [20](#) e [22](#).

ANGGOROJATI, B.; MAHALLE, P. N.; PRASAD, N. R.; PRASAD, R. Capability-based access control delegation model on the federated iot network. In: IEEE. **Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on**. [S.l.], 2012. p. 604–608. Citado na página [22](#).

_____. Secure access control and authority delegation based on capability and context awareness for federated iot. **Internet of Things and M2M Communications**, River Publishers, p. 135, 2013. Citado na página [22](#).

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I. *et al.* A view of cloud computing. **Communications of the ACM**, ACM, v. 53, n. 4, p. 50–58, 2010. Citado na página [11](#).

ASCHOFF, R.; ROSENDO, D.; MACHADO, M.; SANTOS, A.; SADOK, D. A network access control solution combining orbac and sdn. In: **2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)**. [S.l.: s.n.], 2017. p. 483–489. Citado na página [21](#).

ATLAM, H.; ALENEZI, A.; ALHARTHI, A.; WALTERS, R.; WILLS, G. *et al.* Integration of cloud computing with internet of things: challenges and open issues. 2017. Citado na página 12.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Comput. Netw.**, Elsevier North-Holland, Inc., New York, NY, USA, v. 54, n. 15, p. 2787–2805, out. 2010. ISSN 1389-1286. Disponível em: <<http://dx.doi.org/10.1016/j.comnet.2010.05.010>>. Citado na página 1.

BABU, S. M.; LAKSHMI, A. J.; RAO, B. T. A study on cloud based internet of things: Clouddiot. In: IEEE. **Communication Technologies (GCCT), 2015 Global Conference on**. [S.l.], 2015. p. 60–65. Citado nas páginas 12 e 13.

BAINA, A.; LAAROUCHI, Y. Multilevel-orbac: Multi-level integrity management in organization based access control framework. In: **2012 International Conference on Multimedia Computing and Systems**. [S.l.: s.n.], 2012. p. 933–938. Citado na página 21.

BAKER, S. B.; XIANG, W.; ATKINSON, I. Internet of things for smart healthcare: Technologies, challenges, and opportunities. **IEEE Access**, v. 5, p. 26521–26544, 2017. Citado na página 9.

BALTE, A.; KASHID, A.; PATIL, B. Security issues in internet of things (iot): A survey. In: . [S.l.: s.n.], 2015. Citado na página 14.

_____. Security issues in internet of things (iot): A survey. In: . [S.l.: s.n.], 2015. Citado na página 18.

BARKA, E.; MATHEW, S. S.; ATIF, Y. Securing the web of things with role-based access control. In: SPRINGER. **International Conference on Codes, Cryptology, and Information Security**. [S.l.], 2015. p. 14–26. Citado na página 20.

BARROS, V. A. **Uma abordagem para a integração de diferentes fontes de dados provenientes de redes de sensores no contexto de internet das coisas**. Tese (Doutorado) — Universidade de São Paulo, 2019. Citado nas páginas 15, 8, 9 e 11.

BARRY, D. K. **Web services, service-oriented architectures, and cloud computing**. [S.l.]: Elsevier, 2003. Citado nas páginas 11 e 12.

BECKERT, S. **Empire of cotton: A global history**. [S.l.]: Vintage, 2015. Citado na página 1.

BELTRÁN, M.; CALVO, M.; GONZÁLEZ, S. Federated system-to-service authentication and authorization combining pufs and tokens. In: **2017 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)**. [S.l.: s.n.], 2017. p. 1–8. Citado nas páginas 4, 26, 41, 44 e 47.

BERNABE, J. B.; RAMOS, J. L. H.; GOMEZ, A. F. S. Taciot: multidimensional trust-aware access control system for the internet of things. **Soft Computing**, Springer, v. 20, n. 5, p. 1763–1779, 2016. Citado na página 22.

BORGIA, E. The internet of things vision: Key features, applications and open issues. **Computer Communications**, v. 54, p. 1 – 31, 2014. ISSN 0140-3664. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0140366414003168>>. Citado na página 7.

BOTTA, A.; DONATO, W. D.; PERSICO, V.; PESCAPÉ, A. Integration of cloud computing and internet of things: a survey. **Future Generation Computer Systems**, Elsevier, v. 56, p. 684–700, 2016. Citado nas páginas 12 e 13.

BOUIJ-PASQUIER, I.; KALAM, A. A. E.; OUAHMAN, A. A.; MONTFORT, M. D. A security framework for internet of things. In: SPRINGER. **International Conference on Cryptology and Network Security**. [S.l.], 2015. p. 19–31. Citado na página 21.

CARO, N. D.; COLITTI, W.; STEENHAUT, K.; MANGINO, G.; REALI, G. Comparison of two lightweight protocols for smartphone-based sensing. In: IEEE. **Communications and Vehicular Technology in the Benelux (SCVT), 2013 IEEE 20th Symposium on**. [S.l.], 2013. p. 1–6. Citado na página 15.

CARR, N. G. **The big switch: Rewiring the world, from Edison to Google**. [S.l.]: WW Norton & Company, 2008. Citado na página 1.

CENTURION, A. M. **Impacto das rajadas no desempenho de serviços executados em ambientes em nuvens**. Tese (Doutorado) — Universidade de São Paulo, 2015. Citado na página 36.

CHEN, N.; DAHANAYAKE, A. Role-based situation-aware information seeking and retrieval for crisis response. **Journal of Intelligent Control Systems**, v. 12, n. 2, p. 186–197, 2007. Citado na página 19.

CHHETRI, S. R.; RASHID, N.; FAEZI, S.; FARUQUE, M. A. A. Security trends and advances in manufacturing systems in the era of industry 4.0. In: **2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.: s.n.], 2017. p. 1039–1046. Citado na página 1.

CHOUDHARY, G.; JAIN, A. K. Internet of things: A survey on architecture, technologies, protocols and challenges. In: **2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)**. [S.l.: s.n.], 2016. p. 1–8. ISSN 5090-2806. Citado nas páginas 2, 13, 14 e 15.

CIRANI, S.; PICONE, M.; GONIZZI, P.; VELTRI, L.; FERRARI, G. Iot-oas: An oauth-based authorization service architecture for secure services in iot scenarios. **IEEE sensors journal**, IEEE, v. 15, n. 2, p. 1224–1234, 2015. Citado nas páginas 26 e 46.

DELVAUX, J.; PEETERS, R.; GU, D.; VERBAUWHEDE, I. A survey on lightweight entity authentication with strong pufs. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 48, n. 2, p. 26:1–26:42, out. 2015. ISSN 0360-0300. Disponível em: <<http://doi.acm.org/10.1145/2818186>>. Citado na página 44.

DENNIS, J. B.; HORN, E. C. V. Programming semantics for multiprogrammed computations. **Commun. ACM**, ACM, New York, NY, USA, v. 9, n. 3, p. 143–155, mar. 1966. ISSN 0001-0782. Disponível em: <<http://doi.acm.org/10.1145/365230.365252>>. Citado nas páginas 3 e 21.

DUNKELS, A.; ERIKSSON, J.; TSIFTES, N. Low-power interoperability for the ipv6-based internet of things. In: **Proceedings of the 10th Scandinavian Workshop on Wireless Ad-Hoc Networks (ADHOC'11), Stockholm, Sweden**. [S.l.: s.n.], 2011. p. 10–11. Citado na página 15.

DUNKELS, A.; MOTTOLA, L.; TSIFTES, N.; ÖSTERLIND, F.; ERIKSSON, J.; FINNE, N. The announcement layer: Beacon coordination for the sensornet stack. In: SPRINGER. **European Conference on Wireless Sensor Networks**. [S.l.], 2011. p. 211–226. Citado na página 15.

EL-AZIZ, A. A.; KANNAN, A. A comprehensive presentation to xacml. IET, 2013. Citado na página 23.

ELSALEH, T.; GLUHAK, A.; MOESSNER, K. Service continuity for subscribers of the mobile real world internet. In: IEEE. **Communications Workshops (ICC), 2011 IEEE International Conference On**. [S.l.], 2011. p. 1–5. Citado na página 14.

EMERSON, S.; CHOI, Y.-K.; HWANG, D.-Y.; KIM, K.-S.; KIM, K.-H. An oauth based authentication mechanism for iot networks. In: IEEE. **Information and Communication Technology Convergence (ICTC), 2015 International Conference on**. [S.l.], 2015. p. 1072–1074. Citado nas páginas 24, 41, 42 e 46.

EVANS, D. The internet of things: How the next evolution of the internet is changing everything. **CISCO white paper**, v. 1, n. 2011, p. 1–11, 2011. Citado na página 8.

FAROOQ, M.; WASEEM, M.; MAZHAR, S.; KHAIRI, A.; KAMAL, T. A review on internet of things (iot). **International Journal of Computer Applications**, Foundation of Computer Science, v. 113, n. 1, 2015. Citado na página 17.

FERNANDES, J. L.; LOPES, I. C.; RODRIGUES, J. J.; ULLAH, S. Performance evaluation of restful web services and amqp protocol. In: IEEE. **Ubiquitous and Future Networks (ICUFN), 2013 Fifth International Conference on**. [S.l.], 2013. p. 810–815. Citado na página 15.

FERNÁNDEZ, F.; ALONSO, Á.; MARCO, L.; SALVACHÚA, J. A model to enable application-scoped access control as a service for iot using oauth 2.0. In: **2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)**. [S.l.: s.n.], 2017. p. 322–324. Citado nas páginas 26, 41 e 45.

FERSI, G. Middleware for internet of things: A study. In: **2015 International Conference on Distributed Computing in Sensor Systems**. Institute of Electrical & Electronics Engineers (IEEE), 2015. Disponível em: <<http://dx.doi.org/10.1109/DCOSS.2015.43>>. Citado na página 10.

FONGEN, A. Identity management and integrity protection in the internet of things. In: IEEE. **Emerging Security Technologies (EST), 2012 Third International Conference on**. [S.l.], 2012. p. 111–114. Citado na página 17.

FREMANTLE, P.; AZIZ, B.; KOPECKÝ, J.; SCOTT, P. Federated identity and access management for the internet of things. In: IEEE. **Secure Internet of Things (SIoT), 2014 International Workshop on**. [S.l.], 2014. p. 10–17. Citado na página 26.

FU, H.-L.; LIN, P.; YUE, H.; HUANG, G.-M.; LEE, C.-P. Group mobility management for large-scale machine-to-machine mobile networking. **IEEE Transactions on Vehicular Technology**, IEEE, v. 63, n. 3, p. 1296–1305, 2014. Citado na página 14.

GANZ, F.; LI, R.; BARNAGHI, P.; HARAI, H. A resource mobility scheme for service-continuity in the internet of things. In: IEEE. **Green Computing and Communications (GreenCom), 2012 IEEE International Conference On**. [S.l.], 2012. p. 261–264. Citado na página 14.

GARTNER, G. S. **4.9 Billion Connected “Things” Will Be in Use in 2015** (2014). **Press Release**. 2016. Citado na página 8.

GERDES, S.; BERGMANN, O.; BORMANN, C. Delegated coap authentication and authorization framework (dcaf). **IETF draftgerdes-core-dcaf-authorize-02**, 2014. Citado na página 26.

GHORBANI, H. R.; AHMADZADEGAN, M. H. Security challenges in internet of things: survey. In: **2017 IEEE Conference on Wireless Sensors (ICWiSe)**. [S.l.: s.n.], 2017. p. 1–6. Citado na página 17.

GOMES, M.; RIGHI, R. da R.; COSTA, C. A. da. Internet of things scalability: Analyzing the bottlenecks and proposing alternatives. In: IEEE. **Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2014 6th International Congress on**. [S.l.], 2014. p. 269–276. Citado na página 15.

GONG, L. A secure identity-based capability system. In: IEEE. **Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on**. [S.l.], 1989. p. 56–63. Citado na página 22.

GOUD, E. A. K.; REDDY, M. J. A survey on open authorization (oauth). 2013. Citado na página 26.

GUOPING, Z.; WENTAO, G. The research of access control based on ucon in the internet of things. **Journal of Software**, Academy Publisher, PO Box 40 Oulu 90571 Finland, v. 6, n. 4, p. 724–731, 2011. Citado na página 21.

GUSMEROLI, S.; PICCIONE, S.; ROTONDI, D. A capability-based security approach to manage access control in the internet of things. **Mathematical and Computer Modelling**, v. 58, n. 5, p. 1189 – 1205, 2013. ISSN 0895-7177. The Measurement of Undesirable Outputs: Models Development and Empirical Analyses and Advances in mobile, ubiquitous and cognitive computing. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S089571771300054X>>. Citado nas páginas 3 e 19.

_____. A capability-based security approach to manage access control in the internet of things. **Mathematical and Computer Modelling**, Elsevier, v. 58, n. 5-6, p. 1189–1205, 2013. Citado na página 22.

HAMMER-LAHAV, E. The oauth 1.0 protocol. 2010. Citado na página 24.

HARDT, D. The oauth 2.0 authorization framework. 2012. Citado nas páginas 24, 42 e 45.

HEMDI, M.; DETERS, R. Using rest based protocol to enable abac within iot systems. In: IEEE. **Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual**. [S.l.], 2016. p. 1–7. Citado na página 26.

HERNÁNDEZ-RAMOS, J. L.; JARA, A. J.; MARIN, L.; SKARMETA, A. F. Distributed capability-based access control for the internet of things. **Journal of Internet Services and Information Security (JISIS)**, v. 3, n. 3/4, p. 1–16, 2013. Citado na página 22.

HERNÁNDEZ-RAMOS, J. L.; JARA, A. J.; MARÍN, L.; GÓMEZ, A. F. S. Dcapbac: embedding authorization logic into smart things through ecc optimizations. **International Journal of Computer Mathematics**, Taylor & Francis, v. 93, n. 2, p. 345–366, 2016. Citado nas páginas 22 e 23.

- HERNANDEZ-RAMOS, J. L.; PAWLOWSKI, M. P.; JARA, A. J.; SKARMETA, A. F.; LADID, L. Toward a lightweight authentication and authorization framework for smart objects. **IEEE Journal on Selected Areas in Communications**, IEEE, v. 33, n. 4, p. 690–702, 2015. Citado na página 26.
- HERREWEGE, A. V.; KATZENBEISSER, S.; MAES, R.; PEETERS, R.; SADEGHI, A.-R.; VERBAUWHEDE, I.; WACHSMANN, C. Reverse fuzzy extractors: Enabling lightweight mutual authentication for puf-enabled rfids. In: KEROMYTIS, A. D. (Ed.). **Financial Cryptography and Data Security**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 374–389. ISBN 978-3-642-32946-3. Citado na página 44.
- HINAI, S. A.; SINGH, A. V. Internet of things: Architecture, security challenges and solutions. In: **2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)**. [S.l.: s.n.], 2017. p. 1–4. Citado na página 10.
- HU, P.; DHELM, S.; NING, H.; QIU, T. Survey on fog computing: architecture, key technologies, applications and open issues. **Journal of Network and Computer Applications**, v. 98, n. Supplement C, p. 27 – 42, 2017. ISSN 1084-8045. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1084804517302953>>. Citado na página 2.
- ISHAQ, I.; CARELS, D.; TEKLEMARIAM, G. K.; HOEBEKE, J.; ABEELE, F. V. d.; POORTER, E. D.; MOERMAN, I.; DEMEESTER, P. Ietf standardization in the field of the internet of things (iot): a survey. **Journal of Sensor and Actuator Networks**, Multidisciplinary Digital Publishing Institute, v. 2, n. 2, p. 235–287, 2013. Citado na página 16.
- JAIN, R. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. [S.l.]: John Wiley & Sons, 1990. Citado nas páginas 31, 32, 48, 69 e 70.
- JARADAT, O.; SLJIVO, I.; HABLI, I.; HAWKINS, R. Challenges of safety assurance for industry 4.0. In: **2017 13th European Dependable Computing Conference (EDCC)**. [S.l.: s.n.], 2017. p. 103–106. Citado na página 1.
- JINDOU, J.; XIAOFENG, Q.; CHENG, C. Access control method for web of things based on role and sns. In: IEEE. **Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on**. [S.l.], 2012. p. 316–321. Citado na página 20.
- JONES, M. **Json web algorithms (jwa)**. 2015. Citado nas páginas 17, 29 e 30.
- _____. **Json web key (jwk)**. 2015. Citado na página 29.
- JONES, M.; BRADLEY, J.; SAKIMURA, N. **Json web token (jwt)**. [S.l.], 2015. Citado nas páginas 3, 27, 28 e 29.
- JONES, M.; HILDEBRAND, J. **Json web encryption (jwe)**. [S.l.], 2015. Citado nas páginas 28 e 29.
- JUNG, S. W.; JUNG, S. Personal oauth authorization server and push oauth for internet of things. **International Journal of Distributed Sensor Networks**, v. 13, n. 6, p. 1550147717712627, 2017. Citado nas páginas 41, 46 e 47.
- JUNIOR, M.; PIMENTA, H. **Ferramentas de avaliação de desempenho para servidores web: análise, implementação de melhorias e testes**. Tese (Doutorado) — Universidade de São Paulo, 2004. Citado nas páginas 35 e 36.

KALAM, A. A. E.; BAIDA, R. E.; BALBIANI, P.; BENFERHAT, S.; CUPPENS, F.; DESWARTE, Y.; MIEGE, A.; SAUREL, C.; TROUESSIN, G. Organization based access control. In: **Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks**. [S.l.: s.n.], 2003. p. 120–131. Citado na página 21.

KALAM, A. A. E.; DESWARTE, Y.; BAÏNA, A.; KAÂNICHE, M. Polyorbac: a security framework for critical infrastructures. **International Journal of Critical Infrastructure Protection**, Elsevier, v. 2, n. 4, p. 154–169, 2009. Citado na página 21.

KALAM, A. A. E.; HASSANI, A. A. E.; OUAHMAN, A. A. Integrity-orbac: An orbac enhancement that takes into account integrity. In: IEEE. **Intelligent Systems: Theories and Applications (SITA), 2013 8th International Conference on**. [S.l.], 2013. p. 1–7. Citado na página 21.

KESHAV, S. How to read a paper. **ACM SIGCOMM Computer Communication Review**, ACM, v. 37, n. 3, p. 83–84, 2007. Citado na página 40.

KESHAVARZI, A.; HAGHIGHAT, A. T.; BOHLOULI, M. Research challenges and prospective business impacts of cloud computing: A survey. In: **2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)**. [S.l.: s.n.], 2013. v. 02, p. 731–736. Citado nas páginas 11 e 12.

KHALED, A.; HUSAIN, M. F.; KHAN, L.; HAMLEN, K. W.; THURASINGHAM, B. A token-based access control system for rdf data in the clouds. In: IEEE. **Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on**. [S.l.], 2010. p. 104–111. Citado na página 22.

KHAN, R.; KHAN, S. U.; ZAHEER, R.; KHAN, S. Future internet: The internet of things architecture, possible applications and key challenges. In: **2012 10th International Conference on Frontiers of Information Technology**. Institute of Electrical & Electronics Engineers (IEEE), 2012. Disponível em: <<http://dx.doi.org/10.1109/FIT.2012.53>>. Citado na página 10.

KIEBITZ, F. Nikola tesla zum fünfundsiebzigsten geburtstage. **Naturwissenschaften**, Springer, v. 19, n. 31, p. 665–666, 1931. Citado na página 1.

KINIKAR, S.; TERDAL, S. Implementation of open authentication protocol for iot based application. In: IEEE. **Inventive Computation Technologies (ICICT), International Conference on**. [S.l.], 2016. v. 1, p. 1–4. Citado na página 26.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. Systematic literature reviews in software engineering—a systematic literature review. **Information and software technology**, Elsevier, v. 51, n. 1, p. 7–15, 2009. Citado na página 37.

_____. Systematic literature reviews in software engineering – a systematic literature review. **Information and Software Technology**, v. 51, n. 1, p. 7 – 15, 2009. ISSN 0950-5849. Special Section - Most Cited Articles in 2002 and Regular Research Papers. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0950584908001390>>. Citado na página 38.

KITCHENHAM, B.; PRETORIUS, R.; BUDGEN, D.; BRERETON, O. P.; TURNER, M.; NIAZI, M.; LINKMAN, S. Systematic literature reviews in software engineering – a tertiary

study. **Information and Software Technology**, v. 52, n. 8, p. 792 – 805, 2010. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584910000467>. Citado na página 38.

KORESHOFF, T. L.; ROBERTSON, T.; LEONG, T. W. Internet of things: A review of literature and products. In: **Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration**. New York, NY, USA: ACM, 2013. (OzCHI '13), p. 335–344. ISBN 978-1-4503-2525-7. Disponível em: <http://doi.acm.org/10.1145/2541016.2541048>. Citado nas páginas 2 e 7.

LAMBA, S.; SHARMA, M. An efficient elliptic curve digital signature algorithm (ecdsa). In: **2013 International Conference on Machine Intelligence and Research Advancement**. [S.l.: s.n.], 2013. p. 179–183. Citado na página 42.

LAZOUSKI, A.; MARTINELLI, F.; MORI, P. Usage control in computer security: A survey. **Computer Science Review**, Elsevier, v. 4, n. 2, p. 81–99, 2010. Citado na página 21.

LEE, I.; LEE, K. The internet of things (iot): Applications, investments, and challenges for enterprises. **Business Horizons**, v. 58, n. 4, p. 431 – 440, 2015. ISSN 0007-6813. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0007681315000373>. Citado na página 18.

LEE, S.-H.; HUANG, K.-W.; YANG, C.-S. Tbas: Token-based authorization service architecture in internet of things scenarios. **International Journal of Distributed Sensor Networks**, SAGE Publications Sage UK: London, England, v. 13, n. 7, p. 1550147717718496, 2017. Citado nas páginas 41 e 46.

LI, F.; VÖGLER, M.; CLAESSENS, M.; DUSTDAR, S. Efficient and scalable iot service delivery on cloud. In: IEEE. **Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on**. [S.l.], 2013. p. 740–747. Citado na página 15.

LI, L.; JIN, Z.; LI, G.; ZHENG, L.; WEI, Q. Modeling and analyzing the reliability and cost of service composition in the iot: A probabilistic approach. In: IEEE. **Web Services (ICWS), 2012 IEEE 19th International Conference on**. [S.l.], 2012. p. 584–591. Citado na página 14.

LIU, J.; XIAO, Y.; CHEN, C. P. Authentication and access control in the internet of things. In: IEEE. **Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on**. [S.l.], 2012. p. 588–592. Citado na página 20.

MAALEL, N.; NATALIZIO, E.; BOUABDALLAH, A.; ROUX, P.; KELLIL, M. Reliability for emergency applications in internet of things. In: IEEE. **Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference On**. [S.l.], 2013. p. 361–366. Citado na página 14.

MACEDO, D.; GUEDES, L. A.; SILVA, I. A dependability evaluation for internet of things incorporating redundancy aspects. In: IEEE. **Networking, Sensing and Control (ICNSC), 2014 IEEE 11th International Conference on**. [S.l.], 2014. p. 417–422. Citado na página 14.

MAES, R. Physically unclonable functions: Constructions, properties and applications (fysisch onkloonbare functies: constructies, eigenschappen en toepassingen). 2012. Citado na página 44.

MAHALLE, P. N.; ANGGOROJATI, B.; PRASAD, N. R.; PRASAD, R. *et al.* Identity authentication and capability based access control (iacac) for the internet of things. **Journal of Cyber Security and Mobility**, v. 1, n. 4, p. 309–348, 2013. Citado na página 22.

- MAHALLE, P. N.; THAKRE, P. A.; PRASAD, N. R.; PRASAD, R. A fuzzy approach to trust based access control in internet of things. In: IEEE. **Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), 2013 3rd International Conference on**. [S.l.], 2013. p. 1–5. Citado na página 22.
- MALER, E.; MISHRA, P.; PHILPOTT, R. Assertions and protocols for the oasis security assertion markup language (saml). **OASIS, September**, 2003. Citado na página 24.
- MANYIKA, J.; CHUI, M.; BUGHIN, J.; DOBBS, R.; BISSON, P.; MARRS, A. **Disruptive technologies: Advances that will transform life, business, and the global economy**. [S.l.]: McKinsey Global Institute San Francisco, CA, 2013. v. 180. Citado nas páginas 15, 9, 10 e 48.
- MARINESCU, D. C. **Cloud computing: theory and practice**. [S.l.]: Morgan Kaufmann, 2017. Citado na página 12.
- MARRA, A. L.; MARTINELLI, F.; MORI, P.; SARACINO, A. Implementing usage control in internet of things: A smart home use case. In: **2017 IEEE Trustcom/BigDataSE/ICSS**. [S.l.: s.n.], 2017. p. 1056–1063. Citado na página 24.
- MARTIN, K.; WANG, W. Aya: An efficient access-controlled storage and processing for cloud-based sensed data. In: **2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)**. [S.l.: s.n.], 2015. p. 130–134. Citado na página 47.
- MARTINS JÚLIO CEZAR ESTRELLA, S. M. B. L. J. d. M. d. A. G. T. A. Welington da S. Performance evaluation for signing json tokens in access control for the cloud of things. 2020. Citado na página 41.
- MENDEZ, D. M.; PAPAPANAGIOTOU, I.; YANG, B. Internet of things: Survey on security and privacy. **arXiv preprint arXiv:1707.01879**, 2017. Citado nas páginas 16 e 17.
- MIORANDI, D.; SICARI, S.; PELLEGRINI, F. D.; CHLAMTAC, I. Internet of things: Vision, applications and research challenges. **Ad hoc networks**, Elsevier, v. 10, n. 7, p. 1497–1516, 2012. Citado nas páginas 16 e 17.
- MISRA, S.; AGARWAL, P. Bio-inspired group mobility model for mobile ad hoc networks based on bird-flocking behavior. **Soft computing**, Springer, v. 16, n. 3, p. 437–450, 2012. Citado na página 14.
- NEISSE, R.; FOVINO, I. N.; BALDINI, G.; STAVROULAKI, V.; VLACHEAS, P.; GIAFFREDA, R. A model-based security toolkit for the internet of things. In: IEEE. **Availability, Reliability and Security (ARES), 2014 Ninth International Conference on**. [S.l.], 2014. p. 78–87. Citado na página 23.
- NEISSE, R.; STERI, G.; BALDINI, G. Enforcement of security policy rules for the internet of things. In: **2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)**. [S.l.: s.n.], 2014. p. 165–172. ISSN 2160-4886. Citado na página 18.
- NUNES, L. H. **Métodos de análise de decisão multicritério para a seleção de recursos em ambientes IoT**. Tese (PhD thesis) — Instituto de Ciências Matemáticas e de Computação ICMC/USP, 2018. Citado nas páginas 15, 2 e 11.

OKANO, M. T. Iot and industry 4.0: The industrial new revolution. In: **International Conference on Management and Information Systems September**. [S.l.: s.n.], 2017. v. 25, p. 26. Citado na página 1.

ORACEVIC, A.; DILEK, S.; OZDEMIR, S. Security in internet of things: A survey. In: **2017 International Symposium on Networks, Computers and Communications (ISNCC)**. [S.l.: s.n.], 2017. p. 1–6. Citado na página 1.

_____. Security in internet of things: A survey. In: **2017 International Symposium on Networks, Computers and Communications (ISNCC)**. [S.l.: s.n.], 2017. p. 1–6. Citado nas páginas 16 e 17.

ORLANDI, R. Ferramentas para análise de desempenho de sistemas computacionais distribuídos. São Carlos, 1995. Citado nas páginas 34 e 35.

OUADDAH, A.; BOUIJ-PASQUIER, I.; ELKALAM, A. A.; OUAHMAN, A. A. Security analysis and proposal of new access control model in the internet of thing. In: IEEE. **Electrical and Information Technologies (ICEIT), 2015 International Conference on**. [S.l.], 2015. p. 30–35. Citado na página 21.

OUADDAH, A.; MOUSANNIF, H.; ELKALAM, A. A.; OUAHMAN, A. A. Access control in the internet of things: Big challenges and new opportunities. **Computer Networks**, v. 112, n. Supplement C, p. 237 – 262, 2017. ISSN 1389-1286. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1389128616303735>>. Citado nas páginas 2, 3, 18, 19, 20, 21, 22, 23 e 24.

PARK, J. Usage control: A unified framework for next generation access control. George Mason University, 2003. Citado na página 20.

PARK, J.; JE, S.; JUNG, S.; JUNG, S. A secure patient information transfer method through delegated authorization. In: **2016 International Conference on Information and Communication Technology Convergence (ICTC)**. [S.l.: s.n.], 2016. p. 56–60. Citado nas páginas 41 e 43.

PATIDAR, S.; RANE, D.; JAIN, P. A survey paper on cloud computing. In: IEEE. **Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on**. [S.l.], 2012. p. 394–398. Citado na página 12.

PAWAR, A. B.; GHUMBRE, S. A survey on iot applications, security challenges and counter measures. In: **2016 International Conference on Computing, Analytics and Security Trends (CAST)**. [S.l.: s.n.], 2016. p. 294–299. Citado nas páginas 2 e 16.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: **EASE**. [S.l.: s.n.], 2008. v. 8, p. 68–77. Citado na página 37.

RABAI EI, K. A. A.; HAROUS, S. Internet of things: Applications and challenges. In: **2016 12th International Conference on Innovations in Information Technology (IIT)**. [S.l.: s.n.], 2016. p. 1–6. Citado na página 13.

RAZA, S.; DUQUENNOY, S.; HÖGLUND, J.; ROEDIG, U.; VOIGT, T. Secure communication for the internet of things—a comparison of link-layer security and ipsec for 6lowpan. **Security and Communication Networks**, Wiley Online Library, v. 7, n. 12, p. 2654–2668, 2014. Citado na página 15.

- REN, Z.; LIU, X.; YE, R.; ZHANG, T. Security and privacy on internet of things. In: **2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)**. [S.l.: s.n.], 2017. p. 140–144. Citado na página 17.
- RICHER, J.; SANSONO, A.; GLAZER, I. **OAuth 2 in Action**. [S.l.]: Manning Publications Shelter Island, 2017. Citado nas páginas 15 e 26.
- RISSANEN, E. *et al.* extensible access control markup language (xacml) version 3.0. **OASIS standard**, v. 22, 2013. Citado na página 23.
- SADEGHI, A.-R.; WACHSMANN, C.; WAIDNER, M. Security and privacy challenges in industrial internet of things. In: **ACM. Proceedings of the 52nd annual design automation conference**. [S.l.], 2015. p. 54. Citado na página 17.
- SAIN, M.; KANG, Y. J.; LEE, H. J. Survey on security in internet of things: State of the art and challenges. In: **2017 19th International Conference on Advanced Communication Technology (ICACT)**. [S.l.: s.n.], 2017. p. 699–704. Citado na página 17.
- SANTANA, R.; SANTANA, M.; ORLANDI, R.; SPOLON, R.; CALÔNEGO, N. Técnicas para avaliação de desempenho de sistemas computacionais. **Notas didáticas do ICMC, ICMC-USP**, 1994. Citado nas páginas 15, 33, 34 e 35.
- SARKAR, C.; NAMBI, S. A. U.; PRASAD, R. V.; RAHIM, A. A scalable distributed architecture towards unifying iot applications. In: **IEEE. Internet of Things (WF-IoT), 2014 IEEE World Forum on**. [S.l.], 2014. p. 508–513. Citado na página 15.
- SCHMIDT, R.; MÖHRING, M.; HÄRTING, R.-C.; REICHSTEIN, C.; NEUMAIER, P.; JOZINOVIC, P. Industry 4.0 - potentials for creating smart products: Empirical research results. In: _____. **Business Information Systems: 18th International Conference, BIS 2015, Poznań, Poland, June 24-26, 2015, Proceedings**. Cham: Springer International Publishing, 2015. p. 16–27. Citado na página 1.
- SCIANCELEPORE, S.; PIRO, G.; CALDAROLA, D.; BOGGIA, G.; BIANCHI, G. Oauth-iot: An access control framework for the internet of things based on open standards. In: **2017 IEEE Symposium on Computers and Communications (ISCC)**. [S.l.: s.n.], 2017. p. 676–681. Citado nas páginas 24, 26, 41 e 45.
- SEITZ, L.; SELANDER, G.; GEHRMANN, C. Authorization framework for the internet-of-things. In: **IEEE. World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a**. [S.l.], 2013. p. 1–6. Citado na página 23.
- SEITZ, L.; SELANDER, G.; WAHLSTROEM, E.; ERDTMAN, S.; TSCHOFENIG, H. Authentication and authorization for constrained environments (ace). **ACE Working Group Internet-Draft**, 2016. Citado na página 26.
- SHI, F.; LI, Q.; ZHU, T.; NING, H. A survey of data semantization in internet of things. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 18, n. 1, p. 313, 2018. Citado na página 8.
- SICARI, S.; RIZZARDI, A.; GRIECO, L.; COEN-PORISINI, A. Security, privacy and trust in internet of things: The road ahead. **Computer Networks**, v. 76, p. 146 – 164, 2015. ISSN 1389-1286. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1389128614003971>>. Citado na página 18.

SIEKKINEN, M.; HIIENKARI, M.; NURMINEN, J. K.; NIEMINEN, J. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In: IEEE. **Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE**. [S.l.], 2012. p. 232–237. Citado na página 15.

SIEMENS, W. Ueber telegraphische leitungen und apparate. In: **Wissenschaftliche und Technische Arbeiten**. [S.l.]: Springer, 1889. p. 15–29. Citado na página 1.

SILVA, I.; LEANDRO, R.; MACEDO, D.; GUEDES, L. A. A dependability evaluation tool for the internet of things. **Computers & Electrical Engineering**, Elsevier, v. 39, n. 7, p. 2005–2018, 2013. Citado na página 14.

SILVA, N. B.; PIGATTO, D. F.; MARTINS, P. S.; BRANCO, K. R. Case studies of performance evaluation of cryptographic algorithms for an embedded system and a general purpose computer. **Journal of Network and Computer Applications**, Elsevier, v. 60, p. 130–143, 2016. Citado na página 33.

SINGH, D. Developing an architecture: Scalability, mobility, control, and isolation on future internet services. In: IEEE. **Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on**. [S.l.], 2013. p. 1873–1877. Citado na página 15.

SKARMETA, A. F.; HERNANDEZ-RAMOS, J. L.; MORENO, M. V. A decentralized approach for security and privacy challenges in the internet of things. In: IEEE. **Internet of Things (WF-IoT), 2014 IEEE World Forum on**. [S.l.], 2014. p. 67–72. Citado nas páginas 2, 16, 41 e 47.

SOLAPURKAR, P. Building secure healthcare services using oauth 2.0 and json web token in iot cloud scenario. In: **2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)**. [S.l.: s.n.], 2016. p. 99–104. Citado nas páginas 41, 42 e 47.

SU, X.; RIEKKI, J.; NURMINEN, J. K.; NIEMINEN, J.; KOSKIMIES, M. Adding semantics to internet of things. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 27, n. 8, p. 1844–1860, 2015. Citado na página 7.

SURESH, P.; DANIEL, J. V.; PARTHASARATHY, V.; ASWATHY, R. H. A state of the art review on the internet of things (iot) history, technology and fields of deployment. In: **2014 International Conference on Science Engineering and Management Research (ICSEMR)**. [S.l.: s.n.], 2014. p. 1–8. Citado na página 17.

THANGAVEL, D.; MA, X.; VALERA, A.; TAN, H.-X.; TAN, C. K.-Y. Performance evaluation of mqtt and coap via a common middleware. In: IEEE. **Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on**. [S.l.], 2014. p. 1–6. Citado na página 15.

TRNKA, M.; CERNY, T. Identity management of devices in internet of things environment. In: **2016 6th International Conference on IT Convergence and Security (ICITCS)**. [S.l.: s.n.], 2016. p. 1–4. Citado nas páginas 26, 41 e 43.

TSIATSIS, V.; GLUHAK, A.; BAUGE, T.; MONTAGUT, F.; BERNAT, J.; BAUER, M.; VILLALONGA, C.; BARNAGHI, P.; KRCO, S. The sensei real world internet architecture. Ios Pr Inc, 2010. Citado na página 22.

- UCKELMANN, D. Performance measurement and cost benefit analysis for rfid and internet of things implementations in logistics. In: **Quantifying the Value of RFID and the EPCglobal Architecture Framework in Logistics**. [S.l.]: Springer, 2012. p. 71–100. Citado na página 16.
- UCKELMANN, D.; ISENBERG, M.-A.; TEUCKE, M.; HALFAR, H.; SCHOLZ-REITER, B. Autonomous control and the internet of things: Increasing robustness, scalability and agility in logistic networks. **Unique radio innovation for the 21st century**, Springer, p. 163–181, 2010. Citado na página 15.
- WHITE, G.; NALLUR, V.; CLARKE, S. Quality of service approaches in iot: A systematic mapping. **Journal of Systems and Software**, v. 132, p. 186 – 203, 2017. ISSN 0164-1212. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S016412121730105X>>. Citado nas páginas 15 e 40.
- WHITMORE, A.; AGARWAL, A.; XU, L. D. The internet of things—a survey of topics and trends. **Information Systems Frontiers**, v. 17, n. 2, p. 261–274, Apr 2015. ISSN 1572-9419. Disponível em: <<https://doi.org/10.1007/s10796-014-9489-2>>. Citado na página 8.
- WITCHALLS, C.; CHAMBERS, J. The internet of things business index: A quiet revolution gathers pace. **The Economist Intelligence Unit**. Retrieved from <http://www.economistinsights.com/analysis/internet-things-business-index>, 2013. (Accessed on 13/05/2017). Citado nas páginas 15, 2, 8 e 9.
- WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: **Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering**. New York, NY, USA: ACM, 2014. (EASE '14), p. 38:1–38:10. ISBN 978-1-4503-2476-2. Disponível em: <<http://doi.acm.org/10.1145/2601248.2601268>>. Citado na página 38.
- YAN, Z.; ZHANG, P.; VASILAKOS, A. V. A survey on trust management for internet of things. **Journal of Network and Computer Applications**, v. 42, p. 120 – 134, 2014. ISSN 1084-8045. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1084804514000575>>. Citado na página 18.
- YE, N.; ZHU, Y.; WANG, R.-c.; MALEKIAN, R.; QIAO-MIN, L. An efficient authentication and access control scheme for perception layer of internet of things. **Applied Mathematics & Information Sciences**, Natural Sciences Publishing Corp, v. 8, n. 4, p. 1617, 2014. Citado na página 20.
- ZHANG, G.; TIAN, J. An extended role based access control model for the internet of things. In: IEEE. **Information Networking and Automation (ICINA), 2010 International Conference on**. [S.l.], 2010. v. 1, p. V1–319. Citado na página 20.
- ZHANG, S.; YAN, H.; CHEN, X. Research on key technologies of cloud computing. **Physics Procedia**, Elsevier, v. 33, p. 1791–1797, 2012. Citado na página 11.
- ZHANG, X.; PARISI-PRESICCE, F.; SANDHU, R.; PARK, J. Formal model and policy specification of usage control. **ACM Transactions on Information and System Security (TISSEC)**, ACM, v. 8, n. 4, p. 351–387, 2005. Citado na página 20.
- ZHOU, J.; LEPPANEN, T.; HARJULA, E.; YLIANTTILA, M.; OJALA, T.; YU, C.; JIN, H.; YANG, L. T. Cloudthings: A common architecture for integrating the internet of things with

cloud computing. In: IEEE. **Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on**. [S.l.], 2013. p. 651–657. Citado na página [12](#).

ZHU, Z.; ZHANG, L.; WAKIKAWA, R. Supporting mobility for internet cars. **IEEE Communications Magazine**, IEEE, v. 49, n. 5, 2011. Citado na página [14](#).