UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Tecnologia

**Natascha Sander de Abreu**

# Viabilidade de implementação de algoritmos criptográficos em dispositivos para Internet das Coisas

Limeira
2023

**Natascha Sander de Abreu**

# Viabilidade de implementação de algoritmos criptográficos em dispositivos para Internet das Coisas

Dissertação apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestra em Tecnologia, na área de Sistemas de Informação e Comunicação.

**Orientador: Prof. Dr. André Leon Sampaio Gradvohl**

Este trabalho corresponde à versão final da Dissertação defendida por Natascha Sander de Abreu e orientada pelo Prof. Dr. André Leon Sampaio Gradvohl.

Limeira
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

Ab86v

Abreu, Natascha Sander de, 1997-
Viabilidade de implementação de algoritmos criptográficos em dispositivos para internet das coisas / Natascha Sander de Abreu. – Limeira, SP : [s.n.], 2023.

Orientador: André Leon Sampaio Gradvohl.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Criptografia. 2. Internet das coisas. I. Gradvohl, André Leon Sampaio, 1973-. II. Universidade Estadual de Campinas. Faculdade de Tecnologia. III. Título.

Informações Complementares

**Título em outro idioma:** Feasibility of implementing cryptographic algorithms in internet of things devices
**Palavras-chave em inglês:**
Cryptography
Internet of things
**Área de concentração:** Sistemas de Informação e Comunicação
**Titulação:** Mestra em Tecnologia
**Banca examinadora:**
André Leon Sampaio Gradvohl [Orientador]
Carlos Eduardo Câmara
Vitor Augusto Galhardo Moia
**Data de defesa:** 27-02-2023
**Programa de Pós-Graduação:** Tecnologia

**Identificação e informações acadêmicas do(a) aluno(a)**
- ORCID do autor: https://orcid.org/0000-0003-4226-1244
- Currículo Lattes do autor: http://lattes.cnpq.br/5976072039972930

# FOLHA DE APROVAÇÃO

Abaixo se apresentam os membros da comissão julgadora da sessão pública de defesa de dissertação para o Título de Mestra em Tecnologia na área de concentração Sistemas de Informação e Comunicação, a que se submeteu a aluna Natascha Sander de Abreu, em 27 de fevereiro de 2023 na Faculdade de Tecnologia – FT/UNICAMP, em Limeira/SP.

**Prof. Dr. André Leon Sampaio Gradvohl**
Presidente da Comissão Julgadora

**Prof. Dr. Carlos Eduardo Câmara**
Pesquisador Independente

**Dr. Vitor Hugo Galhardo Moia**
Instituto de Pesquisas Eldorado

Ata da defesa, assinada pelos membros da Comissão Examinadora, encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós Graduação da Faculdade de Tecnologia.

A Deus, pela vida.
Aos meus avós, pelo incentivo, ensino e segurança.

*The task is... not so much to see what no one has yet seen; but to think what nobody has yet thought, about that which everybody sees.*

Erwin Schrödinger

# Agradecimentos

Primeiramente, agradeço a Deus pela vida, por me proporcionar saúde e determinação para não desanimar. Por guiar minhas escolhas e por tudo que conquistei até agora, obrigada Deus.

Agradeço à minha família, minha mãe Maria Goreti, meu pai Virgilio e meu irmão Murilo, por serem minha base, onde sempre encontrei todo apoio e auxílio para a realização dos meus sonhos. Agradeço aos meus avós por sempre aspirar os melhores planos para minha formação e pelo incentivo a educação.

Ao meu orientador, prof. André, obrigada pelo apoio, incentivo, paciência e dedicação. Agradeço pelo imenso conhecimento compartilhado, correções e contribuições. Pela confiança e esperança depositadas no decorrer do trabalho.

Agradeço ao membros da banca de qualificação e defesa pelos comentários e pela avaliação deste trabalho. Estendo os agradecimentos a todos os professores que participaram da minha formação até aqui. Agradeço aos meus professores e amigos da Universidade de Passo Fundo e do Núcleo de Eletrônica por todo apoio e incentivo.

Aos meus amigos do Rotaract Club de Passo Fundo Planalto Médio Universitário, aos colegas e amigos do mestrado, obrigada pela amizade e companheirismo que tornaram esta etapa mais leve e divertida. Agradeço às minhas amigas, que sempre estiveram ao meu lado, pela escuta e compreensão em momentos difíceis.

Agradeço a todos que direta ou indiretamente fizeram parte desta conquista.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

# Resumo

A tendência do crescimento de dispositivos autônomos conectados à Internet avança diariamente. A chamada internet das coisas ou IoT, é uma tecnologia emergente e em evolução que deve conectar bilhões de dispositivos, viabilizando a transmissão de dados entre dispositivos inteligentes. Para o uso desta tecnologia é indispensável que se considere a implementação da segurança da informação nesses dispositivos. A segurança deve evitar ataques e vazamento de informações, construindo a confiança entre os dispositivos e seus usuários. Um método necessário para atingir a segurança em dispositivos para IoT são os algoritmos criptográficos. No entanto, os algoritmos criptográficos são, na sua maioria, caros computacionalmente. A limitação no poder de processamento e memória para IoT, estão entre as principais razões para ausência da implementação de uma comunicação criptografada nesses dispositivos. O objetivo desse trabalho é apresentar um estudo sobre os algoritmos criptográficos como solução para os problemas na troca de informações. Para isso, analisou-se, teórica e experimentalmente, algumas das soluções em dispositivos para IoT, usando o simulador do Contik OS, Cooja, para simular o ambiente IoT. A análise envolveu a comparação entre as respostas das métricas referentes ao consumo de energia para a execução dos algoritmos AES, Present, Simon e Speck. Dentre os resultados ensaiados nos tempos 10 minutos, 60 minutos e 120 minutos, o consumo de energia obedeceu um padrão mais estável a partir dos 60 minutos em todos os casos. Independente da escolha do algoritmo de criptografia, visto que o consumo de energia deles é praticamente o mesmo, o que se recomenda é adotar um algoritmo que se encaixe ao armazenamento disponível do dispositivo IoT e que seja testado e validado por entidades de padronização.

# Abstract

The growth trend of stand-alone devices connected to the Internet advances daily. The Internet of things, or IoT, is an emerging and evolving technology that should connect billions of devices, enabling data transmission among intelligent devices. For this technology, it is essential to consider information security in these devices. Security must prevent attacks and information leaks, building trust in devices and their users. A necessary method to achieve security in IoT devices is cryptographic algorithms. However, cryptographic algorithms are, for the most part, computationally expensive. The limitations in power processing and memory for IoT are among the main reasons for the lack of implementation of encrypted communication in these devices. This work aims to present a study on cryptographic algorithms as a solution to the problems in information exchange. Therefore, it was analyzed, theoretically and experimentally, some solutions in devices for IoT. Using the Contik OS simulator, Cooja, to simulate the IoT environment. The analysis involves comparing the responses of the metrics referring to energy consumption for the execution of the AES, Present, Simon, and Speck algorithms. Among the results tested at 10 minutes, 60 minutes, and 120 minutes, energy consumption followed a more stable pattern from minute 60 in all cases. Regardless of the choice of encryption algorithm, since their energy consumption is practically the same, we recommend adopting an algorithm that fits the available storage of the IoT device and is tested and validated by standardization entities.

# Lista de Figuras


1.1 Aplicações da IoT em cidades inteligentes . . . . . 16
1.2 Incidentes reportados ao CERT.br . . . . . 18
1.3 Origem dos incidentes reportados ao CERT.br . . . . . 19

2.1 Algoritmo de Euclides . . . . . 26
2.2 Cifra de César . . . . . 27
2.3 Enigma. . . . . 28
2.4 Modelo genérico de encriptação simétrica . . . . . 31
2.5 Processo de encriptação do AES . . . . . 34
2.6 Modelo de uso de encriptação com criptografia assimétrica . . . . . 35
2.7 Modelo de uso de encriptação com criptografia assimétrica . . . . . 36
2.8 Relação entre segurança, custo e desempenho na criptografia leve . . . . . 40
2.9 Primitivas criptográficas leves para IoT . . . . . 41
2.10 Impactos da IoT . . . . . 44
2.11 Arquitetura IoT . . . . . 45

4.1 Algoritmo Present . . . . . 59
4.2 Pseudocódigo Simon . . . . . 60
4.3 Pseudocódigo Speck . . . . . 61
4.4 Interface do Simulador Cooja . . . . . 62
4.5 Disposição dos sensores . . . . . 64
4.6 Métricas de desempenho . . . . . 64

5.1 Energia consumida na execução do algoritmo AES. . . . . 68
5.2 Energia consumida na execução do algoritmo Present. . . . . 70
5.3 Energia consumida na execução do algoritmo Simon. . . . . 72
5.4 Energia consumida na execução do algoritmo Speck. . . . . 74

A.1 Número de publicações encontradas por ano nas bases de dados. . . . . 86
A.2 Revisão sistemática . . . . . 88

# Lista de Tabelas


2.1 Terminologia comum para a análise de complexidade dos algoritmos. . . . . 27
2.2 Aplicações de criptossistemas de chave pública . . . . . 38
2.3 Características dos métodos para encriptação simétrica e assimétrica . . . . . 43
2.4 Comparação dos algoritmos. . . . . 44
2.5 Comparação dos kits pré montados de processadores. . . . . 46
2.6 Comparação das tecnologias de acesso . . . . . 47

3.1 Resumo principais algoritmos abordados nos trabalhos. . . . . 57

4.1 Plataformas Cooja . . . . . 63

5.1 Consumo de energia no algoritmo AES – minuto 10. . . . . 67
5.2 Consumo de energia no algoritmo AES – minuto 60. . . . . 67
5.3 Consumo de energia no algoritmo AES – minuto 120. . . . . 68
5.4 Consumo de energia no algoritmo Present – minuto 10. . . . . 69
5.5 Consumo de energia no algoritmo Present – minuto 60 . . . . . 69
5.6 Consumo de energia no algoritmo Present – minuto 120. . . . . 70
5.7 Consumo de energia no algoritmo Simon – minuto 10. . . . . 71
5.8 Consumo de energia no algoritmo Simon – minuto 60. . . . . 71
5.9 Consumo de energia no algoritmo Simon – minuto 120. . . . . 72
5.10 Consumo de energia no algoritmo Speck – minuto 10. . . . . 73
5.11 Consumo de energia no algoritmo Speck – minuto 60 . . . . . 73
5.12 Métricas de consumo de energia no algoritmo Speck. . . . . 74
5.13 Métricas de consumo de energia no algoritmo AES. . . . . 75
5.14 Métricas de consumo de energia no algoritmo Present. . . . . 75
5.15 Métricas de consumo de energia no algoritmo Simon. . . . . 75
5.16 Métricas de consumo de energia no algoritmo Speck . . . . . 75

A.1 Exemplo de estratégia de pesquisa para Sopus e número de arquivos encontrados. 86
A.2 Publicações e seus respectivos pontos-chave. . . . . 89

# Lista de Abreviaturas e Siglas

| | |
|---|---|
| 3DES | *Triple DES* |
| AES | *Advanced Encryption Standard* |
| BAN | *Body Area Networks* |
| BLE | *Bluetooth Low Energy* |
| CERT.br | Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil |
| CIA | *Confidentiality, Integrity, Availability* |
| dBm | decibel-miliwatt |
| DES | *Data Encryption Standard* |
| DREDS | *Dynamic Reconfigurable Encryption and Decryption System* |
| DSS | *Digital Signature Standard* |
| ECC | *Elliptic Curve Cryptography* |
| ECDH | Curva Elíptica de Diffie-Hellman |
| FPGA | *Field Programmable Gate Array* |
| IEC | *International Electrotechnical Commission* |
| IoT | Internet das Coisas |
| ISO | *International Organization for Standardization* |
| LED | *Light Encryption Device* |
| LWBC | *Lightweight Block Cipher* |
| LWC | *Lightweigtht Cryptography* |
| LWHF | *Lightweight Hash Functions* |
| LWSC | *Lightweight Stream Ciphers* |
| mdc | máximo divisor comum |
| NIST | *National Institute of Standards and Technology* |

| | |
|---|---|
| OTPs | *One-Time Pads* |
| OWASP | Open Web Application Security Project |
| RSA | Rivest-Shamir-Adleman |
| S-box | caixa de substituição |
| UDP | *User Datagram Protocol* |

# Sumário


**1 Introdução** **16**
1.1 Segurança da informação . . . 17
1.2 Objetivos . . . 20
1.3 Estrutura do texto . . . 20

**2 Referencial teórico** **22**
2.1 Fundamentos matemáticos para a criptografia . . . 22
2.1.1 A operação de divisão . . . 22
2.1.2 Aritmética modular . . . 23
2.1.3 Números primos . . . 24
2.1.4 Máximo divisor comum e o Algoritmo de Euclides . . . 25
2.1.5 Complexidade dos Algoritmos . . . 25
2.2 Criptografia . . . 26
2.2.1 Criptografia Simétrica . . . 30
2.2.2 Criptografia Assimétrica . . . 33
2.2.3 Criptografia Leve . . . 39
2.2.4 Comparação entre algoritmos simétricos e assimétricos . . . 42
2.3 Internet das Coisas . . . 43
2.3.1 Camada de rede . . . 45
2.3.2 Plataforma . . . 46
2.3.3 Camada de aplicação . . . 46
2.4 Considerações finais . . . 47

**3 Levantamento bibliográfico** **49**
3.1 Pesquisas precursoras . . . 49
3.2 Estado da arte . . . 52
3.2.1 Algoritmos genéricos para IoT . . . 52
3.2.2 Algoritmos de criptografia leve . . . 53
3.3 Considerações finais . . . 57

**4 Materiais e Métodos** **58**
4.1 Algoritmos . . . 58
4.1.1 Present . . . 59
4.1.2 Simon e Speck . . . 59
4.2 Sobre o simulador . . . 61
4.3 Critérios de avaliação dos algoritmos . . . 63
4.3.1 Potência e Energia . . . 64
4.4 Considerações finais . . . 65

**5 Resultados e discussões** **66**
5.1 AES . . . 66
5.2 Present . . . 69
5.3 Simon . . . 71
5.4 Speck . . . 73
5.5 Considerações gerais . . . 75

**6 Conclusões** **77**

**Referências bibliográficas** **79**

**A Método PRISMA para revisão bibliográfica** **84**
A.1 Definição do tópico de pesquisa . . . 84
A.2 Definição dos conceitos envolvidos . . . 84
A.3 Seleção das fontes bibliográficas . . . 85
A.4 Definição dos termos de busca . . . 85
A.5 Definição da estratégia de pesquisa . . . 85
A.6 Critério de Seleção . . . 87
A.7 Fluxograma . . . 87
A.8 Resultados . . . 87

# Capítulo 1

# Introdução

Os avanços tecnológicos dos últimos anos permitiram a implementação de tecnologias como a Internet das Coisas (IoT) (KHAN, M. N.; RAO; CAMTEPE, 2020). Essa nova tecnologia estende o alcance da Internet para objetos físicos, com capacidade de sensoriamento, atuação, computação, armazenamento e comunicação. A Figura 1.1 ilustra algumas aplicações da IoT nas chamadas cidades inteligentes.

Figura 1.1: Aplicações da IoT em cidades inteligentes. Adaptado de Pindar et al. (2020).

A IoT facilita o conforto, pois através de um *smartphone* é possível acionar outros dispositivos remotos como câmeras inteligentes, sensores de presença, controles de iluminação e até mesmo controlar a temperatura por meio da climatização inteligente. A IoT está em dispositivos médicos, como um marca-passo, com capacidade computacional de integrar-se à Internet e notificar um centro médico quando detecta algo de errado com o paciente (MORAES; HAYASHI, 2021).

Entre as estimativas citadas por Kane et al. (2020), tem-se que, em 2011, o número de dispositivos conectados à Internet ultrapassou a população mundial e até 2022 haveria aproximadamente 29 bilhões de dispositivos habilitados para a Internet. Desses, 18 bilhões seriam dispositivos IoT (KHAN, M. N.; RAO; CAMTEPE, 2020). Em uma estimativa mais recente, Moraes e Hayashi (2021) aponta que o número de dispositivos IoT deve chegar a 75 bilhões até 2025.

Todo sistema IoT é projetado para fornecer um serviço específico, que depende da coleta e troca de informações. A maioria dos dispositivos IoT são implantados de forma aberta, isto é, utilizando protocolos amplamente conhecidos pela comunidade. Além disso, frequentemente utilizam rede sem fio para transmissão, o que torna esses dispositivos suscetíveis a ataques à segurança da informação (DHANDA; SINGH, B.; JINDAL, 2020).

Na medida em que a Internet se torna parte da vida humana, as superfícies de ataque e os potenciais vetores de ataque aumentam (KHAN, M. N.; RAO; CAMTEPE, 2020). Com primitivas criptográficas, alguns dos componentes fundamentais da segurança da IoT, ou seja, o gerenciamento de identidade, a autenticação, a autorização, a troca e o gerenciamento de chaves e a confiança, podem ser implementados (DHANDA; SINGH, B.; JINDAL, 2020).

Por outro lado, organizações como a Open Web Application Security Project (OWASP)[1] identificaram como as principais vulnerabilidades do IoT a falta de privacidade, os protocolos de autenticação e autorização insuficientes e a ausência de criptografia na comunicação dos dados (DHANDA; SINGH, B.; JINDAL, 2020).

A segurança e a privacidade que os dispositivos IoT dão ao usuário, junto com o equilíbrio exigido entre segurança e o desempenho de dispositivos de baixa potência com recursos limitados, formam a motivação desta pesquisa.

## 1.1 Segurança da informação

A segurança da informação é uma coleção de práticas que protegem o sistema de informação e os dados armazenados nele. Essas práticas auxiliam na segurança das informações privadas (KIM; SOLOMON, 2014).

A *International Organization for Standardization* (ISO), a maior organização de padronização do mundo, junto com a *International Electrotechnical Commission* (IEC) deram

[1] https://owasp.org.

origem às normas relacionadas à Segurança da Informação, representada pela série ISO 27000. Como exemplo, a ISO 27001 define padrões para implementar, operar, manter, revisar e ter um processo de melhoria contínua nos sistemas de segurança, ideal para estruturar as políticas de segurança da informação (MORAES; HAYASHI, 2021).

Mesmo com as normas e boas práticas definidas pela ISO, segundo o Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil (CERT.br), a quantidade de incidentes reportados em 2020 foi de 665.079. Esse número, apesar de ser 24% menor que o total do ano de 2019, ainda é muito significativo. A Figura 1.2 mostra a evolução dos incidentes ao longo dos anos.

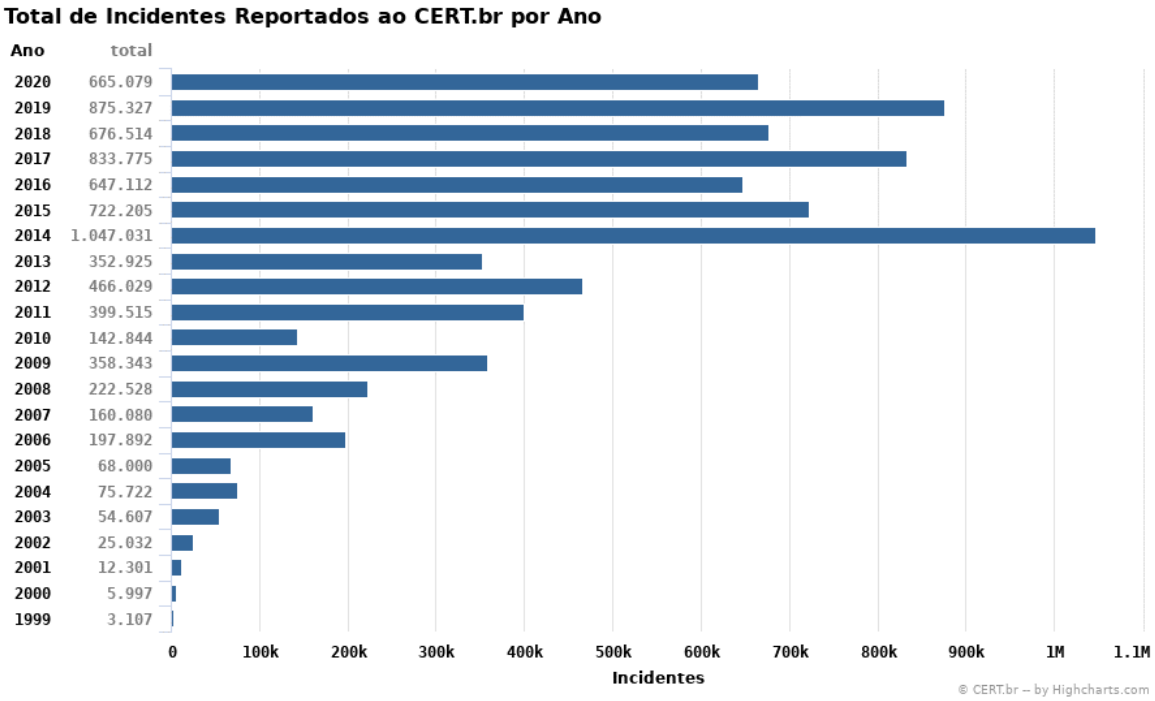



Figura 1.2: Incidentes reportados ao CERT.br. Fonte: (CERT.BR, 2022b).

Outro dado levantado pela pesquisa do CERT.br sobre a origem dos incidentes, ilustrado na Figura 1.3, demonstra que a maioria (51,93%) dos incidentes reportados em 2020 tem como origem o Brasil, seguido pelos Estados Unidos, com 22,59%, e China, com 6,5%. As estatísticas dos incidentes apresentadas pelo CERT.br salientam a necessidade da segurança da informação quando utilizados dispositivos conectados à Internet.

Os incidentes apresentados na Figura 1.3 são do tipo DoS, invasão, *web*, *scan*, fraude e outros que não se aplicam nas categorias anteriores. Neste caso, são excluídos apenas os ataques usando *worms*.

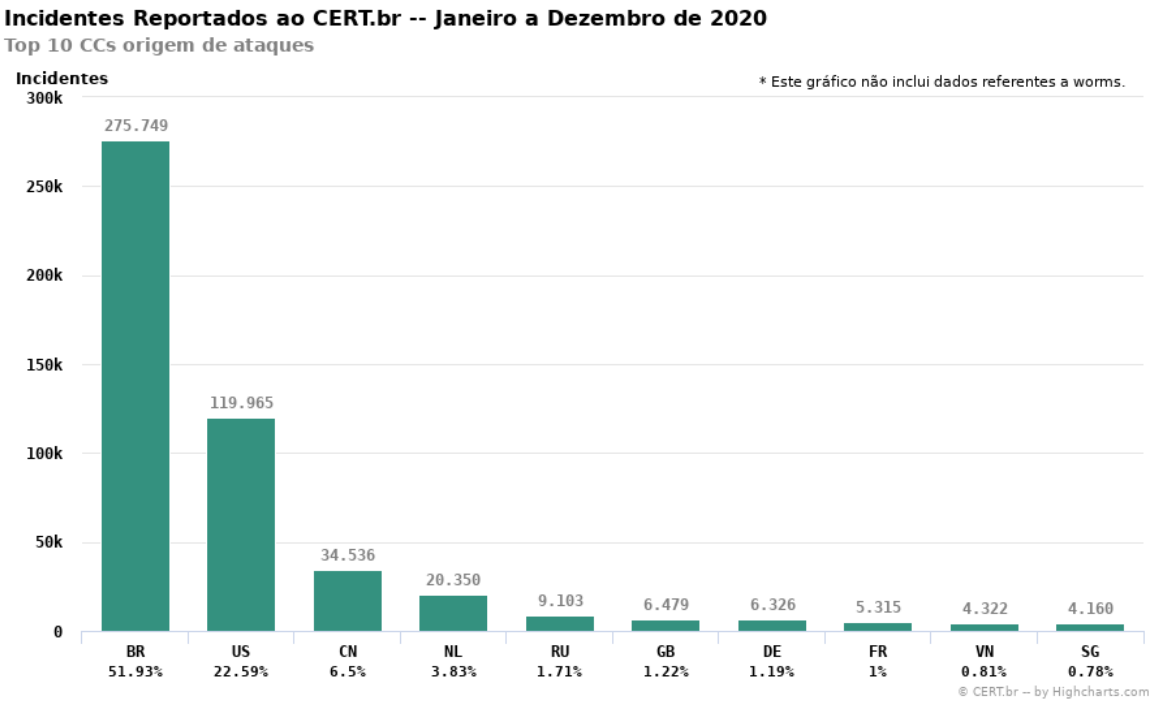



Figura 1.3: Origem dos incidentes reportados ao CERT.br. Fonte: (CERT.BR, 2022a).

Os autores Heartfield e Loukas (2015) e Kim e Solomon (2014) afirmam que, na segurança da informação, o usuário é visto como "o elo mais fraco". Assim, mesmo com sistemas de proteção fortes, as barreiras de segurança podem ser contornadas se o invasor souber manipular o usuário. Visando minimizar o risco fator humano em sistemas de segurança, Moraes e Hayashi (2021) recomendam a criação de políticas de segurança que incluam procedimentos, guias e recomendações, obrigando o usuário a assinar um documento aceitando e respeitando a política de segurança. Apenas essa obrigação não resolve o problema, mas deixa o usuário mais atento à questão.

As informações seguras devem satisfazer três princípios fundamentais de segurança. Esses princípios são a confidencialidade, a integridade e a disponibilidade, também conhecidos como a tríade *Confidentiality, Integrity, Availability* (CIA). De acordo com MACHADO (2014), a disponibilidade é a capacidade que os sistemas têm de executar e disponibilizar os dados de forma adequada ao usuário. A informação pode ser acessada por usuários autorizados sempre que solicitada. Geralmente a disponibilidade é expressa como a quantidade de tempo que um usuário pode usar um sistema (KIM; SOLOMON, 2014).

O processo de integridade garante que uma determinada ação executada pelo sistema ocorra de forma que, durante o processamento, os dados e o fluxo de informações estejam coerentes, ou seja, não houve alteração proposital por usuários não autorizados (MORAES;

HAYASHI, 2021). Nesse caso, *hardware*, *software* e comunicação devem trabalhar em conjunto para manter, processar e movimentar os dados sem qualquer alteração não autorizada ou não esperada (MACHADO, 2014).

Por sua vez, a confidencialidade é o modo de assegurar que a troca de informações entre dispositivos ocorra de modo seguro, ou seja, apenas usuários autorizados tem acesso à informação (MORAES; HAYASHI, 2021). Utilizar a criptografia de dados em dispositivos de IoT, como forma de proteger os dados e informações do usuário, é um meio para fornecer confidencialidade (WANG et al., 2019).

## 1.2 Objetivos

Este trabalho tem como principal objetivo a análise da viabilidade da implementação de algoritmos criptográficos em dispositivos IoT. Também é objetivo do trabalho discutir a importância da implementação de algoritmos criptográficos que asseguram uma maior confidencialidade dos dados dos usuários desses dispositivos.

Foram avaliados quatro algoritmos de criptografia, sendo eles, o AES, Present, Simon e Speck, em diferentes tempos de simulação.

Para atingir o objetivo principal são propostos os seguintes objetivos específicos:

- Realizar uma revisão sistemática da literatura e identificar os principais algoritmos para criptografia já utilizados.
- Selecionar os principais algoritmos utilizados em dispositivos IoT.
- Analisar o consumo de energia de cada algoritmo.
- Analisar a resposta aos diferentes tempos de execução de cada algoritmo.
- Comparar os resultados obtidos no trabalho com os apresentados na literatura.

## 1.3 Estrutura do texto

Este texto contém outros cinco capítulos, estruturados da seguinte forma. O Capítulo 2 apresenta o referencial teórico, no qual são abordados os conceitos introdutórios e a matemática essencial para a criptografia. Além disso, são mencionados alguns dados

históricos e os algoritmos mais relevantes dentro nas subdivisões (simétrico, assimétricos e leves). Esse capítulo também aborda a importância, os desafios e as aplicações da Internet das Coisas no dia-a-dia.

As pesquisas precursoras e os resultados de diferentes métodos aplicados por diferentes autores é o tema do Capítulo 3. Nesse capítulo, são discutidos os resultados e considerações relevantes para estruturação dessa pesquisa.

No Capítulo 4 são descritos os materiais e métodos escolhidos para os experimentos a serem realizados nesta pesquisa. Os resultados e discussões dos experimentos realizados são apresentados no Capítulo 5. Por fim, o Capítulo 6 apresenta as conclusões deste trabalho.

# Capítulo 2

# Referencial teórico

Este capítulo apresenta os principais conceitos utilizados para o desenvolvimento deste trabalho. A Seção 2.1 trata sobre alguns conceitos matemáticos, que são a base dos algoritmos criptográficos. A Seção 2.2 aborda conceitos introdutórios sobre criptografia, que definem e servem para comparar alguns métodos de criptografia.

A Seção 2.3, por sua vez, apresenta um breve histórico e outras informações sobre a importância da Internet das Coisas. Nessa seção é possível observar a relevância da IoT no dia-a-dia e as previsões futuras associadas a essa tecnologia. Por fim, algumas considerações finais sobre este capítulo são feitas na Seção 2.4.

## 2.1 Fundamentos matemáticos para a criptografia

Antes de descrever as técnicas para a criptografia, alguns conceitos matemáticos devem ser observados, em especial a teoria dos números. A teoria dos números é um dos ramos mais antigos da Matemática. Recentemente, essa teoria tornou-se ainda mais relevante quando o assunto é criptografia e segurança dos dispositivos computacionais (SCHEINERMAN, 2016). Os conceitos básicos da teoria dos números incluem a divisibilidade, a aritmética modular, os números primos e o máximo divisor comum (ROSEN, 2009), abordados nas seções a seguir.

### 2.1.1 A operação de divisão

Quando se divide um número inteiro por outro número inteiro diferente de zero, o quociente não é necessariamente um número inteiro. Por exemplo: $16 \div 2 = 8$ o quociente é um número

inteiro. Por outro lado, a divisão dos números inteiros 15 e 2 ($15 \div 2 = 7,5$) não resulta em um quociente inteiro.

A definição descrita em Rosen (2009) diz que, se $a$ e $b$ são números inteiros com $a \neq 0$, então, *a divide b* se houver um número inteiro $c$ de modo que $b = a \times c$. Quando *a divide b*, afirma-se que $a$ é um *fator* de $b$ e $b$ é um *múltiplo* de $a$. A notação $a \mid b$ indica que *a divide b*. A partir dessa definição, tem-se o seguinte Teorema 1.

**Teorema 1** *Sejam a, b e c números inteiros:*

***i.*** *Se $a \mid b$ e $a \mid c$, então $a \mid (b + c)$;*

***ii.*** *Se $a \mid b$, então $a \mid (b \times c)$ para todo inteiro c;*

***iii.*** *Se $a \mid b$ e $b \mid c$, então $a \mid c$;*

Como consequência do Teorema 1, se $a$, $b$ e $c$ são números inteiros, tal que $a \mid b$ e $b \mid c$, então $a \mid (m \times b + n \times c)$, desde que $m$ e $n$ sejam números inteiros.

O algoritmo de divisão euclideana mostra que quando um inteiro $a$ é dividido por um inteiro positivo $d$, há um quociente $q$ e um resto $r$, com $0 \leq r < d$, tal que, $a = d \times q + r$. Nesse caso, $d$ é chamado de *divisor*, $a$ é chamado de *dividendo*, $q$ de quociente e $r$ de resto. Utiliza-se a Equação 2.1 para expressar o quociente e Equação 2.2 para expressar resto.

$$q = a \text{ div } d \tag{2.1}$$

$$r = a \bmod d \tag{2.2}$$

### 2.1.2 Aritmética modular

Na aritmética modular, preocupa-se com o resto de um número inteiro positivo quando ele é dividido por um número inteiro positivo. O exemplo mais comum nesse tópico é o exemplo do relógio. Quando questionamos que horas serão daqui a 50 horas, preocupamo-nos apenas com o resto quando 50 mais a hora atual é dividido por 24. A aritmética modular pode ser aplicada em funções *Hash*[1], geração de números pseudo-aleatórios e em algoritmos criptográficos (ROSEN, 2009).

[1]Como a própria definição do verbo "*to Hash*" indica, *hash* significa cortar algo ou fazer uma confusão. Uma função *Hash* é qualquer função que mapeia um conjunto de dados de diferentes tamanhos em outro conjunto de dados de tamanho fixo (KNUTH, 1998).

Na definição de Burton (2016) para números congruentes, seja $n$ um número inteiro positivo, os inteiros $a$ e $b$ são *congruentes módulo n*, se $n$ divide a diferença $a - b$, ou seja, $a - b = k \times n$ para algum inteiro $k$. Utiliza-se a notação $a \equiv b \mod n$ para indicar que $a$ é congruente a $b$ módulo $n$. A partir da sua definição, Burton (2016) cita ainda os seguintes teoremas 2 e 3:

**Teorema 2** *Para inteiros arbitrários a e b, $a \equiv b \mod n$, se e somente se, $(a \mod n) = (b \mod n)$, ou seja, a e b deixam o mesmo resto não negativo quando divididos por n.*

**Teorema 3** *Seja $n > 1$ um inteiro fixo e a, b, c, d inteiros arbitrários. Então as seguintes propriedades são válidas:*

***i.*** $a \equiv a \pmod{n}$.

***ii.*** *Se* $a \equiv b \pmod{n}$, *então* $b \equiv a \pmod{n}$.

***iii.*** *Se* $a \equiv b \pmod{n}$ *e* $b \equiv c \pmod{n}$, *então* $a \equiv c \pmod{n}$.

***iv.*** *Se* $a \equiv b \pmod{n}$ *e* $c \equiv d \pmod{n}$, *então* $a+c \equiv b+d \pmod{n}$ *e* $(a \times c) \equiv (b \times d) \pmod{n}$.

***v.*** *Se* $a \equiv b \pmod{n}$, *então* $(a + c) \equiv (b + c) \pmod{n}$ *e* $a \times c \equiv b \times c \pmod{n}$.

***vi.*** *Se* $a \equiv b \pmod{n}$, *então* $(a \times k) \equiv (b \times k) \pmod{n}$ *para todo inteiro positivo k.*

Como consequência dos teoremas 2 e 3, Rosen (2009) afirma que tendo um número inteiro positivo $m$ e os inteiros $a$ e $b$, é possível estabelecer as equações 2.3 e 2.4.

$$(a + b) \mod m = ((a \mod m) + (b \mod m)) \mod m \tag{2.3}$$

$$(a \times b) \mod m = ((a \mod m) \times (b \mod m)) \mod m \tag{2.4}$$

### 2.1.3 Números primos

A definição de números primos diz que um inteiro $p > 1$ é *primo*, se e somente se, $p$ é divisível por 1 e por $p$ (TERADA, 2008). Os números 1, 2, 3, 5, 7, 11 e 13 são exemplos de números primos. Assim, denomina-se número composto, um número inteiro maior que 1 que não é primo (BURTON, 2016).

Em suma, o teorema fundamental da aritmética diz que todo número inteiro positivo maior que 1 pode ser escrito unicamente como primo ou como produto de dois ou mais primos, em que o valor dos fatores primos são escritos em ordem não decrescente de tamanho (ROSEN, 2009).

Quanto maior o número de algarismos de um número, maior a dificuldade em encontrar números primos. Estatisticamente, existe um primo em cada quatro números menores que 100. Para números com 10 algarismos, a relação é 1 primo para cada 23 números; em números com 100 algarismos, apenas um entre 230 números é primo (TERADA, 2008).

A densidade dos primos é inversamente proporcional ao seu comprimento em algarismos. A probabilidade de um inteiro $n$ ser primo é de aproximadamente $\frac{1}{(\ln n)}$, onde $(\ln n)$ denota logaritmo natural de $n$ (TERADA, 2008).

### 2.1.4 Máximo divisor comum e o Algoritmo de Euclides

O máximo divisor comum (mdc) é o maior número inteiro que divide dois números inteiros. Considerando $a$ e $b$ números inteiros diferentes de zero. O maior número inteiro $d$, tal que, $(d \mid a)$ e $(d \mid b)$, é chamado máximo divisor comum de $a$ e $b$ (ROSEN, 2009). Indica-se o mdc de $a$ e $b$ através da notação $\text{mdc}(a, b)$.

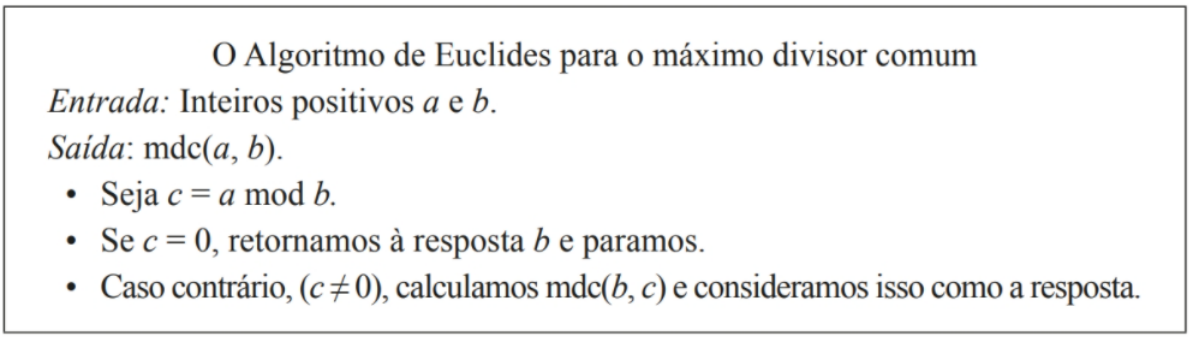
Expressando em números, para encontrar o $\text{mdc}(24, 36)$, primeiro encontram-se os divisores comuns, nesse caso são, 1, 2, 3, 4, 6 e 12. Assim, o $\text{mdc}(24, 36) = 12$.

Quando o mdc de dois números é 1, diz-se que esses números são relativamente primos. Terada (2008) traz o seguinte exemplo: o $\text{mdc}(22, 19) = 1$ e o $\text{mdc}(22, 11) = 11$, então, 22 e 19 são relativamente primos, mas 22 e 11 não.

Assim, Euclides criou um processo para calcular o mdc de dois números positivos. O algoritmo de Euclides pode ser considerado rápido e simples de implementar em um programa de computador (SCHEINERMAN, 2016). A Figura 2.1 apresenta uma descrição formal e simplificada do algoritmo de Euclides.

### 2.1.5 Complexidade dos Algoritmos

A eficiência de um algoritmo pode ser analisada utilizando duas medidas. A primeira medida é o tempo de computação necessário para resolver um problema usando o algoritmo. A segunda medida é a quantidade de memória computacional necessária para implementar o algoritmo (TOSCANI; VELOSO, 2012).


O Algoritmo de Euclides para o máximo divisor comum

*Entrada:* Inteiros positivos $a$ e $b$.

*Saída*: mdc($a$, $b$).

- Seja $c = a \bmod b$.
- Se $c = 0$, retornamos à resposta $b$ e paramos.
- Caso contrário, ($c \neq 0$), calculamos mdc($b$, $c$) e consideramos isso como a resposta.

Figura 2.1: Algoritmo de Euclides. Fonte: (SCHEINERMAN, 2016).

Quando se avalia a resistência de um algoritmo de encriptação contra criptoanálise, na verdade, examina-se a quantidade de tempo que o algoritmo é atacado (STALLINGS, 2015). Uma análise do tempo necessário para resolver um problema de determinado tamanho envolve a complexidade temporal do algoritmo (TOSCANI; VELOSO, 2012).

A complexidade temporal de um algoritmo pode ser expressa em termos da quantidade de operações usadas por esse algoritmo, quando a entrada tem um tamanho definido (ROSEN, 2009). Ao analisar o maior número de operações necessárias para resolver um problema usando um algoritmo com entrada definida, o que se observa é o comportamento do algoritmo no seu pior caso. Tal análise considera a quantidade de operações que um algoritmo necessita para produzir uma solução quando a combinação dos dados de entrada fornecida para esse algoritmo implicar na maior quantidade de operações para chegar ao resultado (ROSEN, 2009).

Existe ainda a análise de caso médio. Nesse tipo de análise encontra-se o número médio de operações usadas para resolver os problemas sobre todas as entradas de tamanhos determinados. A análise do caso médio é pouco usada, pois é considerada muito mais complicada que a análise do pior caso (MUELLER; MASSARON, 2018).

A Tabela 2.1 apresenta um resumo das três principais análises de complexidades e suas terminologias, considerando um algoritmo com uma entrada de tamanho $n$, uma constante $t$ e polinômio $h(n)$.

## 2.2 Criptografia

De acordo com o dicionário[2], a criptografia é um conjunto de regras e técnicas utilizadas para criptografar a escrita, transformando um texto em claro em um código incompreensível para

[2]CRIPTOGRAFIA. In: DICIO, Dicionário Online de Português. Porto: 7Graus, 2022. Disponível em: `https://www.dicio.com.br/criptografia`. Acesso em: 01/04/2022.

Tabela 2.1: Terminologia comum para a análise de complexidade dos algoritmos.

| **Complexidade** | **Terminologia** |
|---|---|
| $O(n)$ | Complexidade linear |
| $O(n^t)$ | Complexidade polinomial |
| $O(t^{h(n)})$ | Complexidade exponencial |

quem não está autorizado a acessá-lo. No entanto, essa definição abrange apenas a comunicação secreta e ainda se refere à criptografia como uma forma de arte, conceito adequado até o Século XX (KATZ; LINDELL, 2020).

Historicamente, a criptografia – muito utilizada por organizações militares e de inteligência – conta com o primeiro uso documentado datado de 1900 a.C., no Egito, quando um escriba usou hieróglifos fora do padrão (KHALIFA et al., 2004). Entre 600 a.C. e 500 a.C., os hebreus utilizaram a cifra ATBASH de substituição simples, monoalfabética e monogrâmica, ou seja, os caracteres são trocados um a um por outros, para escrever o Livro de Jeremias (MORKEL; ELOFF, 2004).

O mais simples e mais conhecido método de uso de uma cifra de substituição é a cifra de César, criada pelo imperador Júlio César, do Império Romano (MORAES; HAYASHI, 2021). A cifra de César substitui cada letra do alfabeto pela letra três posições adiante, a Figura 2.2 mostra como ficaria o alfabeto se aplicado a cifra de César. O número "três"representa a chave secreta utilizada. Sendo assim, e longe de ser segura, a cifra de César possui apenas 25 chaves possíveis (STALLINGS, 2015; MORAES, 2010).

```
claro: a b c d e f g h i j k l m n o p q r s t u v w x y z
cifra: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Figura 2.2: Cifra de César. Fonte: (STALLINGS, 2015).

Uma forma de aprimorar a técnica monoalfabética simples é utilizar a cifra por substituição polialfabética. Tal técnica aplica diferentes substituições monoalfabéticas enquanto se prossegue pela mensagem de texto claro, a mais simples e conhecida é a cifra de Vigenère (STALLINGS, 2015). A cifra de Vigenère pode ser considerada um marco na criptografia, pois é a partir dela que começam a surgir estudos com maior frequência sobre o assunto (KATZ; LINDELL, 2020).

A cifra de Vigenère substituiu gradualmente antigos códigos monoalfabéticos por polialfabéticos. Também começam a surgir estudos sobre a criptoanálise, técnica que explora as características do algoritmo em busca do texto em claro ou da chave utilizada com base apenas no texto criptografado. Ainda assim, criar bons códigos ou quebrá-los, dependia da criatividade e habilidade pessoal (BURTON, 2016; KATZ; LINDELL, 2020).

É a partir do Século XX que a criptografia começou a ser estudada como Ciência, passando a incluir autenticação de mensagens, assinaturas digitais, protocolos para troca de chaves secretas, protocolos de autenticação, eleições eletrônicas e dinheiro digitais (KATZ; LINDELL, 2020).

Em 1918, Arthur Scherbius explorou a tecnologia disponível na época e desenvolveu uma máquina criptográfica chamada Enigma, ilustrada na Figura 2.3, a mais temível da história. Amplamente utilizada pelo governo e organizações estatais, como as ferrovias. No início da Segunda Guerra Mundial as comunicações dos militares alemães foram protegidas pela máquina de Scherbius (SINGH, S., 1999).

Figura 2.3: Enigma. Fonte: Máquina Enigma.

Os decifradores de códigos contribuíram para a criação dos computadores. Mesmo após a guerra, os estudos para desenvolver e empregar tecnologia de computador para quebrar todos os tipos de criptografias não pararam. A velocidade e a flexibilidade dos computadores programáveis em pesquisar todas as chaves possíveis até encontrar a correta, não impediu os

cifradores de revidar, através dos computadores para criar criptografias cada vez mais complexas (KIM; SOLOMON, 2014; SINGH, S., 1999).

Com o avanço da tecnologia e da engenharia, os computadores foram ficando cada vez mais acessíveis, tornando-se uma realidade comercial em 1951 (SINGH, S., 1999). A ascensão aconteceu ainda mais em 1960, com computadores mais poderosos e mais baratos. As empresas passaram então a usá-los e, para fazer a transmissão de dados, surgiu a necessidade de segurança. Com isso, houve a demanda para criptografar transferências de dinheiro e negociações comerciais delicadas.

No entanto, esse crescimento deu espaço para um grande problema, a distribuição de chaves. Assim, os pesquisadores Whitfield Diffie e Martin Hellman, apresentaram em 1976 uma solução para o problema. A solução ficou conhecida como a troca de chaves de Diffie e Hellman (BURTON, 2016; TERADA, 2008).

Atualmente, a criptografia está em todo lugar. Mecanismos de segurança que dependem de criptografia são parte integrante dos sistemas de computador. Toda vez que um site seguro é acessado, métodos criptográficos são empregados. A criptografia atual dificulta que ataques a segredos em sistemas operacionais multiusuários ocorram.

Tem-se ainda métodos de proteção de software e autenticação que utilizam criptografia. Katz e Lindell (2020) definem a criptografia moderna como o estudo científico de técnicas para proteger informações digitais, transações e cálculos distribuídos.

Stallings (2015) afirma que os sistemas criptográficos são caracterizados ao longo de três dimensões independentes. Na primeira dimensão, nas operações usadas para transformar texto claro em texto criptografado são feitas substituições e transposições sem que haja a perda de informações.

Na segunda dimensão, observa-se o número de chaves usadas. Se o emissor e o receptor operam com a mesma chave, trata-se da criptografia simétrica ou também conhecida como criptografia de chave única, abordada na Seção 2.2.1. Se o emissor e o receptor utilizam chaves diferentes, trata-se da criptografia assimétrica ou de chave pública, detalhado na Seção 2.2.2.

Por fim, a terceira dimensão classifica o modo em que o texto é processado, em bloco ou em fluxo. Quando os elementos de entrada são processados em bloco e produzem um bloco de saída para cada entrada tem-se uma criptografia de bloco. Uma criptografia de fluxo processa elementos de entrada continuamente apresentando um elemento por vez na saída (MORAES, 2010).

Vale ressaltar que a segurança de qualquer esquema de criptografia está diretamente relacionada com o tamanho da chave utilizada e do trabalho computacional necessário para quebrar uma criptografia (BATTEN, 2013). Uma característica importante dos algoritmos criptográficos é o efeito avalanche. Esse efeito ocorre, quando a alteração de um bit de entrada implica na alteração de mais de 50% dos bits da saída. É muito comum intrusos atacarem algoritmos criptográficos que possuem um efeito avalanche fraco (MUTHAVHINE; SUMBWANYAMBE, 2018).

### 2.2.1 Criptografia Simétrica

Na criptografia Simétrica ou criptografia de chave única, emissor e destinatário compartilham a mesma chave. De acordo com Beg, Al-Kharobi e Al-Nasser (2019), os algoritmos simétricos exigem que os envolvidos na comunicação compartilhem um segredo, usado para criptografar e decriptar os dados. Esse segredo não é necessariamente a chave, mas um meio de obtê-la.

Em uma definição sistêmica apresentada por Moraes e Hayashi (2021), um esquema de encriptação simétrica possui os seguintes cinco itens, ilustrados também na Figura 2.4:

- Texto claro: mensagem ou dados originais, que são a entrada do algoritmo de encriptação.
- Algoritmo de encriptação: aquele que faz diversas transformações no texto claro.
- Chave secreta: é também uma entrada para o algoritmo de encriptação e deve ser um valor independente do texto claro. A saída do algoritmo será diferente e depende da chave utilizada.
- Texto criptografado: é a mensagem embaralhada, a saída do algoritmo de encriptação. É um conjunto de dados aparentemente aleatório e ininteligível.
- Algoritmo de decriptação: aquele que faz o processo reverso da criptografia, transforma os dados encriptados em texto claro.

Stallings (2015) também descreve dois requisitos para o uso seguro da criptografia simétrica. O primeiro requisito diz que o algoritmo escolhido deve ser seguro, de modo que, mesmo que um atacante conheça o algoritmo e tenha acesso a um ou mais de um texto criptografado, seja incapaz de descriptografá-lo ou descobrir a chave secreta utilizada. Esse é

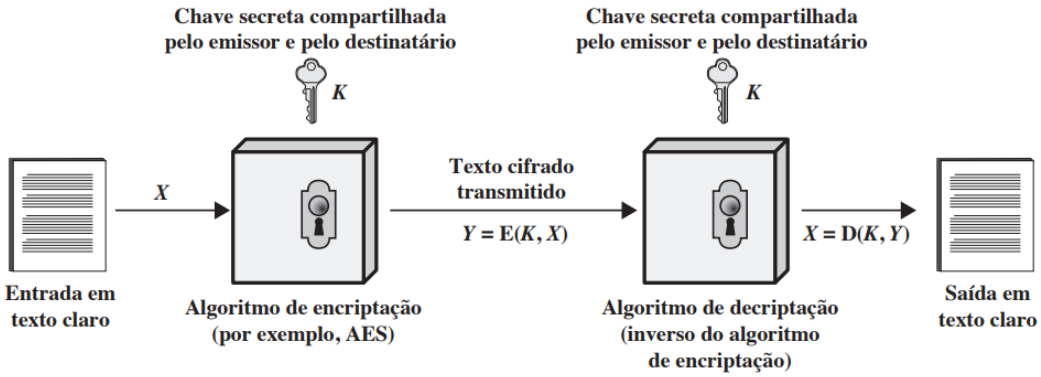



Figura 2.4: Modelo genérico de encriptação simétrica. Fonte: (STALLINGS, 2015).

o chamado princípio de Kerckhoffs (1883) que afirma que um criptossistema deve ser seguro, mesmo que tudo sobre o sistema, exceto a chave, seja do conhecimento público.

O segundo requisito envolve a troca de chaves entre emissor e receptor. Essa troca das chaves deve ser feita através de um canal seguro pois, caso contrário, se o atacante descobre a chave e o algoritmo, toda a comunicação que faz uso dessa chave e do algoritmo poderá ser descoberta.

Segundo Stallings (2015) e Moraes e Hayashi (2021), os dois principais algoritmos criptográficos simétricos são o *Data Encryption Standard* (DES) e o *Advanced Encryption Standard* (AES). Em 1999, o Instituto Nacional de Padrões e Tecnologia (*National Institute of Standards and Technology* – NIST) dos Estados Unidos emitiu uma nova versão indicando que o DES deveria ser substituído pelo Triple DES (3DES) que, em resumo, seria repetir o DES três vezes sobre o texto claro com duas ou três chaves diferentes para produzir o texto criptografado (STALLINGS, 2015). Posteriormente, o NIST recomendou ainda que o algoritmo 3DES fosse substituído pelo AES.

A principal limitação que envolve o uso da criptografia simétrica é o gerenciamento das chaves secretas. Para começar, é necessário um canal seguro para que emissor e receptor troquem o segredo. No entanto, tratando-se de criptografia simétrica, toda vez que uma nova entidade desejar se comunicar com as partes do sistema, uma nova chave deverá ser gerada, implicando na demanda do gerenciamento de muitas chaves. Além disso, algoritmos de chave secreta não podem ser aplicados para fins de autenticação (KATZ; LINDELL, 2020).

### *Data Encryption Standard*

O *Data Encryption Standard*, publicado em janeiro de 1977 pelo *United States' Bureau of Standard* para ser usado em dados não relacionados à segurança nacional, foi amplamente utilizado no mundo e aplicado dominantemente por bancos para garantir segurança na transferência de fundos. Aceito por um período de cinco anos originalmente, o DES resistiu aos testes do tempo sendo aprovados por mais três períodos de cinco anos (MAO, 2003).

O DES foi projetado para ser implementado em *hardware* (KIM; SOLOMON, 2014). Segundo Terada (2008), o DES é considerado inadequado para implementações em *software* devido à ocorrência de diversas permutações de sequências longas de bits.

O processo de encriptação consiste em múltiplas operações binárias de permutações em blocos de 64 bits. Os blocos de 64 bits passam pela permutação utilizando a chave de 56 bits, a saída é então um bloco de 64 bits já encriptado (MORAES; HAYASHI, 2021). No entanto, com os avanços em capacidade de *hardware* e métodos de ataque, é possível ter resultados de um ataque ao DES em alguns dias (KIM; SOLOMON, 2014).

Como alternativa, indica-se o uso do *Triple DES* (3DES), criado em 1977, com a intenção de aumentar a segurança do DES. Aplica-se o DES três vezes, porém com diferentes chaves de 56 bits, isso possibilita que as chaves criptográficas tenham 168 bits ou 112 bits (MORAES; HAYASHI, 2021). Segundo Kim e Solomon (2014), o 3DES tornou-se computacionalmente seguro, devido à segurança subjacente do DES e ao aumento dos bits das chaves.

### *Advanced Encryption Standard*

O *Advanced Encryption Standard* (AES) é considerado o mais seguro e usado na criptografia simétrica. Projetado por dois criptógrafos belgas, Vincent Rijmen e Joan Daemen, o algoritmo venceu a competição do NIST em 2001 (WU; DWIVEDI; SRIVASTAVA, 2021).

O AES é uma criptografia de bloco que surgiu da intenção de substituir o DES como o padrão para variadas aplicações (STALLINGS, 2015). A razão para o uso do AES é devido ao nível de segurança elevado com o menor tamanho de chave quando comparado com outros algoritmos como o de curva elíptica, o El-Gamal, o RSA e outros (WU; DWIVEDI; SRIVASTAVA, 2021).

O AES pode trabalhar com três cifras de blocos, o AES-128, o AES-192 e o AES-256. Cada uma das cifras criptografa e decriptografa blocos de 128, 192 ou 256 bits respectivamente. O processo criptográfico possui 10 etapas quando utilizadas chaves de 128 bits, 12 etapas com

chaves 192 bits e 14 com chaves de 256 bits. Nas etapas realiza-se o procedimento de substituição, transposição do texto claro para torná-lo texto criptografado, utilizando sempre as funções matemáticas do algoritmo (MORAES; HAYASHI, 2021). As etapas repetem quatro transformações básicas:

1. *SubBytes*: etapa de substituição não linear, onde cada byte é substituído por outro de acordo com a caixa de substituição (S-box).

2. *ShiftRows*: permutação simples.

3. *MixColumns*: operação de mistura linear que opera nas colunas do estado, combinando os quatro bytes de cada coluna

4. *Add Round Key*: a chave da rodada, derivada da chave principal, é adicionada.

Moraes e Hayashi (2021) descrevem, de forma resumida, a primeira e a última transformação feita pelo algoritmo AES. Na primeira transformação acontece a substituição dos dados em uma tabela, logo em seguida, os dados são deslocados nas linhas e depois nas colunas.

A última transformação realiza uma operação *ou-exclusivo* (XOR) em cada coluna, usando uma parte diferente da chave de encriptação. É devido a essa operação que chaves maiores exigem mais etapas de processamento. A Figura 2.5 apresenta a estrutura do processo de encriptação do AES.

### 2.2.2 Criptografia Assimétrica

De acordo com Mao (2003), em 1976, Diffie e Hellman propuseram uma compreensão totalmente nova para a criptografia, isto é, a criptografia assimétrica ou também conhecida como criptografia de chave pública.

Na criptografia assimétrica, emissor e destinatário possuem diferentes chaves. As chaves são conhecidas como chave pública e chave privada. A chave pública é de conhecimento geral e todos podem ter acesso. Em contrapartida, a chave privada é secreta e deve ser mantida em sigilo pela entidade portadora (BEG; AL-KHAROBI; AL-NASSER, 2019).

Algoritmos criptográficos assimétricos fornecem autenticação e confidencialidade, sendo então usadas por provedores de segurança. O principal motivo para a não escolha de tal

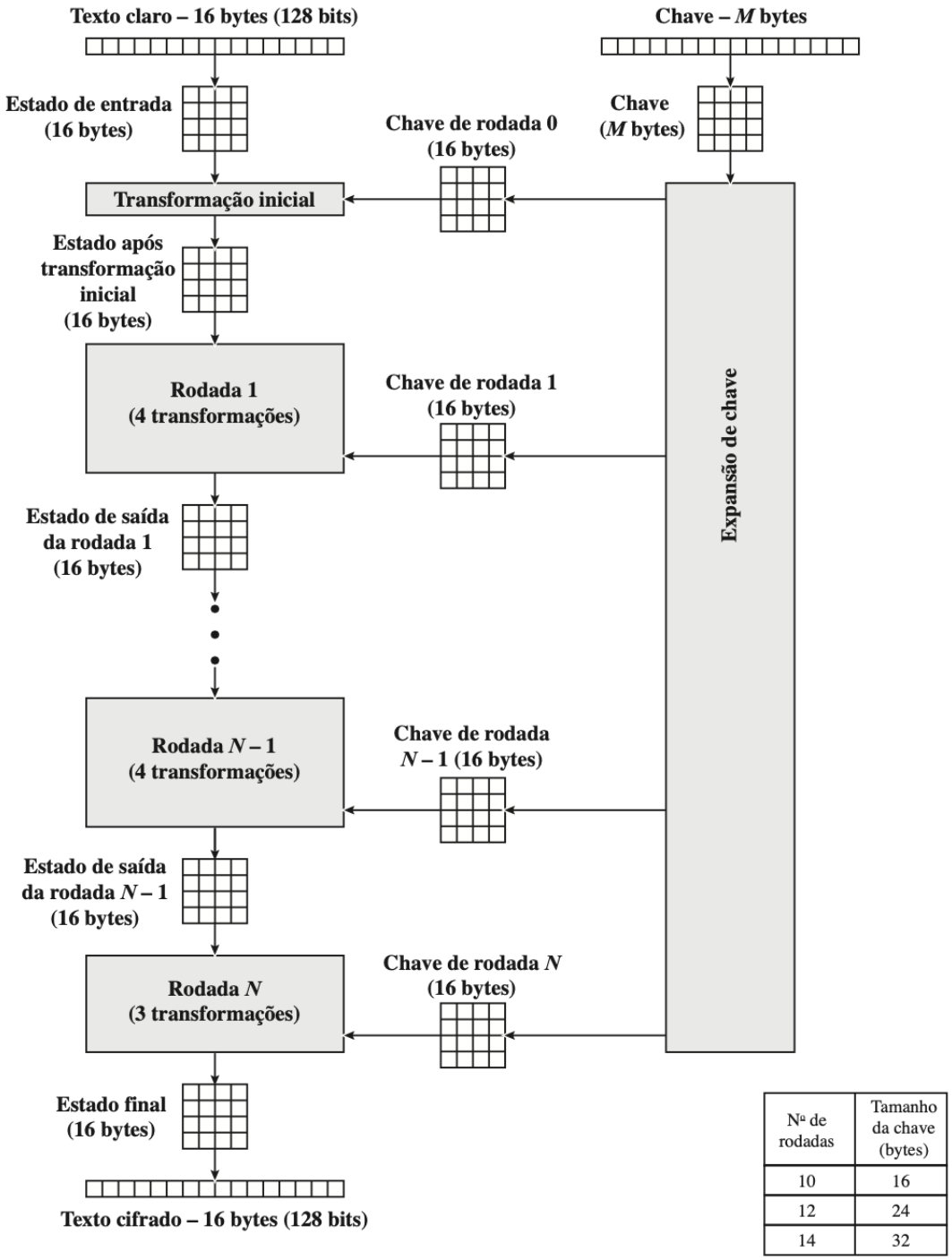



Figura 2.5: Processo de encriptação do AES. Fonte: (STALLINGS, 2015).

esquema é o maior tamanho de chave, maior consumo de memória e alto custo de processamento (DHANDA; SINGH, B.; JINDAL, 2020).

De acordo com Stallings (2015), a característica mais importante de algoritmos assimétricos é a inviabilidade computacional de determinar a chave privada dado o conhecimento da chave pública. Ainda, alguns algoritmos como Rivest-Shamir-Adleman

(RSA), demonstram que qualquer uma das chaves (pública ou privada) pode ser usada para a encriptação, com a outra então sendo usada para a decriptação. Um esquema de encriptação de chave pública, conforme ilustrado na Figura 2.6, possui os seguintes cinco elementos:

- Texto claro: mensagem ou dados originais, que são a entrada do algoritmo de encriptação.
- Algoritmo de encriptação: aquele que faz diversas transformações no texto claro.
- Chave pública e privada: é um par de chaves selecionadas de modo que, quando uma é empregada na encriptação, a outra servirá para decriptação. Aqui, é importante destacar que uma das chaves não pode ser obtida matematicamente a partir da outra.
- Texto criptografado: é a mensagem embaralhada, a saída do algoritmo de encriptação. É um conjunto de dados aparentemente aleatório e ininteligível.
- Algoritmo de decriptação: aquele que faz o processo reverso da criptografia, transforma os dados encriptados em texto claro.

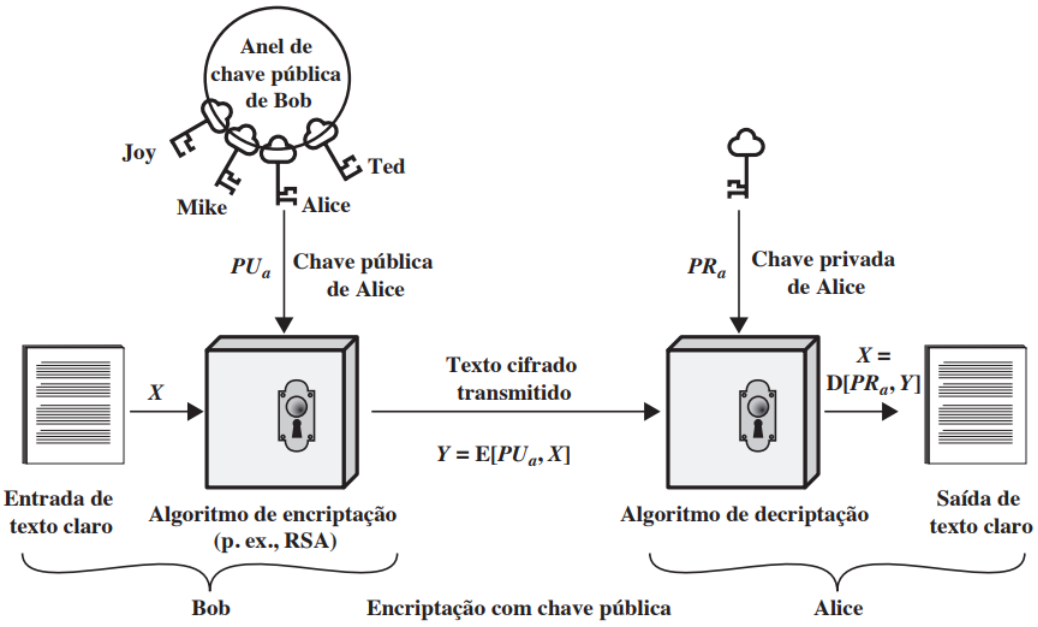



Figura 2.6: Modelo de uso de encriptação com criptografia assimétrica utilizado para dar sigilo à informação. Fonte: (STALLINGS, 2015).

A Figura 2.6 mostra como um esquema de criptografia assimétrica garante a confidencialidade. Nessa figura, Bob possui um aglomerado de chaves, com as chaves públicas de Joy, Mike, Alice e Ted, e deseja enviar uma mensagem secreta para Alice. Sendo

assim, basta Bob utilizar algum algoritmo de encriptação e a chave pública de Alice para encriptar a mensagem. Alice receberá a mensagem encriptada, em posse da chave privada e do algoritmo de decriptação Alice consegue ler a mensagem em claro que Bob mandou. Isso vale para qualquer entidade que deseje se comunicar com Alice de forma secreta. Basta ter a chave pública e o algoritmo de encriptação. Por isso, não há o compartilhamento de chaves secretas como na criptografia assimétrica.

A criptografia assimétrica também pode ser utilizada para garantir autenticação, conforme descrito na Figura 2.7. Nesse caso, usa-se a chave privada e o algoritmo de encriptação para assinar o texto. Assim, pode-se então verificar a assinatura, com a chave pública e o algoritmo de decriptação.

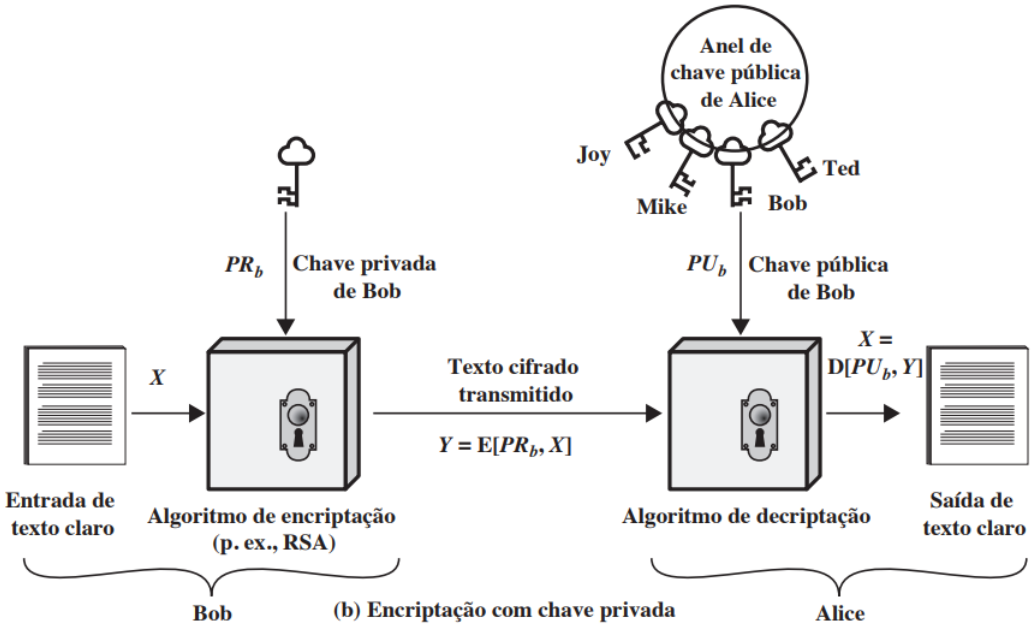



Figura 2.7: Modelo de uso de encriptação com criptografia assimétrica utilizada para a autenticação da informação. Fonte: (STALLINGS, 2015).

Diffie e Hellman (1976) estabeleceram as condições que os algoritmos assimétricos precisam atender. A primeira condição é que deve ser viável computacionalmente que uma parte $B$ gere um par de chaves: chave pública ($\mathrm{PU}_B$), chave privada ($\mathrm{PR}_B$). A segunda condição é que deve ser viável um emissor $A$, que conhece a chave pública e a mensagem a ser encriptada ($M$), gere o texto criptografado ($C$) correspondente, conforme a Equação 2.5.

$$C = E(\mathrm{PU}_B, M) \tag{2.5}$$

A terceira condição é que também deve ser viável que o receptor $B$ descriptografe o texto criptografado resultante usando a chave privada, recuperando assim a mensagem original, conforme a Equação 2.6.

$$M = D(\mathrm{PR}_B, C) \tag{2.6}$$

Diffie e Hellman (1976) ainda estabeleceram que deve ser inviável computacionalmente que um invasor, conhecendo a chave pública, consiga determinar a chave privada. Inclusive, estabeleceram que deve ser inviável que um invasor, que conheça a chave pública e um texto criptografado, consiga descobrir a mensagem original.

Desde que o conceito de chave pública foi proposto, poucos algoritmos receberam ampla aceitação. Os mais conhecidos são o RSA (desenvolvido em 1977 e publicado em 1978 por Ron Rivest, Adi Shamirm e Len Adleman, no MIT) e a criptografia de curva elíptica *Elliptic Curve Cryptography* (ECC), sugerida por Neal Koblitz e Victor S. Miller em 1985, baseada na estrutura algébrica de curvas elípticas sobre corpos finitos (DHANDA; SINGH, B.; JINDAL, 2020). Também são conhecidos o método de Diffie-Hellman (publicado em 1976 por Whitfield Diffie e Martin Hellman, método seguro para troca de chaves em canal público) e o padrão de assinatura digital, *Digital Signature Standard* (DSS), que utiliza o algoritmo de assinatura digital para seu algoritmo de assinatura e SHA-1 como algoritmo de *hash* de mensagens (DIFFIE; HELLMAN, 1976; STALLINGS, 2015).

O uso dos criptossistemas de chave pública pode ser classificado em três aplicações: (1) quando o emissor criptografa uma mensagem com a chave pública do destinatário (Encriptação); (2) quando o emissor usa sua chave privada para assinar uma mensagem (Assinatura Digital); e (3) quando o emissor e o receptor cooperam para a troca de uma chave de sessão (Troca de Chave). A Tabela 2.2 mostra as aplicações com os algoritmos citados no parágrafo anterior (STALLINGS, 2015).

### Algoritmo RSA

O algoritmo mais popular dentre os que usam criptografia assimétrica é o Rivest-Shamir-Adleman (RSA). O RSA envolve a multiplicação de dois números primos grandes, de preferência aleatórios, que devem ser mantidos em segredo. O produto dos números primos é compartilhado junto como a chave pública e um valor auxiliar.

Tabela 2.2: Aplicações de criptossistemas de chave pública. Fonte: (STALLINGS, 2015).

| **Algoritmo** | **Encriptação/Decriptação** | **Assinatura Digital** | **Troca de Chave** |
|---|---|---|---|
| RSA | Sim | Sim | Sim |
| Curva elíptica | Sim | Sim | Sim |
| Diffie-Hellman | Não | Não | Sim |
| DSS | Não | Sim | Não |

No RSA, usam-se expressões matemáticas com exponenciais. Trata-se de um algoritmo criptográfico de bloco, onde cada bloco possui um valor binário menor que algum número $n$. A Equação 2.7 expressa o processo de criptografia no RSA, e a Equação 2.8 descreve a decriptografia para blocos de texto claro $M$ e texto criptografado $C$.

$$C = M^e \mod n \tag{2.7}$$

$$M = C^d \mod n \tag{2.8}$$

O valor de $n$ é o produto de dois números primos grandes, dispondo $n$, geralmente, 309 dígitos decimais ou 1024 bits. Os primos são denominados $p$ e $q$. Para determinar a chave de encriptação $e$, deve se respeitar a condição que $1 < e < \phi(n)$. Determina-se $\phi(n)$ utilizando os primos $p$ e $q$. Então, $\phi(n) = (p-1) \times (q-1)$. Através da Equação 2.9 determina-se a chave de decriptção $d$, que deve ser maior que 0 e menor que $n$.

$$e \cdot d \equiv 1 \mod \phi(n) \tag{2.9}$$

O valor de $n$ é de conhecimento tanto do portador da chave privada, quanto do portador da chave pública. O valor de $e$ é conhecido pelo emissor e o receptor sabe o valor de $d$. Assim, a chave pública é PU $= \{e, n\}$ e a chave privada é PR $= \{d, n\}$.

As chaves mais trabalhadas no RSA variam entre 1024 e 2048 bits. Alguns autores, como Moraes e Hayashi (2021), recomendam trabalhar com chaves de 4096 bits, pois há relatos que chaves de 300 bits podem ser quebradas em algumas horas (MILANOV, 2009). O RSA não pode ser categorizado como um algoritmo leve, como os algoritmos apresentados na Seção 2.2.3, limitando o seu uso em dispositivos para IoT. Por outro lado, esse algoritmo pode ser usado

para comunicar a chave secreta compartilhada por algoritmos simétricos na fase de iniciação, aplicando em seguida um algoritmo simétrico mais rápido que o RSA (BEG; AL-KHAROBI; AL-NASSER, 2019).

#### Criptografia de Curva Elíptica

A criptografia por curva elíptica, do inglês, *Elliptic Curve Cryptography* (ECC), pode ser tão segura quanto o RSA com a vantagem de o ECC gerar uma chave muito menor exigindo menos memória e apresentando uma maior velocidade de processamento (BEG; AL-KHAROBI; AL-NASSER, 2019). Entretanto, o ECC pode pode apresentar um desempenho 100 a 1000 vezes mais lento que o algoritmo AES (EISENBARTH et al., 2007).

A técnica ECC, com o algoritmo de troca de chaves Diffie-Hellman, Curva Elíptica de Diffie-Hellman (ECDH), e El-Gamal são implementados por projetistas para fornecer segurança no cenário da IoT (DHANDA; SINGH, B.; JINDAL, 2020).

A segurança da ECC contra criptoanálise é baseada na dificuldade computacional de resolver o Problema do Logaritmo Discreto[3] sobre curvas elípticas (TERADA, 2008). A técnica ECC pode realizar as mesmas funções de um sistema criptográfico tradicional, como criptografia de grandes volumes de dados, verificação de assinaturas digitais e prover o não-repúdio (MORAES; HAYASHI, 2021).

### 2.2.3 Criptografia Leve

Os algoritmos criptográficos tradicionais são aplicáveis à dispositivos computacionais com amplos recursos de memória e processamento. Portanto, quando se fala de dispositivos leves esses algoritmos não são práticos em razão das limitações de hardware e software dos dispositivos.

Surgem então os estudos sobre algoritmos criptográficos leves, *Lightweigtht Cryptography* – LWC, também conhecidos como primitivas criptográficas leves. Esses algoritmos priorizam a resolução do problema de desempenho em dispositivos com recursos limitados, como no caso de redes de sensores sem fio e dispositivos embarcados (BEG; AL-KHAROBI; AL-NASSER, 2019).

[3] O Problema do Logaritmo Discreto pode ser definido da seguinte forma. Dados os elementos – $r$ e $q$ – e um primo $p$, deseja-se achar um número $k$, tal que, $r = q^k \mod p$. O número $k$ é denominado logaritmo discreto de $R$ na base $Q$. A segurança da ECC está ligada à dificuldade de determinar $k$, dados $r$, $q$ e $p$.

A LWC enfatiza implementações eficientes de algoritmos criptográficos e é um subcampo científico que se posiciona na interseção entre a Ciência da Computação, a Engenharia Elétrica e a criptografia. A LWC se ocupa do gerenciamento do desempenho, da segurança e do custo dos algoritmos leves (TAUSIF et al., 2017).

No entanto, o gerenciamento desses três objetivos costuma ser um desafio. Dois deles podem ser melhorados. Porém, é muito difícil alcançar os três objetivos simultaneamente. A Figura 2.8 ilustra esse equilíbrio entre desempenho, segurança e custo (TAUSIF et al., 2017).

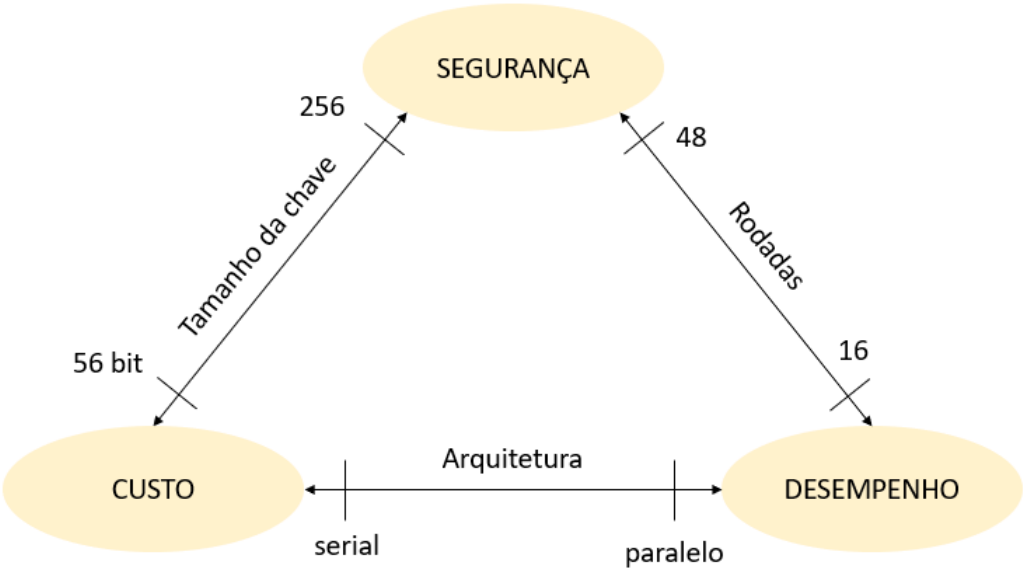



Figura 2.8: Relação entre segurança, custo e desempenho na criptografia leve. Adaptado de Tausif et al. (2017).

Dhanda, Brahmjit Singh e Jindal (2020) apresentam as quatro principais primitivas da criptografia leve disponíveis para uso, demonstradas e classificadas na Figura 2.9. São elas, a Cifra de bloco leve (*Lightweight Block Cipher* – LWBC), as Cifras de fluxo leves (*Lightweight Stream Ciphers* – LWSC), as Funções *Hash* leves (*Lightweight Hash Functions* – LWHF) e a ECC. Os fatores nos quais as primitivas criptográficas leves podem ser analisadas são: (1) o tamanho da chave; (2) o tamanho do bloco; (3) o número de rodadas; e (4) as estruturas.

As cifras de bloco leves diferem das cifras de bloco convencionais em quatro aspectos, descritos a seguir:

- Com base no tamanho dos blocos, recortando o conjunto de informações em blocos menores para diminuir o requisito de memória para cada avaliação de bloco.
- Com base no tamanho da chave, reduzindo o tamanho da chave para aumentar a velocidade do algoritmo.

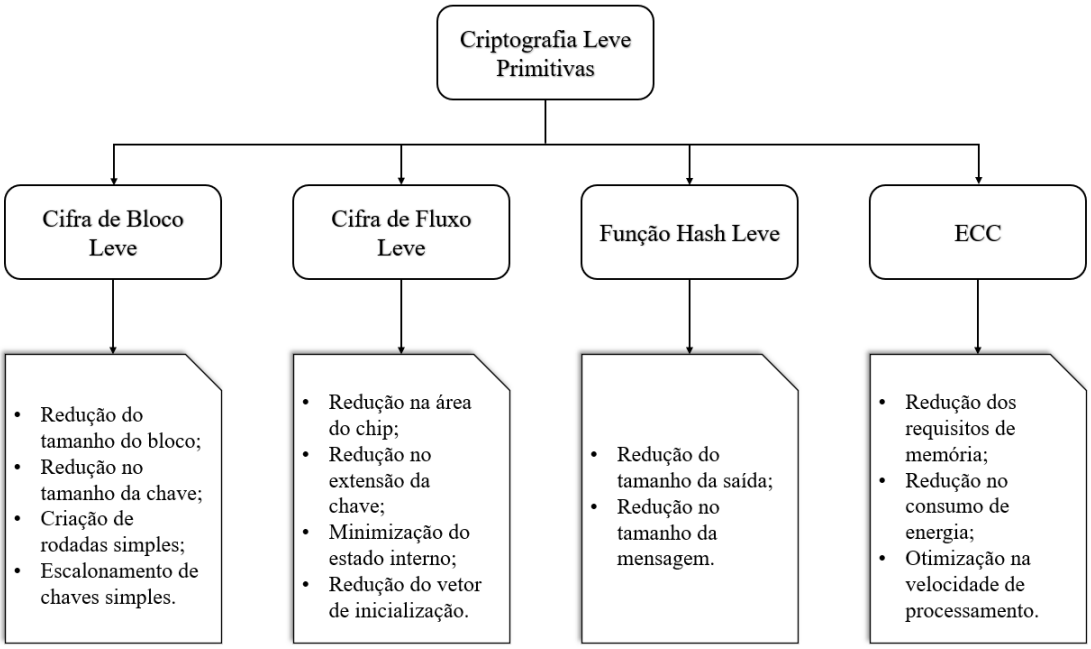



Figura 2.9: Primitivas criptográficas leves para IoT. Adaptado de Tausif et al. (2017).

- Com base na complexidade das rodadas, onde as rodadas das cifras de bloco leves têm fases mais simples de operações e componentes.

- Com base na complexidade dos escalonamentos de chaves, onde a geração das subchaves para cada rodada ou operação em cada rodada das cifras de bloco leves são mais simples economizando memória, latência e consumo de energia. Cifras de bloco leves podem cortar operações inteiras, reduzindo o número de operações (BEG; AL-KHAROBI; AL-NASSER, 2019).

As cifras de fluxo leve possuem uma maior taxa de transferência, menor requisito de tamanho de *buffer* e menor complexidade. Elas são mais rápidas e apropriadas para dados em tempo real, pois o processo de criptografar um byte de cada vez é mais rápido do que criptografar um bloco inteiro em uma cifra de bloco (BEG; AL-KHAROBI; AL-NASSER, 2019). Os principais desafios nas cifras de fluxo são minimizar o comprimento da chave, o estado interno e o vetor de inicialização mantendo a segurança proposta pela criptografia (DHANDA; SINGH, B.; JINDAL, 2020).

As funções *hash* convertem um texto claro em um texto criptografado aplicando uma função determinística. Assim, qualquer alteração no texto claro produz, na saída, um texto criptografado completamente diferente.

As funções *hash* leves foram introduzidas para minimizar o problema do tamanho de estado interno e do consumo de energia requeridos por funções *hash* típicas, viabilizando o uso em dispositivos altamente limitados. É o caso das funções PHOTON, Spongent e Quark que possuem um menor espaço interno, tamanho de saída e mensagem menores (BEG; AL-KHAROBI; AL-NASSER, 2019).

No caso da ECC, comparando-a aos demais algoritmos de criptografia assimétrica, tem-se que a ECC requer um tamanho de chave menor, tem uma velocidade de processamento maior e requer menos memória. Por isso é uma boa candidata para ser aplicada em IoT, levando a uma computação rápida em tempo real (SINGH, S. et al., 2017).

Os algoritmos de criptografia leve proporcionam um nível bem semelhante de segurança em relação aos algoritmos tradicionais. Ainda mostram um alto desempenho e baixo consumo de memória. A criptografia leve apresenta a solução para dispositivos IoT que possuem menos memória e tendem a funcionar com menos energia (PALLAVI; KUMAR; SRIKRISHNA, 2020).

### 2.2.4 Comparação entre algoritmos simétricos e assimétricos

Algoritmos simétricos apresentam vantagens no seu desempenho quando comparados com algoritmos assimétricos por ser necessário aplicar apenas uma chave tanto para o processo de criptografia quanto para a descriptografia (BEG; AL-KHAROBI; AL-NASSER, 2019).

A desvantagem apresentada pelos algoritmos simétricos é a exigência de que as partes envolvidas conheçam a chave compartilhada. Sendo assim, um acordo prévio e seguro entre as partes é indispensável (BEG; AL-KHAROBI; AL-NASSER, 2019). A Tabela 2.3 resume e compara algumas características importantes da encriptação simétrica e assimétrica.

A criptografia leve surge do aumento do uso de dispositivos com restrição de recursos que visam menor consumo de energia e espaço de armazenamento reduzido.

As vantagens das cifras leves é que elas não demandam a criptografia de uma grande quantidade de dados para serem processados. Como a quantidade de dados é pequena, a probabilidade de ocorrer um ataque também é menor devido à falta de informação. Cifras leves podem ser implementadas com projeto compacto em hardware e com programa eficiente em software (SINGH, P.; ACHARYA; CHAURASIYA, 2019).

Por fim, a Tabela 2.4 compara os algoritmos DES, 3DES, AES, RSA, ECC, e a LWC, já citados no texto. Apresentando uma análise de quais seriam ou não indicados para IoT e a razão para isso.

Tabela 2.3: Características dos métodos para encriptação simétrica e assimétrica. Adaptado de Stallings (2015).

| **Encriptação Simétrica** | **Encriptação Assimétrica** |
|---|---|
| *Necessário para funcionar:* | |
| O mesmo algoritmo com a mesma chave é usado para encriptação e decriptação. | Um algoritmo é usado para encriptação, e um relacionado, para decriptação com um par de chaves, uma para encriptação e outra para decriptação. |
| O emissor e o receptor precisam compartilhar o algoritmo e a chave. | O emissor e o receptor precisam ter, cada um, uma chave do par (não a mesma). |
| *Necessário para a segurança:* | |
| A chave precisa permanecer secreta. | Uma das duas chaves precisa ser secreta. |
| Deverá ser impossível, ou pelo menos impraticável, descriptografar uma mensagem se a chave for mantida secreta. | Deverá ser impossível, ou pelo menos impraticável, descriptografar uma mensagem se a chave privrada for mantida em segredo. |
| O conhecimento do algoritmo mais amostras do texto criptografado precisam ser insuficientes para determinar a chave. | O conhecimento do algoritmo, da chave pública e amostras do texto criptografado precisam ser insuficientes para determinar a outra chave. |

## 2.3 Internet das Coisas

O conceito IoT, proposto por Kevin Ashton, em 1999, no Instituto de Tecnologia de Massachusetts (MIT), e refere-se à conexão de objetos – geralmente sensores – com a Internet. Esse conceito pode ser considerado a evolução da Internet convencional tendo em vista sua capacidade de reunir, analisar e distribuir dados armazenados na web para dispositivos com conexão remota (EVANS, 2011).

O número de dispositivos conectados à internet cresce exponencialmente (EL-HAII et al., 2018). Já é possível encontrar casas automatizadas com dispositivos inteligentes, todos conectados e controlados via Internet.

No entanto, as aplicações da IoT não estão restritas às casas. Seu uso pode ser visto também nas áreas de previsão de desastres, saúde, transporte público, industria 4.0 e serviços automotivos, como carros autônomos (LOPES JUNIOR et al., 2019). A Figura 2.10 esquematiza os impactos que a IoT pode ter no cotidiano das pessoas e dos negócios.

Tabela 2.4: Comparação dos algoritmos.

| Algoritmo | Indicado para IoT | Considerações |
|---|---|---|
| DES | Não | Demonstra fraquezas. Já que é considerado um algoritmo não seguro, ineficiente para proteger a informação. |
| 3DES | Sim | Apesar da lentidão, é implementado em dispositivos IoT por ter uma segurança satisfatória. Ainda não foi substituído por completo pelo AES. |
| AES | Sim | É o algoritmo recomendado pelo governo dos Estados Unidos e padronizado pela ISO. Apresenta um desempenho melhor que o 3DES. A segurança desse algoritmo é maior, dado que utiliza chaves de criptografia maiores. |
| RSA | Não | Algoritmo lento para encriptação, não recomendado para dados que exigem velocidade de transmissão. |
| ECC | Sim | Muito utilizado por atingir um nível de segurança atrativo com chaves de apenas 256 bits. Possui ainda um desempenho computacional que consome menos energia, ideal para dispositivos de IoT. |
| LWC | Sim | A LWC apresenta algoritmos especialmente projetados para IoT. A segurança pode ser algo que não os torne tão recomendados. No entanto, há uma melhora no desempenho quando comparados com os algoritmos tradicionas. |

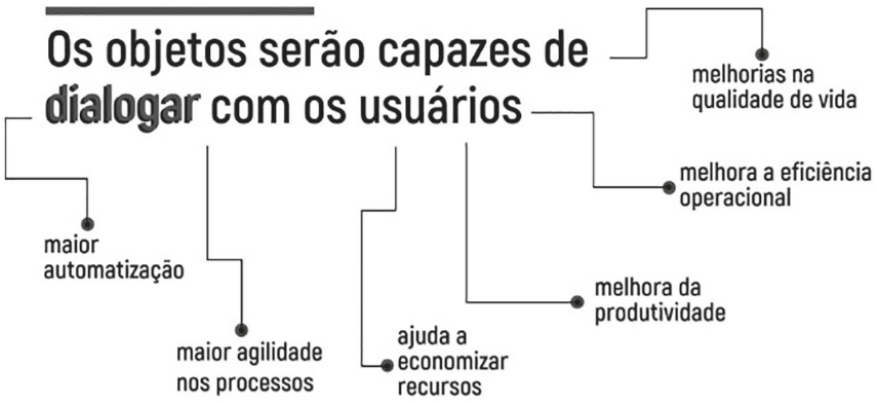



Figura 2.10: Impactos da IoT. Fonte: (PINHEIRO, 2021).

O descontrole do crescimento de dispositivos conectados à Internet levanta desafios devido às limitações e requisitos das aplicações. Exemplos desses desafios são a conectividade para

que bilhões de dispositivos se comuniquem, os desafios da segurança e do consumo de energia, impossibilitando a implementação de mecanismos criptográficos tradicionais e protocolos de segurança (EL-HAII et al., 2018).

Os desafios de segurança encontrados nos dispositivos para a IoT estão relacionados com a integridade, confidencialidade e disponibilidade dos dados e também a verificação e aprovação. A integridade dos dados é implementada durante o projeto do banco de dados através de procedimentos e regras com padrões específicos. Um projeto equivocado permite que terceiros capturem ou alterem os dados de forma maliciosa, prejudicando todo sistema.

Quanto à confidencialidade, utiliza-se o controle de acesso e a criptografia. A criptografia é a técnica utilizada para o sigilo e integridade dos dados, enquanto o controle de acesso é o método para controle das entidades que acessam os dados (PALLAVI; KUMAR; SRIKRISHNA, 2020).

De acordo com Moraes e Hayashi (2021) a arquitetura da IoT segue o padrão apresentado na Figura 2.11. As três partes são explicadas a seguir.

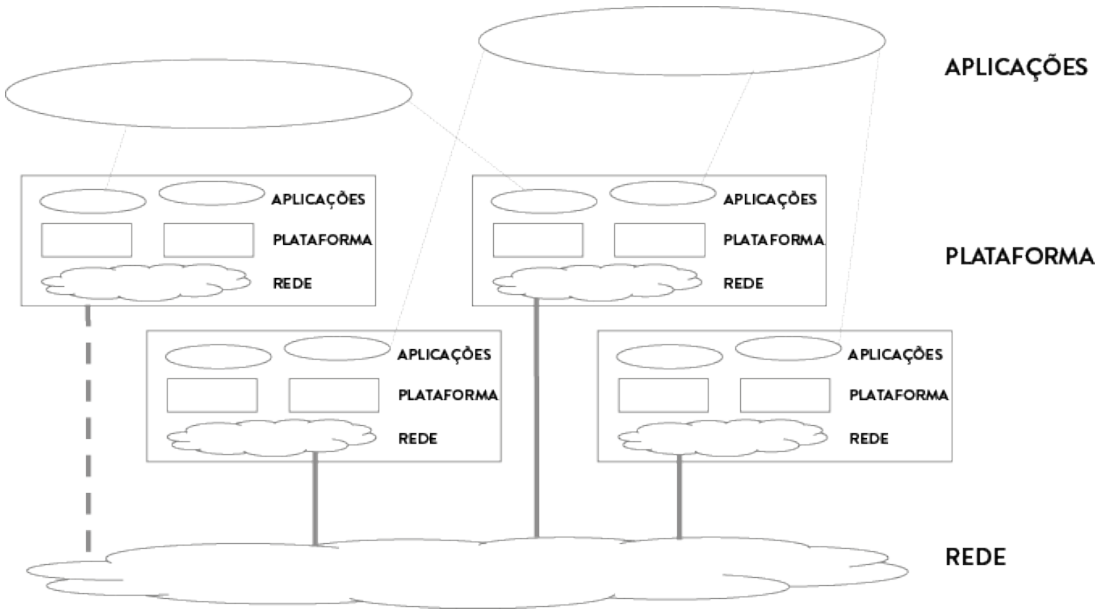



Figura 2.11: Arquitetura IoT. Fonte: (MORAES; HAYASHI, 2021).

### 2.3.1 Camada de rede

Na arquitetura da IoT, os dispositivos também são conhecidos como objetos e são responsáveis pela coleta de informações. Esta camada digitaliza e transfere os dados através de canais de comunicação seguros para sistemas de tratamento de dados (MORAES; HAYASHI, 2021).

Os dispositivos IoT normalmente utilizam kits minicomputadores ou microcontroladores como Arduino, Raspberry e ESP. A Tabela 2.5 apresenta um pouco das características de alguns kits pré-montados.

Tabela 2.5: Comparação dos kits pré montados de processadores.

| **Kit pré-montado** | **Características** |
|---|---|
| Arduino Nano 33 IoT | Microchip SAMD21G18A Arm Cortex-M0+ microcontroller operando em até 48 MHz. Memória RAM de 32 KB. Possui conectividade WiFi, Bluetooth e BLE. |
| Raspberry PI 4 model B | Possui memória RAM de até 4 GB. Conectividade 2.4 GHz e 5.0 GHz IEEE 802.11ac *wireless*, Bluetooth 5.0, BLE Gigabit Ethernet. Processador quad-core 64-Bit rodando a 1.5GHz. |
| ESP 32 | Possui dois Microprocessadores Xtensa 32-bit LX6 com até 600 DMIPS. Memória RAM estática interna de 520 KB. Interface Bluetooth v4.2 BR/EDR, BLE e WIFI, até 150 Mbps. |

Os avanços da inteligência artificial permitiram que novos dispositivos se conectassem à camada de rede. Um exemplo pode ser os sistemas de resposta audível, como a Alexa da empresa Amazon. Esses dispositivos se encaixam em dispositivos IoT nesta camada, com a adição de uma aplicação embarcada (MORAES; HAYASHI, 2021).

### 2.3.2 Plataforma

A plataforma é responsável por transmitir os dados da camada de rede para as aplicações. Para essa conexão podem ser usadas tecnologia e protocolos como ZigBee, WiFi, Bluetooth e *Bluetooth Low Energy* (BLE), entre outras. Essas tecnologias possibilitam a transmissão de dados com baixo consumo de energia e alta eficiência em curtas distâncias (PINHEIRO, 2021). A Tabela 2.6 compara essas tecnologias.

### 2.3.3 Camada de aplicação

Na camada de aplicação, usam-se os dados processados na camada de transporte. Normalmente, as aplicações estão hospedadas na nuvem computacional e são acessadas via aplicativos ou via *web* (MORAES; HAYASHI, 2021).

Tabela 2.6: Comparação das tecnologias de acesso. Fonte: adaptado de Maschietto et al. (2021).

| Tecnologia | Padrão | Frequência | Alcance | Bateria |
|---|---|---|---|---|
| ZigBee | IEEE 802.15.4 | 20 a 250 Kbps | + de 100 metros | Meses/anos |
| WiFi | IEEE 802.11ax | 9,6 Gbps | + de 100 metros | Dias |
| Bluetooth | IEEE 802.15.1 | 3 Mbps | Entre 1 e 10 metros | Dias |

A função da camada de aplicação é também o armazenamento, processamento e análise dos dados dos objetos, a fim de extrair informações relevantes para o contexto que está sendo aplicada a solução (DE PAULA, 2015).

## 2.4 Considerações finais

A conectividade e a acessibilidade da IoT evoluem constantemente. Podemos afirmar que a IoT é um conjunto de tecnologias capaz de reduzir custos na saúde, auxiliar na desobstrução do tráfego, minimizar a emissão de gases poluentes e muitos outros benefícios que proporcionam uma melhor qualidade de vida (LOPES JUNIOR et al., 2019).

Regulamentos e padrões impulsionam a adoção de dispositivos conectados. Além disso, questões como a segurança dos dados que esses dispositivos carregam têm recebido mais atenção, com vistas à melhoria da confiabilidade do sistema. Existem duas maneiras de fornecer segurança para dispositivos inteligentes por meio de cifras: a partir da implementação de mecanismos no hardware ou a partir da implementação de software (SINGH, P.; ACHARYA; CHAURASIYA, 2019).

Por sua vez, o uso da tecnologia criptográfica está vinculado às normas e às leis nacionais. A administração da distribuição de chaves requer proteção e sigilo através de um sistema de gerenciamento de operação e manutenção de segurança. São necessários recursos de hardware para ter capacidade de acelerar as operações de criptografia. Além disso, os recursos computacionais devem ter boa escalabilidade, para evitar desperdício de recursos (XU, Y. et al., 2018).

O Capítulo 3 a seguir apresenta um levantamento bibliográfico da literatura científica. O objetivo é para analisar o estado da técnica e situar esta pesquisa em relação a outras.

# Capítulo 3

# Levantamento bibliográfico

Neste capítulo são apresentados os trabalhos precursores sobre a criptografia utilizada em IoT. Dividimos os trabalhos em dois grupos: aqueles que estudam a dificuldade de implementação de algoritmos criptográficos em dispositivos com limitações de memória; e os estudos que implementam algoritmos leves em dispositivos IoT.

Todos os trabalhos citados neste capítulo são focados em solucionar problemas de segurança e confidencialidade em dispositivos IoT. Os levantamentos estão organizados da seguinte maneira:

- **Pesquisas Precursoras:** levantamento e análise em ordem cronológica das pesquisas realizadas até 2016, com ênfase para trabalhos publicados no período entre 2014 a 2016.
- **Estado da arte:** implementação de algoritmos criptográficos em dispositivos para IoT, considerando trabalhos publicados a partir de 2016, apresentados cronologicamente.

## 3.1 Pesquisas precursoras

Na área da Segurança da Informação, com frequência o usuário é visto como a parte fraca do sistema. Heartfield e Loukas (2015) mostram como mesmo os sistemas de proteção que seguem as recomendações da ISO podem ser contornados através da manipulação do usuário. Os autores definem os ataques que manipulam a interface computador/usuário e violam a segurança de sistemas computadorizados como ataques semânticos.

Os ataques semânticos são normalmente agrupados por tipo de exploração em ataques especificamente relacionados. Por exemplo, *sites* e e-mails falsos são classificados como

ataques *phishing*, onde as vítimas são atacadas através de um mecanismo de isca. As operações atuais de crimes cibernéticos podem ter muitas camadas e um adversário pode empregar ataques semânticos compostos que consistem em várias fases dos ataques semânticos individuais, em paralelo ou um após o outro (HEARTFIELD; LOUKAS, 2015).

Conforme Heartfield e Loukas (2015) discutem, a partir das características centrais dos ataques semânticos é possível projetar um sistema de defesa aplicável a todas as categorias de ataque que compartilham as mesmas características. Os autores ainda defendem a combinação de contramedidas técnicas e de gerenciamento de usuários em um único sistema autônomo, onde os usuários são perfilados de acordo com suas vulnerabilidades a determinadas classes de ataques. Além disso, são alocadas permissões aos usuários de acordo com suas classes ou seus perfis.

Por sua vez, Büsching e Wolf (2015) demonstram a importância da segurança em pequenos dispositivos de detecção e computação que aprimoram os cuidados médicos e o bem-estar dos seres humanos. Esses dispositivos são chamados de *Body Area Networks* (BAN). Nesses dispositivos, os autores defendem o uso de cifras de uso único (*One-Time Pads* – OTPs) para proteger e garantir uma transmissão de dados segura e duradoura.

Büsching e Wolf (2015) consideram três cenários diferentes do uso de memória pela criptografia OTP em nós sem fio. No primeiro cenário, os dados brutos registrados são criptografados e transmitidos para um coletor. Assim, toda memória disponível pode ser preenchida com OTPs e as OTPs podem ser marcadas como usadas ou excluídas. Em seguida, a memória pode ser liberada.

Se os dados criptografados não podem ser transmitidos de imediato, então eles podem ser armazenados no nó. Esse é o segundo cenário, onde é realizada uma criptografia *online* imediata dos dados gravados do sensor. Os dois cenários são utilizados em caso de coleta de dados puros. O terceiro cenário é a pior das hipóteses, quando os todos os dados têm que estar disponíveis em texto claro no nó (BÜSCHING; WOLF, 2015).

Nesse caso, ao realizar um processamento no nó de todos os dados e a transmissão são sincronizados, apenas metade da memória pode ser usada para armazenar OTPs. A outra metade é usada para armazenar dados não criptografados em *cache*. Em suma, a pesquisa de Büsching e Wolf (2015) mostra que a capacidade de memória disponível nos dispositivos, dependendo da taxa de dados gerada, é suficiente para armazenar uma grande quantidade de OTPs pré-gerados.

Em seguida, Machado et al. (2016) trazem a importância da aplicação da segurança de dados em redes elétricas inteligentes. O setor de fornecimento de energia está passando por uma digitalização com adoção de tecnologias de informação e comunicação nas operações do setor. As *Smarts Grids* são descritas por Machado et al. (2016) como redes elétricas que utilizam tecnologias digitais avançadas para monitorar e gerenciar o transporte de eletricidade de todas as fontes de geração, aumentando a confiabilidade, segurança e eficiência energética.

O método utilizado por Machado et al. (2016) consiste em identificar um caso de uso, levantar ou identificar os ativos, ameaças e impactos e, por fim, analisar o cumprimento e aplicação dos requisitos de segurança. A avaliação do nível de maturidade de segurança cibernética em redes inteligentes que os autores propõem é flexível e pode ser aplicada em outros sistemas inteligentes.

Por sua vez, Meng Xu et al. (2016) estudam as vulnerabilidades do Android – uma plataforma popular para dispositivos móveis que impulsiona a IoT – para suavizar as ameaças dos ataques. No estudo, algumas soluções específicas para problemas são propostas, como a proteção do *kernel*, as análises dinâmica ou estática de programa e a detecção de reempacotamento de aplicativos.

No entanto, os autores definem a partir das soluções existentes – como o redesenho do modelo de permissões, a reengenharia do sistema e os procedimentos de atualização do aplicativo –, a criação de novos algoritmos de análise de intenção do usuário para a detecção de *malware*, e alguns problemas mais fundamentais, como a baixa adoção das soluções existentes.

A abordagem proposta por Meng Xu et al. (2016) é composta por três partes: a construção da infraestrutura da *Aplication Programming Interface* (API) aberta de serviço; o compartilhamento e a consolidação dos dados relacionados à segurança por meio de um mecanismo *Open Ranking*[1]; e a promoção de funções igualitárias, compartilhando o *Security Test Suite*[2] e nivelando o fluxo de atualizações.

O modelo de API apresentado por Meng Xu et al. (2016) é do tipo colaborativo para atrair mais participantes para a segurança do ecossistema Android. Através do acesso aberto, inúmeras entidades podem colaborar com a avaliação de segurança dos aplicativos carregados. Este é o mesmo princípio da segurança de código aberto, que afirma que mais

[1]Pontuações avaliadas compartilhadas em canais abertos, consumidas por usuários finais.

[2]Sistema que testa a eficácia e a compatibilidade de cada atualização de segurança com o maior número de modelos de dispositivos possíveis.

vulnerabilidades e erros podem ser detectados quanto mais pessoas têm acesso ao código[3]. Os autores colocam ainda que as pontuações de segurança avaliadas ficam armazenadas no *ranking* aberto, compartilhado em todo o ecossistema e consumido pelos usuários finais e o *Security Test Suite*.

## 3.2 Estado da arte

Nesta seção são considerados os levantamentos e análises dos últimos trabalhos, desde 2016 até o presente, destacando o uso de algoritmos criptográficos em dispositivos IoT. Subdivide-se esta seção em trabalhos que analisam o desempenho de algoritmos genéricos para IoT e algoritmos de criptografia leve. Além dos algoritmos, destaca-se o desempenho que os dispositivos apresentam durante os estudos.

### 3.2.1 Algoritmos genéricos para IoT

O trabalho de Nuzhat Khan et al. (2017) utiliza um Raspberry PI 3 modelo B para simular um ambiente restrito de IoT junto com duas bibliotecas criptográficas, a FLECC_IN_C (WENGER; UNTERLUGGAUER; WERNER, 2017) e a Crypto++ (DAI, 2021). Os autores contribuem para o entendimento e estabelecem diretrizes para projetos de algoritmos para dispositivos IoT, facilitando a escolha de um algoritmo para um dispositivo específico, por meio do tamanho de chave ou através das comparações gráficas de desempenho que os algoritmos apresentaram no ambiente proposto.

Yingxin Xu et al. (2018) destacam soluções de segurança existentes, propostas para fornecer segurança da informação na nuvem computacional. Além disso, apresentam uma estrutura de *pool* de recursos de segurança reconfigurável para fornecer serviços de segurança sob demanda. Os autores tratam a importância da segurança da informação à medida que mais aplicativos estão migrando para a nuvem computacional.

Na plataforma OpenStack, Yingxin Xu et al. (2018) projetaram um protótipo de sistema, com soluções que melhoram a segurança e confidencialidade do sistema, com base na melhoria do desempenho do serviço. A segurança do *pool* de recursos seguros e reconfiguráveis é dividida em segurança externa – que impede acesso ilegal, ou seja, evita riscos externos – e segurança interna, que garante a integridade.

[3] Também conhecido como Lei de Linus: "*Given enough eyeballs, all bugs are shallow*".

Em seguida, Muthavhine e Sumbwanyambe (2018) analisam os problemas presentes nos algoritmos criptográficos utilizados na IoT. Eles utilizaram algoritmos criptográficos de bloco existentes para autenticação e criptografia de diferentes dispositivos empregados na IoT. Os algoritmos utilizados foram os seguintes: AES, Blowfish, Camelia, Cast-128, Clefia, DES, MMB, RC5, Serpent, Skijack.

No modelo proposto pelos autores, é possível aumentar o efeito avalanche para 60% dos algoritmos testados, quando a chave é corrigida, e para 70%, quando o texto claro é corrigido. A velocidade quando a chave é corrigida aumentou nos algoritmos AES e Clefia. Quando o texto claro é corrigido, o modelo aumenta a velocidade dos algoritmos AES, Camellia, DES e MMB.

Uma aplicação de algoritmos criptográficos mais frequente é vista no trabalho de Ahtsham, Yan e Ali (2019), que apresenta um sistema de fechadura de porta seguro baseado em senha e blindado criptograficamente. O sistema completo inclui um aplicativo para *smartphone* Android, a implementação dos algoritmos criptográficos AES-128 e SHA-512 para comunicação segura e um *hardware* programável com sensores e atuadores para controlar o acesso não autorizado.

### 3.2.2 Algoritmos de criptografia leve

Especificamente sobre os algoritmos de criptografia leve, o trabalho de Tausif et al. (2017) avalia treze cifras de bloco leves usadas para comunicação segura na IoT. As cifras são: AES-128, Fantomas, HIGHT, L Block, *Light Encryption Device* (LED), Piccolo, Present Cipher, PRINCE, RC5, Rob, Simon, Speck e TWINE. Os autores comparam as cifras com base no tamanho do código, uso da memória RAM e tempo de execução. Tudo isso em três plataformas diferentes: 8 bits, 16 bits e 32 bits.

Os resultados obtidos mostram que o projetista deve sempre lembrar os pré-requisitos do algoritmo ao implementá-lo. A intenção deve demandar o mínimo possível dos recursos do dispositivo (TAUSIF et al., 2017). Os autores usam a cifra planejada, que mantém as chaves fixas. Isso facilita a programação ocupando um espaço pequeno nos dispositivos. Como os algoritmos são implementados diretamente no dispositivo, não há necessidade de redigitação. No entanto, os autores frisam que o desafio principal está em alcançar um equilíbrio ótimo entre os parâmetros: custo, segurança e desempenho.

Por sua vez, El-Haii et al. (2018) analisam o desempenho de alguns algoritmos criptográficos (simétricos, assimétricos, *Hash* e assinatura digital) no Raspberry PI 3 modelo B. Os autores comparam os custos básicos do dispositivo, fornecendo evidências para determinar a sobrecarga que um protocolo de segurança que usa primitivas criptográficas pode introduzir em um sistema.

Dos resultados que El-Haii et al. (2018) apresentam, destacam-se as seguintes conclusões: (i) o melhor algoritmo simétrico para Raspberry Pi 3 quando o consumo de energia é analisado é o AES; (ii) os algoritmos de *Hash* apresentam uma taxa de transferência semelhante aos algoritmos simétricos; e (iii) quando analisados os algoritmos de geração/troca de chaves, a *Elliptic Curve Diffie-Hellman* alcança um desempenho melhor que o RSA. Em suma, o uso de algoritmos criptográficos em sistemas em rede indica que o problema não está nos custos computacionais para a criptografia, mas sim no tempo total de processamento do protocolo que envolve o consumo de energia.

Lopes Junior et al. (2019), em contrapartida, aplicam algoritmos criptográficos de cifra leve (AES, Simon, Speck, Curupira1 e Curupira2) no microcontrolador ESP8266 para simular o ambiente IoT. Nesse trabalho, a análise se baseia na avaliação do tempo de execução, consumo de energia e vazão (*throughput*). Os algoritmos Curupira (1 e 2) abordados no trabalho demonstraram ser mais econômicos em relação ao consumo de energia e alcançaram maior velocidade no processo de encriptação/decriptação. No entanto, o algoritmo AES destacou-se em outros aspectos como maior vazão, ou seja, maior capacidade de criptografar dados e informações computacionais, maior estabilidade em relação ao tempo de execução, a memória utilizada e o gasto energético. Essas características proporcionam maior segurança pela criptografia aplicada.

Para melhorar a segurança da informação e o desempenho de processamento no *hardware* da IoT, Wang et al. (2019) propõem alguns algoritmos que suportam o sistema de criptografar e descriptografar reconfigurável dinâmico (*Dynamic Reconfigurable Encryption and Decryption System* – DREDS) com alta flexibilidade e alta utilização de recursos, bem como a implementação da série ZYNQ-7000 da Xillinx *Field Programmable Gate Array* (FPGA).

Wang et al. (2019) implementaram os algoritmos AES-128, 3DES e o *Upper Computer Control Software* projetado para controlar a comutação de algoritmos e aquisição de dados. A análise dos dados e a comparação com o sistema de criptografia e descriptografia estática, mostra que o sistema proposto pelos autores reduz o consumo de recursos no *chip* e garante

que o sistema de criptografia e descriptografia complete a troca do tipo do algoritmo em alta velocidade de configuração em um estado de trabalho estável.

Já Pulkit Singh, Acharya e Chaurasiya (2019) limitam o escopo de sua pesquisa em busca de soluções criptográficas simétricas, leves e modernas empregadas para proteger dispositivos de comunicação de pequena escala. Os autores abordam os requisitos, procedimentos, abordagens, técnicas e tendências para facilitar a implementação de cifras leves com recursos limitados para ambientes com restrição de recursos, auxiliando pesquisadores e projetistas na escolha do equilíbrio adequado entre desempenho, custo e segurança.

Em seguida, Beg, Al-Kharobi e Al-Nasser (2019) simula – dentro do Cooja, descrito na Seção 4.2 – e compara os algoritmos AES e PRESENT usando métricas relevantes para um ambiente de IoT, como tempo de criptografia/descrptografia, tempo gasto para enviar uma mensagem, consumo de energia e tamanho de código. Apesar de o AES ser considerado em muitos trabalhos o mais recomendado para IoT, os resultados obtidos por Beg, Al-Kharobi e Al-Nasser (2019) mostram o Present como o mais indicado para configurações de IoT, pois esse algoritmo apresentou um desempenho melhor no tempo gasto para criptografia e descrptografia. Ainda, esse algoritmo consome menos energia com um tamanho de implementação menor, ideal para dispositivos IoT.

Já Pindar et al. (2020) analisaram dispositivos inteligentes e suas necessidades para garantir a confidencialidade e integridade da troca de informações, considerando um esquema leve que equilibra segurança e desempenho. Eles implementaram o algoritmo MAC, baseado na função de transformação de *string* de permutação-quasigrupo no Raspberry PI B+. Os resultados do trabalho mostram que para o *hashing* de mensagens de tamanho 32 bits, 64 bits e 128 bits são utilizados respectivamente 3,73%, 3,55% e 2,88% dos recursos da CPU, ou seja, o algoritmo mostrou-se eficiente no dispositivo.

Dhanda, Brahmjit Singh e Jindal (2020) apresentam um estudo aprofundado sobre criptografia leve como solução para o problema de segurança de dispositivos com restrição de recursos em IoT, fornecendo uma pesquisa detalhada e das primitivas criptográficas leves disponíveis até 2019. Os autores avaliaram 21 cifras de bloco leve, 19 cifras de fluxo leves, nove funções hash leves e cinco variantes de ECC. Essa avaliação considerou a área do chip, energia e potência, eficiência de *hardware* e *software*, latência e figura de mérito[4].

[4]A figura de mérito considera o quociente do rendimento, $T$, pela área de implementação, $A$. Assim, figura de mérito $= \frac{T}{A}$.

No entanto, apesar do desenvolvimento contínuo das cifras, Dhanda, Brahmjit Singh e Jindal (2020) concluem que o AES continua sendo a escolha mais recomendada no quesito segurança. Além disso, é a cifra mais competitiva entre as cifras de bloco. Segundo os autores, na criptografia assimétrica, a ECC continua sendo a opção para fornecer confidencialidade, autenticação e não repúdio.

Por sua vez, Muhammad Nauman Khan, Rao e Camtepe (2020) descrevem a estrutura da IoT, os recursos computacionais dos dispositivos nas plataformas de ponta, embarcadas e nuvem computacional, e classificam os protocolos leves existentes. O trabalho faz ainda uma análise comparativa das soluções criptográficas existentes, junto com suas vantagens, desvantagens e vulnerabilidades. Apesar de vários estudos que alcançam a confidencialidade usando a criptografia leve, os autores destacam que garantir confidencialidade das informações continua sendo um desafio.

Já Kane et al. (2020) oferecem um *framework* para analisar métricas de desempenho como as principais áreas de consumo de energia, custos de tempo e de energia, de dispositivos microcontrolados, comumente utilizados na IoT. Uma revisão das cifras AES, ChaCha e Acorn executadas nos dispositivos microcontrolados STM32F103C8T6, ATmega328 e ESP8266 Wi-Fi *Witty Cloud Development Board* foi realizada para demonstrar o uso do *framework*, testando operações de criptografia, descrptografia e configuração de chave.

Dos dispositivos utilizados, o que atingiu o melhor equilíbrio entre desempenho e velocidade foi o STM32F103C8T6. Nesse dispositivo, os algoritmos ChaCha e Acorn são substancialmente mais rápidos que o AES e ainda, consomem menos energia. Por isso, Kane et al. (2020) recomendam considerá-los para usos de criptografia leve.

Por fim, um trabalho mais atual de Wu, Dwivedi e Srivastava (2021) propõe um método baseado na tecnologia *blockchain* para proteger informações de sistemas médicos. O método é testado nos aspectos de eficiência de armazenamento e transmissão de informações e desempenho de controle de segurança. A tecnologia *blockchain* pode garantir a rastreabilidade dos registros armazenados a qualquer momento, mas podem ocorrer problemas de segurança, como a exposição de chaves na transmissão. Por este motivo, Wu, Dwivedi e Srivastava (2021) aplicam a troca de chaves de Diffie-Hellman baseada em curva elíptica às suas informações de armazenamento interno. Os resultados comprovam que o método é eficiente no armazenamento e transmissão de informações e possui alto desempenho de controle de segurança da informação (WU; DWIVEDI; SRIVASTAVA, 2021).

## 3.3 Considerações finais

Neste capítulo foram apresentadas as pesquisas precursoras para o tema criptografia em IoT. Realizamos o levantamento bibliográfico em ordem cronológica, destacando a aplicabilidade de alguns algoritmos e estudos sobre segurança em dispositivos inteligentes. Para tanto, avaliamos os trabalhos desenvolvidos a partir de 2015. O Apêndice A mostra com mais detalhes os parâmetros utilizados para o levantamento bibliográfico.

Como destaque deste capítulo, a Tabela 3.1 apresenta os principais algoritmos presentes nos trabalhos avaliados. A última linha aponta quais algoritmos este trabalho se concentrou. O Capítulo 4 a seguir apresenta os materiais e métodos utilizados neste trabalho.

Tabela 3.1: Resumo principais algoritmos abordados nos trabalhos.

| **Trabalhos** | **3DES** | **AES** | **DES** | **Klein** | **Piccolo** | **Present** | **Prince** | **PRINT** | **Rectangle** | **Simon** | **Speck** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Tausif et al. (2017) | | X | | | X | X | X | | | X | X |
| Muthavhine e Sumbwanyambe (2018) | | X | X | | | | | | | | |
| El-Haii et al. (2018) | | X | X | | | | | | | | |
| Lopes Junior et al. (2019) | | X | | | | | | | | X | X |
| Wang et al. (2019) | X | X | | | | | | | | | |
| Pulkit Singh, Acharya e Chaurasiya (2019) | | | | X | X | X | X | X | X | X | X |
| Ahtsham, Yan e Ali (2019) | | X | X | | | | | | | | |
| Beg, Al-Kharobi e Al-Nasser (2019) | | X | | | | X | | | | | |
| Dhanda, Brahmjit Singh e Jindal (2020) | | X | | X | X | X | X | X | X | X | X |
| Pallavi, Kumar e Srikrishna (2020) | | | | X | | X | | X | X | | |
| Kane et al. (2020) | | X | | | | | | | | | |
| Wu, Dwivedi e Srivastava (2021) | X | X | | | | | | | | | |
| Algoritmos que foram testados neste trabalho | | X | | | | X | | | | X | X |

# Capítulo 4

# Materiais e Métodos

Este capítulo aborda os materiais e métodos que foram aplicados na parte experimental do trabalho. É importante recapitular que o objetivo da pesquisa é avaliar os algoritmos criptográficos para utilizá-los em dispositivos IoT, ou seja, dispositivos com poder computacional limitado.

Portanto, a Seção 4.1 apresenta os algoritmos que serão implementados e analisados. Em seguida, na Seção 4.2 está a descrição e características do ambiente escolhido para simular dispositivos IoT. Por fim, os critérios de avaliação que serão utilizados para comparar o desempenho dos algoritmos constam na Seção 4.3.

## 4.1 Algoritmos

Os algoritmos que serão implementados e avaliados são o AES, o Present, o Simon e o Speck. Para a escolha dos algoritmos consideramos os estudos apresentados no Capítulo 3.

O algoritmo AES foi o que demonstrou maior ocorrência nos trabalhos levantados. Sendo assim, pode ser considerado um algoritmo padrão, bom para comparações. Os demais algoritmos – Present, Simon e Speck – também são, conforme o levantamento bibliográfico, os que mais se destacaram quando o foco é IoT. Assim, foram definidos como os algoritmos a serem estudados, implementados e analisados nesta pesquisa.

As seções a seguir destacam algumas características importantes desses algoritmos, exceto o algoritmo AES, já mencionado no Capítulo 2 deste trabalho.

### 4.1.1 Present

A cifra Present foi desenvolvida para dispositivos com recursos limitados (PAAR; PELZL, 2009). Essa cifra está disponível na rede de substituição-permutação (SPN) com 31 rodadas. A rede SPN consiste em operações aritméticas e lógicas simples, com funções de deslocamentos circulares à esquerda, XOR bit-a-bit e AND bit-a-bit (TAUSIF et al., 2017). A cifra Present criptografa blocos com comprimento de 64 bits, com chaves de 80 bits e 128 bits (SINGH, P.; ACHARYA; CHAURASIYA, 2019).

A Figura 4.1 apresenta um resumo esquemático de como o algoritmo Present funciona. Em cada rodada, é realizada uma operação XOR utilizando a chave da rodada, seguido da aplicação das duas funções sBoxLayer e pLayer. A função sBoxLayer consiste em realizar substituições não lineares dos elementos de state por elementos correspondente da tabela S-box do Present. A operação pLayer consiste em permutar a posição de cada elemento gerado anteriormente, chamado de vetor Estado.

```
gerarChavesRodada();
for i = 1 até 31 do
    addRoundKey();
    sBoxLayer();
    pLayer();
end for
addRoundKey();
```

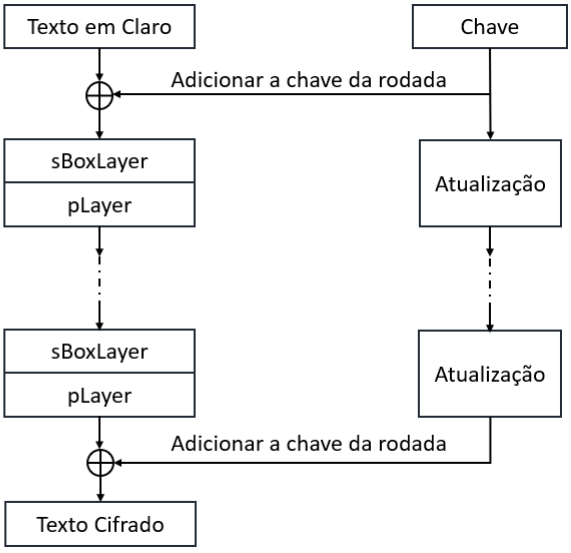



Figura 4.1: Algoritmo Present. Adaptado de (BOGDANOV et al., 2007).

### 4.1.2 Simon e Speck

Ambas as cifras, Simon e Speck, são cifras de bloco leve. São adequadas para aplicações em IoT, sistemas embarcados e cenários onde o *hardware* é limitado quanto ao processamento, consumo de energia e memória (LOPES JUNIOR et al., 2019).

Simon e Speck são baseados na estrutura de Feistel e em redes de substituição-permutação (LOPES JUNIOR et al., 2019). Uma estrutura de Feistel permite obter

uma permutação computável e invertível a partir de uma função (BARBOSA; FARSHIM, 2015). Além da força criptográfica, outra vantagem das estruturas de Feistel é que a criptografia e a decriptografia apresentam operações similares (PAAR; PELZL, 2009).

Beaulieu et al. (2013) apresentam um pseudocódigo completo, ilustrado nas figuras 4.2 e 4.3, com instruções de como devem ser implementados os algoritmos Simon e Speck para a criptografia de mensagens.

```
------------------------- definições -------------------------
  n = word size (16, 24, 32, 48, or 64)
  m = number of key words (must be      4 if n = 16,
                                    3 or 4 if n = 24 or 32,
                                    2 or 3 if n = 48,
                                 2, 3, or 4 if n = 64)

  z = [11111010001001010110000111001101111101000100101011000011100110,
       10001110111110010011000010110101000111011111001001100001011010,
       10101111011100000011010010011000101000010001111110010110110011,
       11011011101011000110010111100000010010001010011100110100001111,
       11010001111001101011011000100000010111000011001010010011101111]

  (T, j) = (32,0)                    if n = 16
         = (36,0) or (36,1)          if n = 24, m = 3 or 4
         = (42,2) or (44,3)          if n = 32, m = 3 or 4
         = (52,2) or (54,3)          if n = 48, m = 2 or 3
         = (68,2), (69,3), or (72,4) if n = 64, m = 2, 3, or 4

  x,y          = plaintext words
  k[m-1]..k[0] = key words
----------------------expansão de chaves----------------------
for i = m..T-1
   tmp  ← S^-3 k[i-1]
   if (m = 4) tmp ← tmp ⊕ k[i-3]
   tmp  ← tmp ⊕ S^-1 tmp
   k[i] ← ~k[i-m] ⊕ tmp ⊕ z[j][(i-m) mod 62] ⊕ 3
end for
------------------------- encriptação -------------------------
for i = 0..T-1
   tmp ← x
   x   ← y ⊕ (Sx & S^8 x) ⊕ S^2 x ⊕ k[i]
   y   ← tmp
end for
```

Figura 4.2: Pseudocódigo Simon. Adaptado de (BEAULIEU et al., 2013).

```
------------------------ definições ------------------------
  n   = word size (16, 24, 32, 48, or 64)
  m   = number of key words (must be     4 if n = 16,
                                     3 or 4 if n = 24 or 32,
                                     2 or 3 if n = 48,
                                     2 or 3 or 4 if n = 64)

  T   = number       = 22             if n = 16
        of rounds    = 22 or 23       if n = 24, m = 3 or 4
                     = 26 or 27       if n = 32, m = 3 or 4
                     = 28 or 29       if n = 48, m = 2 or 3
                     = 32, 33, or 34  if n = 64, m = 2, 3, or 4

  (α,β) = (7,2)  if n = 16
          (8,3)  otherwise

  x,y                = plaintext words
  ℓ[m-2]..ℓ[0],k[0]  = key words
-----------------------expansão de chaves--------------------
for i = 0..T-2
   ℓ[i+m-1] ← (k[i] + S^-α ℓ[i]) ⊕ i
   k[i+1]   ← S^β k[i] ⊕ ℓ[i+m-1]
end for
-------------------------encriptação -------------------------
for i = 0..T-1
   x ← (S^-α x + y) ⊕ k[i]
   y ← S^β y ⊕ x
end for
```

Figura 4.3: Pseudocódigo Speck. Adaptado de (BEAULIEU et al., 2013).

## 4.2 Sobre o simulador

O sistema operacional Contiki OS[1] é de código aberto, feito para ser executado em microcontroladores de baixa potência. O sistema operacional suporta protocolos IPv4, IPv6 e protocolos da família 6LoWPAN. Ele é utilizado em diversos sistemas comerciais e não

[1]https://sourceforge.net/projects/contiki/files/

comerciais como medidores de energia elétrica em rede, iluminação pública, sistemas de alarmes e outros (CONTIKI, 2023).

A simulação apresenta neste trabalho foi realizada em um ambiente preparado na VMWare Workstation Player Versão 16[2], destinando 1 GB de memória RAM para construção e preparação do ambiente utilizando o Ubuntu na versão 12.04.2. Considerando o poder de processamento limitado, o número de sensores utilizados foi restrito a 10 sensores.

O simulador do Contiki OS, Cooja, foi o escolhido para simular os algoritmos selecionados em nós sensores com poder computacional limitado. O Cooja, Figura 4.4, é um simulador baseado em Java, projetado para redes de sensores sem fio. O aplicativo simula uma rede de nós sensores, onde cada nó pode ser diferente dos demais, tanto no *software* que contém, quanto no *hardware* que utiliza (OSTERLIND et al., 2006).

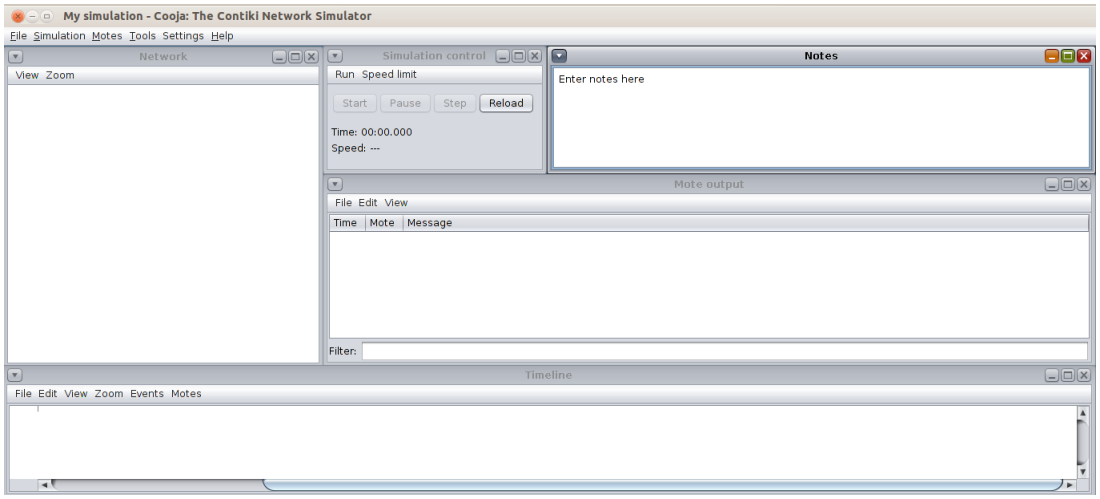



Figura 4.4: Interface do Simulador Cooja

Para auxiliar na captura de informações dos nós, o Cooja disponibiliza uma ferramenta chamada *Collect View*. Essa aplicação apresenta métricas e dados gerados a respeito do consumo de energia, tempo de entrega de pacotes, dentre outras leituras de dados de sensores.

Os *motes*[3] disponíveis no Cooja e seus respectivos microcontroladores são apresentados na Tabela 4.1. No entanto, os *motes* do tipo *sky* se destacam pela emulação que apresentam. Ou seja, *motes* do tipo *sky* são os que mais se aproximam da realidade de dispositivos IoT.

Para este estudo, os sensores foram criados utilizando *motes* do tipo *sky mote*. Nos *motes*, usamos o tipo *udp-sender* como base para a implementação dos algoritmos, para simular

[2]https://customerconnect.vmware.com/en/downloads

[3]O menor pedaço de *hardware* que suporta internet. Cada *mote* utilizado representa um dispositivo IoT na simulação.

Tabela 4.1: Plataformas Cooja

| ***Mote*** | **Microcontrolador** |
|---|---|
| MicaZ | ATmega128L + CC2420 (DUNKELS, 2018) |
| Eth1120 | MSP430 |
| Trxeb1120 e Trxeb2520 | MSP430F5438 (SQUARTINI et al., 2013) |
| Exp2420, Exp1101 e Exp1120 | MSP430F5438 |
| EXP430F5438 | MSP430F5438 |
| CC430 | MSP430 + CC1101 (PEDIREDLA et al., 2013) |
| Sky | MSP430 + CC2420 (DUNKELS, 2012b) |
| Wismote | MSP430 serie 5 + CC2520 (LASSO; CLARKE; NIRMALATHAS, 2018) |
| Z1 | MSP430F2617 (THÉBAUDEAU, 2019) |
| ESB | MSP430 (DUNKELS, 2012a) |

sensores comuns, enviando dados para a rede usando o *User Datagram Protocol* (UDP). Todas as simulações também incluíram um *mote* do tipo *udp-sink*, que simula um roteador.

*Motes* do tipo *udp-sender* tem como comportamento base de enviar informações ao *mote sink*. Em contrapartida, *motes* do tipo *udp-sink* recebem a comunicação de um ou mais *motes* do tipo *sender* que contêm suas informações de roteamento. Sendo assim, optou-se por utilizar apenas um *mote* do tipo *sink* para concentrar as informações geradas pelos motes do tipo *sender* em um roteador, facilitando a análise dos resultados.

A disposição dos sensores nas simulações aconteceu de forma aleatória. A Figura 4.5 apresenta um exemplo da alocação dos dispositivos no ambiente de simulação. Em amarelo estão representados os *motes* do tipo *udp-sender* e o *mote* "1", em azul, representa o *mote* do tipo *udp-sink*. O campo em verde mostra o alcance do roteador, no caso, o *mote udp-sink*. Como todos os *motes udp-sender* encontram-se dentro do campo de alcance e todos se comunicam corretamente com o *mote udp-sink*.

## 4.3 Critérios de avaliação dos algoritmos

Neste trabalho, vamos considerar os critérios de avaliação baseados nas métricas utilizadas para avaliar cifras de bloco leve. As métricas se subdividem em métricas para avaliação de implementações de *hardware* e para avaliação de implementações de *software*. A Figura 4.6

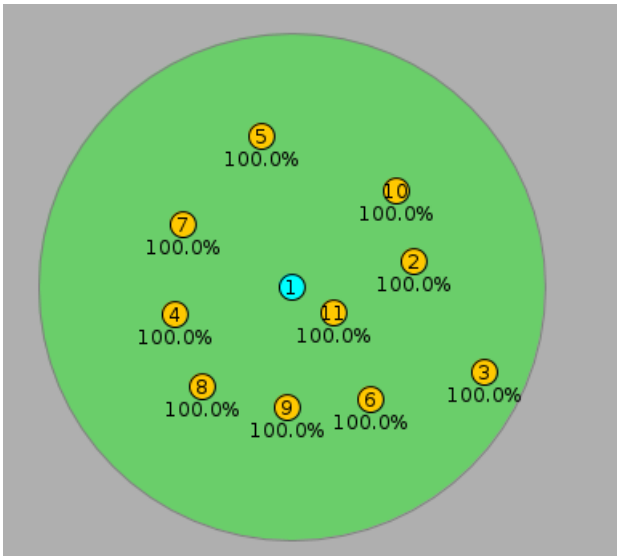



Figura 4.5: Disposição dos sensores

apresenta as métricas de desempenho. A métrica "Potência e Eenergia" é abordada na sequência. Essa foi a métrica a utilizada no desenvolvimento do trabalho.

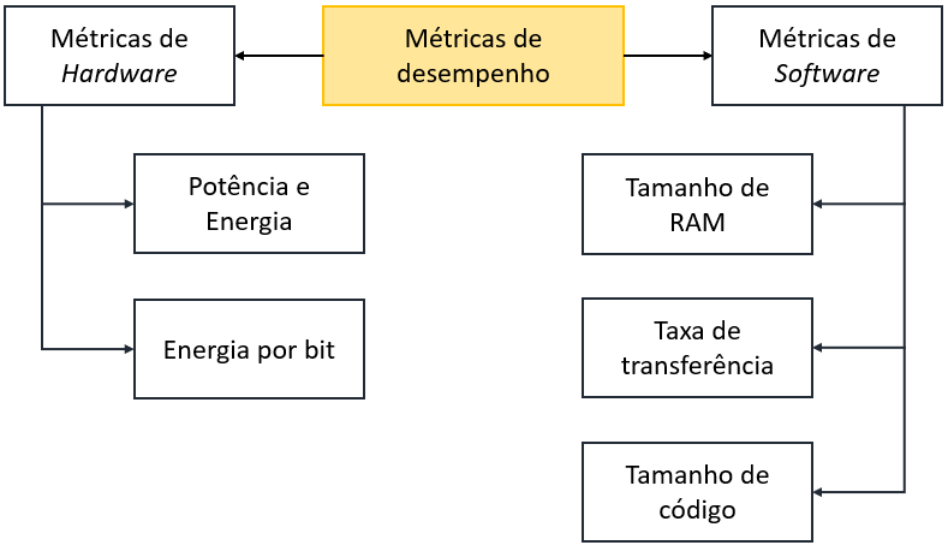



Figura 4.6: Métricas de desempenho. Adaptado de Pulkit Singh, Acharya e Chaurasiya (2019).

### 4.3.1 Potência e Energia

A potência média é definida como a quantidade de energia transferida por unidade de tempo. Em *hardware*, o consumo de energia aumenta conforme aumenta o tamanho das cifras. Minimizar a frequência do *clock* diminui a dissipação de energia. Em contrapartida, isso prejudica o rendimento. O consumo de energia é uma estimativa do consumo de potência ao

longo de um período. A energia reflete uma necessidade de recursos, enquanto a potência reflete uma característica de desempenho (SINGH, P.; ACHARYA; CHAURASIYA, 2019).

O consumo médio de energia em dispositivos IoT depende de quatro métricas: CPU, LPM, RX e TX. A métrica TX representa a energia necessária para um sensor transmitir seu sinal. Essa métrica é medida em decibel-miliwatt (dBm). O sinal recebido, por sua vez, é indicado pela métrica RX, também em dBm.

A métrica LPM representa o modo de pouca energia, ou seja, o gasto do dispositivo quando está ocioso, esperando algum evento. A métrica CPU, por sua vez, apresenta a energia gasta pela unidade central de processamento, isto é, o consumo de energia pelo microcontrolador do dispositivo.

## 4.4 Considerações finais

Neste capítulo, apresentamos os algoritmos selecionados nesta pesquisa, considerando a sua aplicação em dispositivos para IoT. Também apresentamos o simulador Cooja que representa os dispositivos no qual os algoritmos selecionados serão implementados. Além disso, descrevemos as métricas que serão utilizadas no critério de avaliação. O Capítulo 5 a seguir apresenta os resultados e discussões deste trabalho.

# Capítulo 5

# Resultados e discussões

Neste capítulo são apresentados os resultados desta pesquisa. Usando o Cooja, nós simulamos os algoritmos AES-128, Present, Simon e Speck. Todas as comunicações entre os dispositivos IoT com seus respectivos algoritmos foram simuladas no Cooja com *motes* do tipo *sky*. As informações foram coletadas com 10, 60 e 120 minutos de simulação com dez sensores.

Nas seções a seguir são apresentados os resultados dos algoritmos AES, Present, Simon e Speck, respectivamente. Por fim, a Seção 5.5 apresenta um resumo comparativo de todos os resultados obtidos.

As tabelas apresentas nas seções mostram os resultados dos *motes udp-sender*. O *mote udp-sink* não apresenta informações de consumo, pois é configurado para apenas coletar as informações e ser a base do *Collect view*, ferramenta utilizada para a coleta dos dados de consumo dos *motes*.

As figuras mostram os gráficos dos comportamentos dos *motes udp-sender*, ou seja, cada cor representa um sensor. Apesar de frequentes, os *outliers* não influenciaram nos resultados esperados e nas médias obtidas. As simulações não apresentaram nenhuma perda ou falha de comunicação com o *mote udp-sink*. Já a ocorrência dos *outliers* se deu devido a alguma demora na coleta de informações e não perda delas.

## 5.1 AES

Com o algoritmo AES foi simulado um cenário com 10 sensores (*motes*) mais um *mote sink*, responsável pela coleta das informações dos sensores. Neste cenário, foram observados os dados da Tabela 5.1 no minuto 10, da Tabela 5.2 no minuto 60 e da Tabela 5.3 no minuto 120.

Tabela 5.1: Consumo de energia no algoritmo AES – minuto 10.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| | 0,391 | 0,152 | 0,419 | 0,058 | 1,020 |
| | 0,365 | 0,152 | 0,399 | 0,063 | 0,979 |
| | 0,373 | 0,152 | 0,399 | 0,047 | 0,971 |
| | 0,376 | 0,152 | 0,393 | 0,044 | 0,965 |
| | 0,371 | 0,152 | 0,397 | 0,047 | 0,967 |
| | 0,362 | 0,153 | 0,389 | 0,049 | 0,953 |
| | 0,378 | 0,152 | 0,406 | 0,077 | 1,013 |
| | 0,385 | 0,152 | 0,401 | 0,061 | 0,999 |
| | 0,359 | 0,153 | 0,390 | 0,043 | 0,945 |
| | 0,426 | 0,151 | 0,436 | 0,067 | 1,080 |
| **Média** | **0,379** | **0,152** | **0,403** | **0,056** | **0,989** |

Tabela 5.2: Consumo de energia no algoritmo AES – minuto 60.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| | 0,381 | 0,152 | 0,396 | 0,021 | 0,950 |
| | 0,360 | 0,153 | 0,390 | 0,038 | 0,941 |
| | 0,363 | 0,153 | 0,380 | 0,014 | 0,910 |
| | 0,365 | 0,152 | 0,380 | 0,013 | 0,910 |
| | 0,364 | 0,152 | 0,379 | 0,018 | 0,913 |
| | 0,354 | 0,153 | 0,378 | 0,015 | 0,900 |
| | 0,366 | 0,152 | 0,382 | 0,019 | 0,919 |
| | 0,376 | 0,152 | 0,381 | 0,015 | 0,924 |
| | 0,353 | 0,153 | 0,379 | 0,013 | 0,898 |
| | 0,411 | 0,151 | 0,389 | 0,017 | 0,968 |
| **Média** | **0,369** | **0,152** | **0,383** | **0,018** | **0,923** |

Após o minuto 60 já é possível observar um padrão no consumo de energia. Esse padrão se confirma no cálculo do desvio padrão. Quando analisados os dados dos sensores no minuto 10 tem-se um desvio padrão de 0,038 mW seguido de 0,022 mW no minuto 60 e 0,020 mW no minuto 120.

A Figura 5.1 mostra a tendência do consumo. Essa figura mostra o gráfico do consumo de energia total (CPU + LPM + RX + TX) em mW durante os 120 minutos (aproximadamente 2 horas) de simulação.

Tabela 5.3: Consumo de energia no algoritmo AES – minuto 120.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| | 0,381 | 0,152 | 0,395 | 0,017 | 0,945 |
| | 0,359 | 0,153 | 0,386 | 0,030 | 0,928 |
| | 0,363 | 0,153 | 0,378 | 0,010 | 0,904 |
| | 0,364 | 0,152 | 0,379 | 0,012 | 0,907 |
| | 0,364 | 0,152 | 0,379 | 0,013 | 0,907 |
| | 0,354 | 0,153 | 0,377 | 0,013 | 0,897 |
| | 0,365 | 0,152 | 0,380 | 0,013 | 0,910 |
| | 0,377 | 0,152 | 0,381 | 0,012 | 0,922 |
| | 0,353 | 0,153 | 0,378 | 0,010 | 0,894 |
| | 0,410 | 0,151 | 0,385 | 0,013 | 0,959 |
| **Média** | **0,369** | **0,152** | **0,382** | **0,014** | **0,917** |

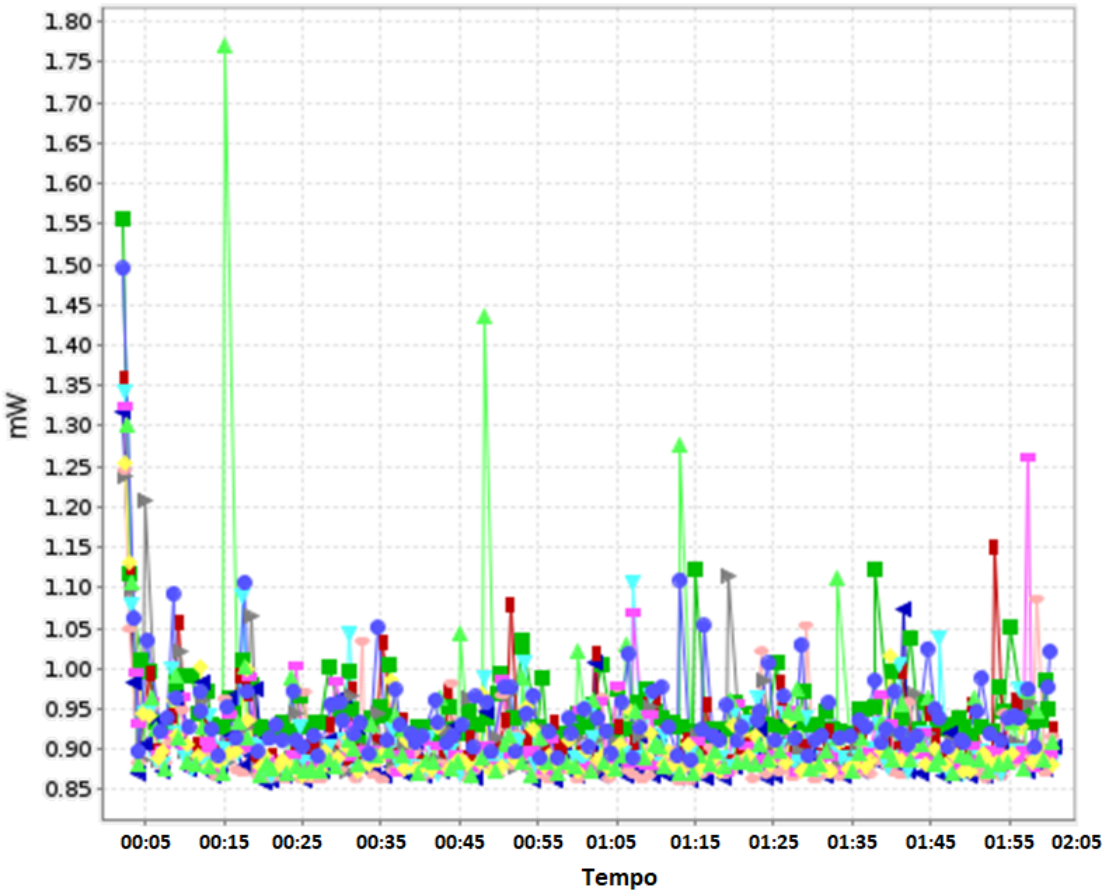



Figura 5.1: Energia consumida na execução do algoritmo AES.

## 5.2 Present

Simulando o algoritmo Present em um cenário com 10 sensores (*motes*) mais um *mote sink*, responsável pela coleta das informações dos sensores, foram observados os dados da Tabela 5.4 no minuto 10, da Tabela 5.5 no minuto 60 e da Tabela 5.6 no minuto 120.

Tabela 5.4: Consumo de energia no algoritmo Present – minuto 10.

| | **CPU** | **LPM** | **RX** | **TX** | **Total (mW)** |
|---|---|---|---|---|---|
| | 0,377 | 0,152 | 0,400 | 0,060 | 0,989 |
| | 0,375 | 0,152 | 0,395 | 0,046 | 0,968 |
| | 0,375 | 0,152 | 0,405 | 0,066 | 0,998 |
| | 0,349 | 0,153 | 0,386 | 0,045 | 0,933 |
| | 0,373 | 0,152 | 0,395 | 0,057 | 0,977 |
| | 0,385 | 0,152 | 0,407 | 0,057 | 1,001 |
| | 0,368 | 0,152 | 0,396 | 0,069 | 0,985 |
| | 0,413 | 0,151 | 0,413 | 0,050 | 1,027 |
| | 0,362 | 0,153 | 0,396 | 0,060 | 0,971 |
| | 0,350 | 0,153 | 0,393 | 0,067 | 0,963 |
| **Média** | **0,373** | **0,152** | **0,399** | **0,058** | **0,981** |

Tabela 5.5: Consumo de energia no algoritmo Present – minuto 60

| | **CPU** | **LPM** | **RX** | **TX** | **Total (mW)** |
|---|---|---|---|---|---|
| | 0,366 | 0,152 | 0,381 | 0,017 | 0,916 |
| | 0,368 | 0,152 | 0,383 | 0,016 | 0,919 |
| | 0,363 | 0,153 | 0,379 | 0,016 | 0,911 |
| | 0,342 | 0,153 | 0,377 | 0,014 | 0,886 |
| | 0,365 | 0,152 | 0,379 | 0,016 | 0,912 |
| | 0,375 | 0,152 | 0,384 | 0,015 | 0,926 |
| | 0,356 | 0,153 | 0,380 | 0,018 | 0,907 |
| | 0,398 | 0,151 | 0,383 | 0,014 | 0,946 |
| | 0,355 | 0,153 | 0,380 | 0,015 | 0,903 |
| | 0,343 | 0,153 | 0,378 | 0,015 | 0,889 |
| **Média** | **0,363** | **0,152** | **0,380** | **0,016** | **0,912** |

Após o minuto 60 já é possível observar um padrão no consumo de energia. Quando analisados os dados dos sensores no minuto 10 tem-se um desvio padrão de 0,024 mW seguido de 0,017 mW nos minutos 60 e 120.

Tabela 5.6: Consumo de energia no algoritmo Present – minuto 120.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| | 0,365 | 0,152 | 0,378 | 0,014 | 0,909 |
| | 0,367 | 0,152 | 0,380 | 0,013 | 0,912 |
| | 0,363 | 0,153 | 0,377 | 0,011 | 0,904 |
| | 0,342 | 0,153 | 0,377 | 0,012 | 0,884 |
| | 0,364 | 0,152 | 0,377 | 0,012 | 0,905 |
| | 0,375 | 0,152 | 0,380 | 0,012 | 0,919 |
| | 0,354 | 0,153 | 0,377 | 0,013 | 0,897 |
| | 0,399 | 0,151 | 0,383 | 0,011 | 0,944 |
| | 0,355 | 0,153 | 0,378 | 0,012 | 0,898 |
| | 0,342 | 0,153 | 0,377 | 0,011 | 0,883 |
| **Média** | **0,363** | **0,152** | **0,378** | **0,012** | **0,906** |

A tendência do consumo também é apresentada na Figura 5.2, que mostra o gráfico do consumo de energia total (CPU + LPM + RX + TX) em mW durante os 120 minutos de simulação.

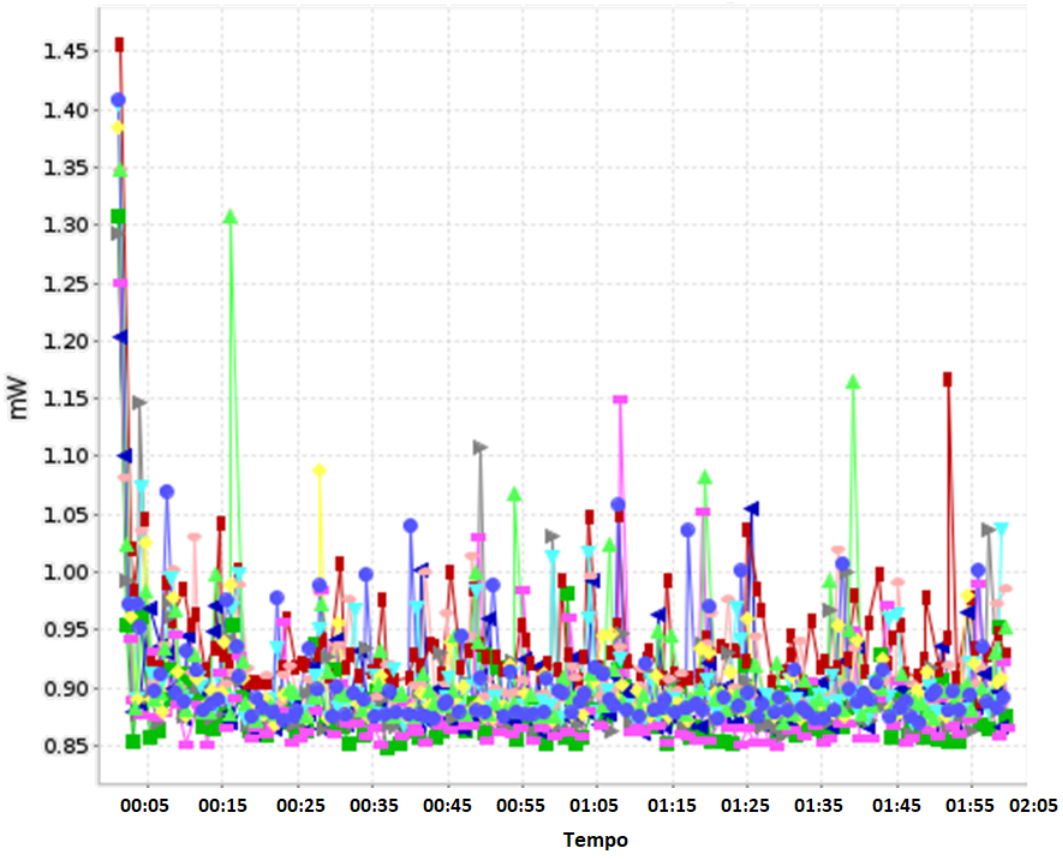



Figura 5.2: Energia consumida na execução do algoritmo Present.

## 5.3 Simon

Com o algoritmo Simon, também, foi simulado um cenário com 10 sensores (*motes*) mais um *mote sink*, responsável pela coleta das informações dos sensores. Neste cenário, foram observados os dados da Tabela 5.7 no minuto 10, da Tabela 5.8 no minuto 60 e da Tabela 5.9 no minuto 120.

Tabela 5.7: Consumo de energia no algoritmo Simon – minuto 10.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| | 0,364 | 0,152 | 0,392 | 0,058 | 0,966 |
| | 0,400 | 0,151 | 0,412 | 0,053 | 1,016 |
| | 0,373 | 0,152 | 0,399 | 0,050 | 0,974 |
| | 0,377 | 0,152 | 0,409 | 0,072 | 1,010 |
| | 0,400 | 0,151 | 0,409 | 0,056 | 1,016 |
| | 0,402 | 0,151 | 0,418 | 0,069 | 1,040 |
| | 0,376 | 0,152 | 0,404 | 0,072 | 1,004 |
| | 0,403 | 0,151 | 0,404 | 0,070 | 1,028 |
| | 0,376 | 0,152 | 0,397 | 0,050 | 0,975 |
| | 0,387 | 0,152 | 0,419 | 0,067 | 1,025 |
| **Média** | **0,386** | **0,152** | **0,406** | **0,062** | **1,005** |

Tabela 5.8: Consumo de energia no algoritmo Simon – minuto 60.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| | 0,355 | 0,153 | 0,382 | 0,017 | 0,907 |
| | 0,389 | 0,152 | 0,384 | 0,016 | 0,941 |
| | 0,364 | 0,152 | 0,380 | 0,014 | 0,910 |
| | 0,365 | 0,152 | 0,383 | 0,017 | 0,917 |
| | 0,388 | 0,152 | 0,382 | 0,018 | 0,940 |
| | 0,387 | 0,152 | 0,383 | 0,017 | 0,939 |
| | 0,367 | 0,152 | 0,385 | 0,018 | 0,922 |
| | 0,388 | 0,152 | 0,383 | 0,019 | 0,942 |
| | 0,365 | 0,152 | 0,381 | 0,016 | 0,914 |
| | 0,377 | 0,152 | 0,386 | 0,017 | 0,932 |
| **Média** | **0,375** | **0,152** | **0,383** | **0,017** | **0,926** |

Após o minuto 60 já é possível observar um padrão no consumo de energia. Quando analisados os dados dos sensores no minuto 10 tem-se um desvio padrão de 0,024 mW

seguido de 0,013 mW nos minutos 60 e 120. A Figura 5.3 apresenta a tendência do consumo, com a energia total (CPU + LPM + RX + TX) em mW nos 120 minutos de simulação.

Tabela 5.9: Consumo de energia no algoritmo Simon – minuto 120.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| | 0,354 | 0,153 | 0,379 | 0,012 | 0,898 |
| | 0,389 | 0,152 | 0,382 | 0,012 | 0,935 |
| | 0,364 | 0,152 | 0,378 | 0,013 | 0,907 |
| | 0,364 | 0,152 | 0,381 | 0,014 | 0,911 |
| | 0,387 | 0,152 | 0,380 | 0,013 | 0,932 |
| | 0,387 | 0,152 | 0,381 | 0,014 | 0,934 |
| | 0,366 | 0,152 | 0,382 | 0,014 | 0,914 |
| | 0,387 | 0,152 | 0,382 | 0,014 | 0,935 |
| | 0,365 | 0,152 | 0,379 | 0,012 | 0,908 |
| | 0,376 | 0,152 | 0,382 | 0,013 | 0,923 |
| **Média** | **0,374** | **0,152** | **0,381** | **0,013** | **0,920** |

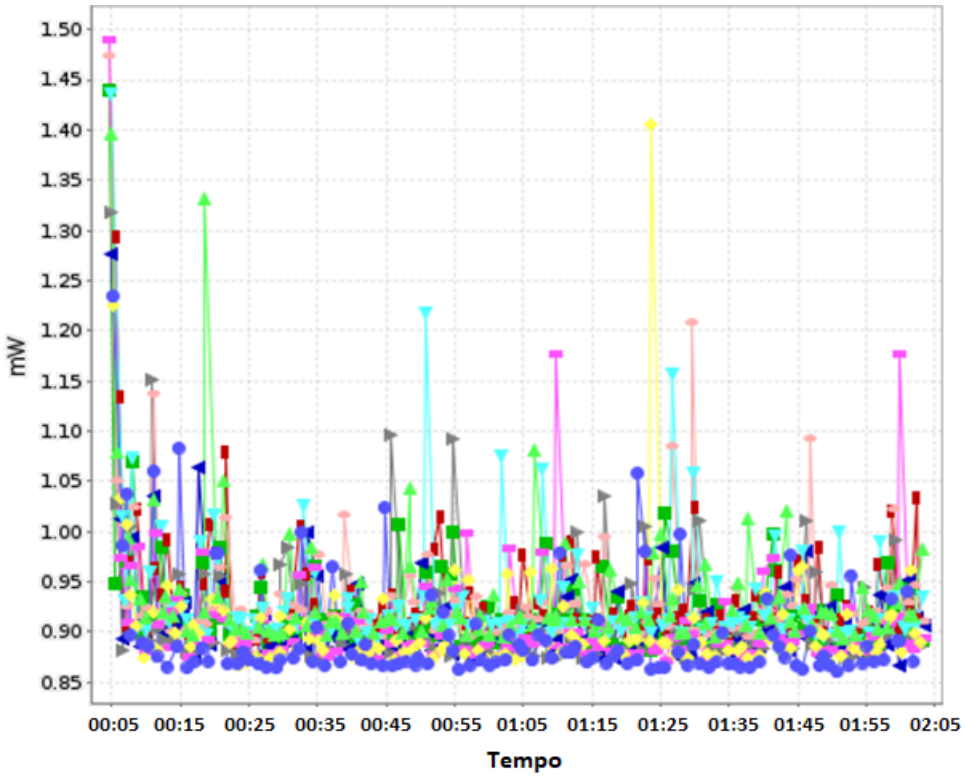



Figura 5.3: Energia consumida na execução do algoritmo Simon.

## 5.4 Speck

Repetindo o cenário de 10 sensores (*motes*), agora com o algoritmo Speck, mais um *mote sink*, responsável pela coleta das informações dos sensores, foram observados os dados da Tabela 5.10 no minuto 10, da Tabela 5.11 no minuto 60 e da Tabela 5.12 no minuto 120.

Tabela 5.10: Consumo de energia no algoritmo Speck – minuto 10.

| | **CPU** | **LPM** | **RX** | **TX** | **Total (mW)** |
|---|---|---|---|---|---|
| | 0,362 | 0,153 | 0,389 | 0,048 | 0,952 |
| | 0,402 | 0,151 | 0,408 | 0,049 | 1,010 |
| | 0,362 | 0,153 | 0,391 | 0,053 | 0,959 |
| | 0,386 | 0,152 | 0,401 | 0,045 | 0,984 |
| | 0,384 | 0,152 | 0,398 | 0,052 | 0,986 |
| | 0,373 | 0,152 | 0,400 | 0,055 | 0,980 |
| | 0,383 | 0,152 | 0,406 | 0,074 | 1,015 |
| | 0,352 | 0,153 | 0,390 | 0,062 | 0,957 |
| | 0,350 | 0,153 | 0,390 | 0,060 | 0,953 |
| | 0,374 | 0,152 | 0,405 | 0,064 | 0,995 |
| **Média** | **0,373** | **0,152** | **0,398** | **0,056** | **0,979** |

Tabela 5.11: Consumo de energia no algoritmo Speck – minuto 60

| | **CPU** | **LPM** | **RX** | **TX** | **Total (mW)** |
|---|---|---|---|---|---|
| | 0,355 | 0,153 | 0,380 | 0,014 | 0,902 |
| | 0,388 | 0,152 | 0,384 | 0,015 | 0,939 |
| | 0,353 | 0,153 | 0,379 | 0,017 | 0,902 |
| | 0,375 | 0,152 | 0,380 | 0,012 | 0,919 |
| | 0,376 | 0,152 | 0,380 | 0,014 | 0,922 |
| | 0,364 | 0,153 | 0,380 | 0,015 | 0,912 |
| | 0,366 | 0,152 | 0,384 | 0,018 | 0,920 |
| | 0,343 | 0,153 | 0,377 | 0,015 | 0,888 |
| | 0,343 | 0,153 | 0,379 | 0,017 | 0,892 |
| | 0,366 | 0,152 | 0,382 | 0,019 | 0,919 |
| **Média** | **0,364** | **0,152** | **0,381** | **0,016** | **0,913** |

O padrão é o mesmo dos algoritmos anteriores, após o minuto 60 a tendência de estabilidade dos sensores é provada na análise do desvio padrão. No minuto 10 tem-se um desvio padrão de 0,022 mW seguido de 0,015 mW no minuto 60 e 0,014 mW no minuto 120. A Figura 5.4

apresenta a tendência do consumo, com a energia total (CPU + LPM + RX + TX) em mW nos 120 minutos de simulação.

Tabela 5.12: Métricas de consumo de energia no algoritmo Speck.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| | 0,354 | 0,153 | 0,378 | 0,011 | 0,896 |
| | 0,388 | 0,152 | 0,382 | 0,014 | 0,936 |
| | 0,353 | 0,153 | 0,377 | 0,013 | 0,896 |
| | 0,374 | 0,152 | 0,380 | 0,010 | 0,916 |
| | 0,375 | 0,152 | 0,378 | 0,011 | 0,916 |
| | 0,364 | 0,152 | 0,378 | 0,011 | 0,905 |
| | 0,365 | 0,152 | 0,380 | 0,014 | 0,911 |
| | 0,344 | 0,153 | 0,378 | 0,012 | 0,887 |
| | 0,344 | 0,153 | 0,378 | 0,013 | 0,888 |
| | 0,365 | 0,152 | 0,379 | 0,014 | 0,910 |
| **Média** | **0,363** | **0,152** | **0,379** | **0,012** | **0,906** |

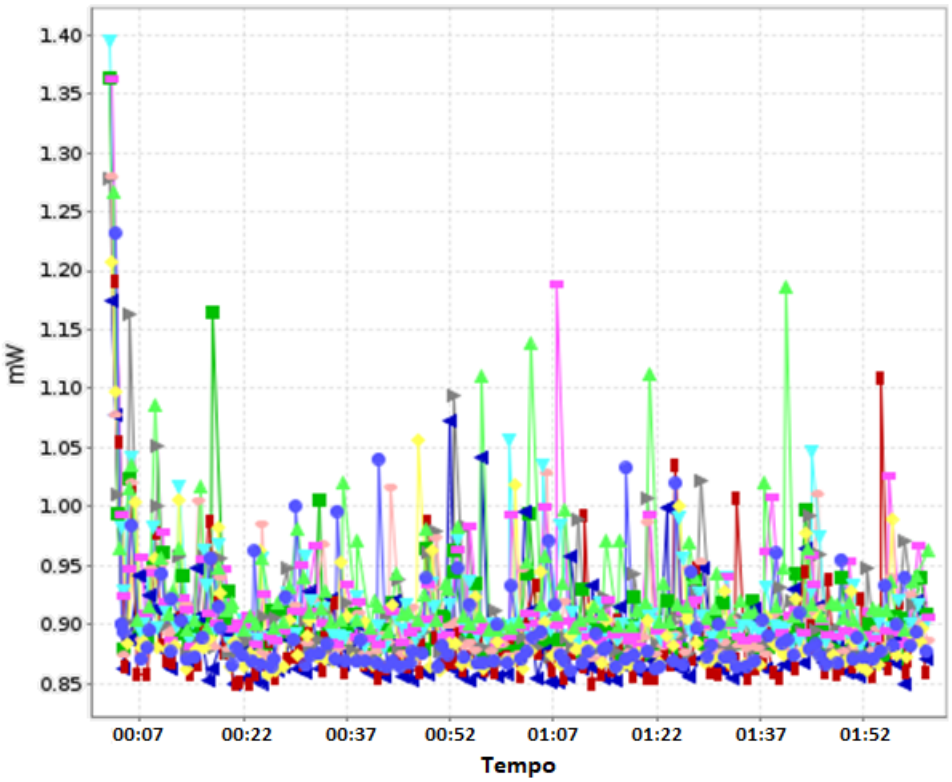



Figura 5.4: Energia consumida na execução do algoritmo Speck.

## 5.5 Considerações gerais

Analisando as Tabelas 5.13, 5.14, 5.15 e 5.16 que apresentam as médias obtidas nas simulações com dez sensores em 10 minutos, 60 minutos e 120 minutos, detectamos que as simulações com 10 minutos são muito instáveis. Essa instabilidade se dá pelo esforço em definir com o que o dispositivo vai se comunicar de forma segura, além do processo de inicialização dos sensores que tende a demorar.

Tabela 5.13: Métricas de consumo de energia no algoritmo AES.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| Média 10 | 0,379 | 0,152 | 0,403 | 0,056 | 0,989 |
| Média 60 | 0,369 | 0,152 | 0,383 | 0,018 | 0,923 |
| Média 120 | 0,369 | 0,152 | 0,382 | 0,014 | 0,917 |

Tabela 5.14: Métricas de consumo de energia no algoritmo Present.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| Média 10 | 0,373 | 0,152 | 0,399 | 0,058 | 0,981 |
| Média 60 | 0,363 | 0,152 | 0,380 | 0,016 | 0,912 |
| Média 120 | 0,363 | 0,152 | 0,378 | 0,012 | 0,906 |

Tabela 5.15: Métricas de consumo de energia no algoritmo Simon.

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| Média 10 | 0,386 | 0,152 | 0,406 | 0,062 | 1,005 |
| Média 60 | 0,375 | 0,152 | 0,383 | 0,017 | 0,926 |
| Média 120 | 0,374 | 0,152 | 0,381 | 0,013 | 0,920 |

Tabela 5.16: Métricas de consumo de energia no algoritmo Speck

| | CPU | LPM | RX | TX | Total (mW) |
|---|---|---|---|---|---|
| Média 10 | 0,373 | 0,152 | 0,398 | 0,056 | 0,979 |
| Média 60 | 0,364 | 0,152 | 0,381 | 0,016 | 0,913 |
| Média 120 | 0,363 | 0,152 | 0,379 | 0,012 | 0,906 |

O desvio padrão apresentado na simulação do AES com 10 minutos chega a 0,038 mW, sendo o maior entre os algoritmos simulados. O cálculo do desvio padrão confirma que, a partir de 60 minutos de simulação, os resultados obtidos são mais estáveis.

A maior instabilidade está no nível TX, que representa a energia gasta na transmissão de dados. Esse parâmetro apresenta uma queda maior, pois com o passar do tempo de simulação os sensores já estabeleceram um nível de confiança entre si, diminuindo o consumo de energia nesta troca de informações.

Considerando o nível de CPU, no minuto com o menor desvio padrão em todos os algoritmos (minuto 120), o algoritmo Simon consumiu 0,011 mW a mais que os algoritmos Present e Speck, os mais econômicos.

No total, os sensores apresentaram um aumento no consumo de energia quando implementado o algoritmo Simon. O algoritmo Simon consome 0,003 mW a mais que o AES e 0,014 mW a mais que os algoritmos Present e Speck.

Percebemos nos gráficos que, nos primeiros minutos, existe uma oscilação maior no consumo. Isso é o esperado, pois os sensores tendem a demorar para inicializar e estabelecer um nível de confiança com o servidor. A partir dos 30 minutos, mesmo com muitos *outliers* em alguns casos, é possível notar que ha uma tendência mais estável no consumo.

Assim como Tausif et al. (2017) cita o cuidado ao implementar algoritmos criptográficos, neste trabalho nos preocupamos em demandar o mínimo possível dos recursos dos dispositivos simulados (*motes*). Dessa forma, utilizamos algoritmos de criptografia leves e mantemos as chaves fixas, facilitando a programação, principalmente no simulador escolhido.

As conclusões deste trabalho se somam as de El-Haii et al. (2018) que evidenciou que problema do uso de algoritmos criptográficos em dispositivos IoT não está nos custos computacionais, visto que é viável a implementação. Divergindo de El-Haii et al. (2018) e Dhanda, Brahmjit Singh e Jindal (2020), que indicam o uso do AES, Beg, Al-Kharobi e Al-Nasser (2019) indicam o Present para ser utilizado em dispositivos IoT.

Neste estudo, o Present também apresentou um consumo de energia inferior ao AES. No entanto, a diferença foi de 0,011 mW. Analisando os argumentos utilizados por El-Haii et al. (2018) e Dhanda, Brahmjit Singh e Jindal (2020), consideramos que esta diferença apresentada não é o suficiente para definir o Present como o mais indicado para dispositivos IoT.

# Capítulo 6

# Conclusões

O presente trabalho teve o intuito de contextualizar e buscar referências que justifiquem o estudo da implementação de algoritmos criptográficos em dispositivos para IoT. Assim, foi realizado um levantamento bibliográfico sobre os algoritmos para criptografia mais indicados para a utilização nos dispositivos para IoT. Como resultado dessa análise bibliográfica, observou-se que o assunto é bastante discutido atualmente e que os algoritmos AES, Present, Simon e Speck são os mais visados nos estudos.

A razão para esse interesse é o aumento da presença dos dispositivos para a IoT no cotidiano das pessoas e indústrias. Com isso, esses dispositivos podem atuar como um ponto vulnerável em que, caso não assegurem um nível suficiente de segurança, podem permitir vazamento de dados ou outros ataques à segurança com possibilidade de danos maiores.

Assim, realizamos vários experimentos no simulador Cooja. A ideia foi fornecer argumentos quantitativos para a escolha da melhor cifra para embarcar em dispositivos para IoT. Fizemos a análise para a escolha da melhor cifra no que se refere ao consumo de energia, isto é, aquela que menos consome energia ao mesmo tempo em que fornece um mecanismo robusto para implementar a confidencialidade dos dados.

O simulador Cooja foi uma ferramenta interessante para a redução dos custos dos experimentos e, principalmente, para a criação de cenários diversos, inclusive cenários com limitações de recursos nos dispositivos. No entanto, para futura replicabilidade desse trabalho, é preciso estar atento ao tempo de execução e a quantidade de sensores utilizados.

O aumento do número de sensores prejudica o desempenho do simulador, pois quanto maior a quantidade de sensores, maior será o esforço do simulador. Mais sensores implicam

em mais informações sendo coletadas, armazenadas e analisadas para a confecção dos gráficos, provocando após um determinado período um *overflow* de memória travando a simulação.

Apesar do ambiente restrito do Cooja, os algoritmos selecionados demonstraram viabilidade de implementação. Este estudo mostrou ainda, que a diferença no quesito consumo entre os algoritmos é na casa dos miliwatts. Portanto, concluímos que o consumo de energia não é um critério de extrema significância para a escolha sobre qual destes algoritmos testados deve ser embarcado na maioria dos dispositivos IoT.

Para efeito de escolha de algoritmos é mais sensato optar por aqueles monitorados pelo NIST, como o AES. Tais algoritmos apresentam uma segurança comprovada e passam por rigorosos testes de validação.

Por isso, estamos confiantes que a continuidade desse trabalho na análise e experimentação dos algoritmos poderá fornecer subsídios para uma decisão que permita aliar uma qualidade satisfatória de segurança a um desempenho compatível com as limitações dos dispositivos para IoT no que se refere ao seu desempenho computacional.

# Referências bibliográficas

AHTSHAM, M.; YAN, H. Y.; ALI, U. IoT based door lock surveillance system using cryptographic algorithms. In: 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC). Banff: IEEE, 2019. P. 448–453. DOI: 10.1109/ICNSC.2019.8743330.

BARBOSA, M.; FARSHIM, P. The Related-Key Analysis of Feistel Constructions. In: CID, C.; RECHBERGER, C. (Ed.). **International Workshop on Fast Software Encryption**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. P. 265–284. DOI: 10.1007/978-3-662-46706-0_14.

BATTEN, L. M. **Public key cryptography: applications and attacks**. Hoboken: John Wiley & Sons, 2013. v. 16. ISBN 978-1118317129.

BEAULIEU, R. et al. The SIMON and SPECK families of lightweight block ciphers. **cryptology eprint archive**, 2013.

BEG, A.; AL-KHAROBI, T.; AL-NASSER, A. Performance Evaluation and Review of Lightweight Cryptography in an Internet-of-Things Environment. In: 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS). Riyadh: IEEE, 2019. P. 1–6. DOI: 10.1109/CAIS.2019.8769509.

BOGDANOV, A. et al. PRESENT: An ultra-lightweight block cipher. In: PAILLIER, P.; VERBAUWHEDE, I. (Ed.). **Cryptographic Hardware and Embedded Systems - CHES 2007**. Berlin, Heidelberg: Springer, 2007. P. 450–466. DOI: 10.1007/978-3-540-74735-2_31.

BURTON, D. M. **Teoria elementar dos números**. Rio de Janeiro: Grupo Gen, 2016. ISBN 978-8521631019.

BÜSCHING, F.; WOLF, L. The rebirth of one-time pads—Secure data transmission from ban to sink. **IEEE Internet of Things Journal**, IEEE, v. 2, n. 1, p. 63–71, 2015. DOI: 10.1109/JIOT.2014.2378783.

CERT.BR. **Incidentes Reportados ao CERT.br – Janeiro a Dezembro de 2020**. Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. 2022. Disponível em: <https://www.cert.br/stats/incidentes/2020-jan-dec/top-cc.html>. Acesso em: 24 mar. 2022.

CERT.BR. **Total de Incidentes Reportados ao CERT.br por Ano**. Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. 2022. Disponível em: <https://www.cert.br/stats/incidentes/>. Acesso em: 24 mar. 2022.

CONTIKI. **Contiki: The Open Source OS for the Internet of Things**. 2023. Disponível em: <https://github.com/contiki-os/contiki>. Acesso em: 15 mar. 2023.

DAI, W. **Crypto++ Library – Free C++ Class Library of Cryptographic Schemes**. Versão 8.6. Jan. 2021. Disponível em: <https://github.com/weidai11/cryptopp>. Acesso em: 5 abr. 2022.

DE PAULA, G. C. S. G. **Modelo de Internet das Coisas para o Parque Estadual do Cantão**. Nov. 2015. F. 66. Trabalho de Conclusão de Curso – Centro Universitário Luterano de Palmas. DOI: 10.13140/RG.2.2.18640.30725.

DHANDA, S. S.; SINGH, B.; JINDAL, P. Lightweight cryptography: A solution to secure IoT. **Wireless Personal Communications**, Springer, v. 112, n. 3, p. 1947–1980, 2020. DOI: 10.1007/s11277-020-07134-3.

DIFFIE, W.; HELLMAN, M. E. Multiuser cryptographic techniques. In: PROCEEDINGS of the June 7-10, 1976, national computer conference and exposition. [S.l.]: ACM Press, 1976. P. 109–112. DOI: 10.1145/1499799.1499815.

DUNKELS, A. **AVR Motes in Cooja**. 2018. Disponível em: <https://github.com/contiki-os/contiki/wiki/AVR-Motes-in-Cooja>. Acesso em: 10 jan. 2023.

DUNKELS, A. **Contiki 2.6: The ESB Embedded Sensor Board**. 2012. Disponível em: <https://contiki.sourceforge.net/docs/2.6/a01781.html>. Acesso em: 10 jan. 2023.

DUNKELS, A. **Contiki 2.6: The Tmote Sky Board**. 2012. Disponível em: <https://contiki.sourceforge.net/docs/2.6/a01784.html>. Acesso em: 10 jan. 2023.

EISENBARTH, T. et al. A survey of lightweight-cryptography implementations. **IEEE Design & Test of Computers**, IEEE, v. 24, n. 6, p. 522–533, nov. 2007. DOI: 10.1109/mdt.2007.178.

EVANS, D. **The Internet of Things: How the Next Evolution of the Internet Is Changing Everything**. Cisco Internet Business Solutions Group. Abr. 2011. Disponível em: <http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf>. Acesso em: 23 mar. 2022.

EL-HAII, M.; CHAMOUN, M.; FADLALLAH, A.; SERHROUCHNI, A. Analysis of cryptographic algorithms on iot hardware platforms. In: 2018 2nd Cyber Security in Networking Conference (CSNet). Paris: IEEE, 2018. P. 1–5. DOI: 10.1109/CSNET.2018.8602942.

HEARTFIELD, R.; LOUKAS, G. A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. **ACM Computing Surveys (CSUR)**, ACM New York, NY, EUA, v. 48, n. 3, p. 1–39, 2015. DOI: 10.1145/2835375.

KANE, L. E. et al. Security and performance in IoT: a balancing act. **IEEE Access**, IEEE, v. 8, p. 121969–121986, 2020. DOI: 10.1109/ACCESS.2020.3007536.

KATZ, J.; LINDELL, Y. **Introduction to modern cryptography**. Boca Raton: CRC press, 2020. ISBN 978-1466570269.

KERCKHOFFS, A. La cryptographie militaire. **Journal des sciences militaires**, Gallica, v. IX, p. 5–38, 161–191, jan. 1883.

KHALIFA, O. O.; ISLAM, M. R.; KHAN, S.; SHEBANI, M. S. Communications cryptography. In: 2004 RF and Microwave Conference (IEEE Cat. No. 04EX924). Selangor, Malaysia: IEEE, 2004. P. 220–223. DOI: 10.1109/RFM.2004.1411111.

KHAN, M. N.; RAO, A.; CAMTEPE, S. Lightweight Cryptographic Protocols for IoT-Constrained Devices: A Survey. **IEEE Internet of Things Journal**, IEEE, v. 8, n. 6, p. 4132–4156, 2020. DOI: 10.1109/JIOT.2020.3026493.

KHAN, N. et al. Performance analysis of security algorithms for IoT devices. In: 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). Dhaka: IEEE, 2017. P. 130–133. DOI: 10.1109/R10-HTC.2017.8288923.

KIM, D.; SOLOMON, M. G. **Fundamentos de segurança de sistemas de informação**. Rio de Janeiro: LTC, 2014. ISBN 978-8521625070.

KNUTH, D. E. **The art of computer programming**. 2. ed. Edinburgh: Addison-Wesley, 1998. v. III. ISBN 0201896850.

LASSO, F. F. J.; CLARKE, K.; NIRMALATHAS, A. A software-defined networking framework for IoT based on 6LoWPAN. In: 2018 Wireless Telecommunications Symposium (WTS). Phoenix, EUA: IEEE, 2018. P. 1–7. DOI: 10.1109/WTS.2018.8363938.

LOPES JUNIOR, V.; MIRANDA, G. A.; GONÇALVES, K. C.; SILVA, C. A. Analysis and Comparison of Cryptographic Algorithms applied to IoT. In: 2019 14th Iberian Conference on Information Systems and Technologies (CISTI). Coimbra: IEEE, 2019. P. 1–6. DOI: 10.23919/CISTI.2019.8760998.

MACHADO, F. N. R. **Segurança da informação: princípios e controle de ameaças**. São José dos Campos: Érica, 2014. ISBN 978-8536507842.

MACHADO, T. et al. Methodology for identifying the cybersecurity maturity level of smart grids. **IEEE Latin America Transactions**, IEEE, v. 14, n. 11, p. 4512–4519, 2016. DOI: 10.1109/TLA.2016.7795822.

MAO, W. **Modern cryptography: theory and practice**. India: Prentice Hall, 2003. ISBN 978-0132887410.

MARAM, B.; GNANASEKAR, J.; MANOGARAN, G.; BALAANAND, M. Intelligent security algorithm for UNICODE data privacy and security in IOT. **Service Oriented Computing and Applications**, Springer, v. 13, n. 1, p. 3–15, 2019. DOI: 10.1007/s11761-018-0249-x.

MASCHIETTO, L. G. et al. **Arquitetura e Infraestrutura de IoT**. Porto Alegre: Grupo A, 2021.

MILANOV, E. The RSA algorithm. **RSA laboratories**, p. 1–11, 2009.

MORAES, A. D.; HAYASHI, V. T. **Segurança em IoT: entendendo os riscos e ameaças em IoT**. Rio de Janeiro: Alta Books, 2021. ISBN 978-8550816517.

MORAES, A. F. **Segurança em redes: fundamentos**. São Paulo: Érica, 2010. ISBN 978-8536522081.

MORKEL, T.; ELOFF, J. Encryption techniques: a timeline approach. **Information and Computer Security Architecture (ICSA) Research Group, University of Pretoria**, v. 2, 2004.

MUELLER, J. P.; MASSARON, L. **Algoritmos para leigos**. Rio de Janeiro: Alta Books, 2018. ISBN 9788550809298.

MUTHAVHINE, K. D.; SUMBWANYAMBE, M. An analysis and a comparative study of cryptographic algorithms used on the Internet of Things (IoT) based on avalanche effect. In: 2018 International Conference on Information and Communications Technology (ICOIACT). Yogyakarta: IEEE, 2018. P. 114–119. DOI: 10.1109/ICOIACT.2018.8350759.

OSTERLIND, F. et al. Cross-level sensor network simulation with cooja. In: PROCEEDINGS. 2006 31st IEEE conference on local computer networks. Tampa, EUA: IEEE, 2006. P. 641–648. DOI: 10.1109/LCN.2006.322172.

PAAR, C.; PELZL, J. **Understanding cryptography: a textbook for students and practitioners**. [S.l.]: Springer Science & Business Media, 2009.

PALLAVI, K.; KUMAR, V. R.; SRIKRISHNA, S. Comparative study of various lightweight cryptographic algorithms for data security between IoT and cloud. In: 2020 5th International Conference on Communication and Electronics Systems (ICCES). Coimbatore: IEEE, 2020. P. 589–593. DOI: 10.1109/ICCES48766.2020.9137984.

PEDIREDLA, B. et al. A 6LoWPAN implementation for memory constrained and power efficient wireless sensor nodes. In: IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society. Vienna, Austria: IEEE, 2013. P. 4432–4437. DOI: 10.1109/IECON.2013.6699849.

PINDAR, Z. A. et al. A Lightweight Message Authentication Code for Virtual Work in Future Smart Cities. In: 2020 IEEE European Technology and Engineering Management Summit (E-TEMS). Dortmund: IEEE, 2020. P. 1–5. DOI: 10.1109/E-TEMS46250.2020.9111859.

PINHEIRO, P. P. **Segurança digital: proteção de dados nas empresas**. São Paulo: Grupo GEN, 2021. ISBN 978-8597026399.

ROSEN, K. H. **Matemática discreta e suas aplicações**. Porto Alegre: Grupo A Educação, 2009. ISBN 978-8563308399.

SCHEINERMAN, E. R. **Matemática Discreta: Uma Introdução - Tradução da 3ª ed. norte-americana.** São Paulo: Cengage Learning Editores, 2016. ISBN 978-8522125388.

SINGH, P.; ACHARYA, B.; CHAURASIYA, R. K. A comparative survey on lightweight block ciphers for resource constrained applications. **International Journal of High Performance Systems Architecture**, Inderscience Publishers (IEL), v. 8, n. 4, p. 250–270, 2019. DOI: 10.1504/IJHPSA.2019.104953.

SINGH, S.; SHARMA, P. K.; MOON, S. Y.; PARK, J. H. Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. **Journal of Ambient**

**Intelligence and Humanized Computing**, Springer, p. 1–18, 2017. DOI: 10.1007/s12652-017-0494-4.

SINGH, S. **The Code Book**: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. New York: Anchor, 1999. v. 7. ISBN 978-0385495325.

SQUARTINI, S. et al. Wireless M-Bus sensor nodes in smart water grids: The energy issue. In: 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP). Beijing, China: IEEE, 2013. P. 614–619. DOI: 10.1109/ICICIP.2013.6568148.

STALLINGS, W. **Criptografia e Segurança de Redes: Princípios e Práticas**. 6. ed. São Paulo: Person Education, 2015. ISBN 978-8543005898.

TAUSIF, M.; FERZUND, J.; JABBAR, S.; SHAHZADI, R. Towards designing efficient lightweight ciphers for internet of things. **KSII Transactions on Internet and Information Systems (TIIS)**, Korean Society for Internet Information, v. 11, n. 8, p. 4006–4024, 2017. DOI: 10.3837/tiis.2017.08.014.

TERADA, R. **Segurança de dados: criptografia em rede de computador**. São Paulo: Editora Blucher, 2008. ISBN 978-8521204398.

THÉBAUDEAU, B. **Zolertia z1 motes**. 2019. Disponível em: <https://github.com/contiki-os/contiki/wiki/Zolertia-z1-motes>. Acesso em: 10 jan. 2023.

TOSCANI, L. V.; VELOSO, P. A. S. **Complexidade de algoritmos – V13 – UFRGS**. Porto Alegre: Bookman, 2012. ISBN 9788540701397.

WANG, Z. et al. Dynamically reconfigurable encryption and decryption system design for the internet of things information security. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 1, p. 143, 2019. DOI: 10.3390/s19010143.

WENGER, E.; UNTERLUGGAUER, T.; WERNER, M. **FLECC_IN_C: FLexible Elliptic Curve Cryptography library written in C**. Nov. 2017. Disponível em: <https://github.com/IAIK/flecc_in_c>. Acesso em: 5 abr. 2022.

WU, H.; DWIVEDI, A. D.; SRIVASTAVA, G. Security and Privacy of Patient Information in Medical Systems Based on Blockchain Technology. **ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)**, ACM New York, NY, v. 17, 2s, p. 1–17, 2021. DOI: 10.1145/3408321.

XU, M. et al. Toward engineering a secure android ecosystem: A survey of existing techniques. **ACM Computing Surveys (CSUR)**, ACM New York, EUA, v. 49, n. 2, p. 1–47, 2016. DOI: 10.1145/2963145.

XU, Y. et al. Research and design of reconfigurable security resource pool framework. In: PROCEEDINGS of the 2018 2nd International Conference on Computer Science and Artificial Intelligence. New York: Association for Computing Machinery, 2018. P. 620–626. DOI: 10.1145/3297156.3297241.

# Apêndice A

# Método PRISMA para revisão bibliográfica

Este apêndice descreve a metodologia utilizada na pesquisa das referências mais relevantes ao processo de revisão bibliográfica aplicado neste trabalho. Utilizou-se uma adaptação do *framework* PRISMA em sete etapas: 1) definição do tópico de pesquisa, 2) definição dos conceitos envolvidos, 3) seleção das fontes bibliográficas, 4) definição dos termos de busca, 5) definição da estratégia de pesquisa, 6) critérios de seleção e 7) fluxograma. Os sete passos citados são apresentados neste anexo.

## A.1 Definição do tópico de pesquisa

Nesta etapa especificou-se o tópico que orientou a definição dos conceitos envolvidos na pesquisa, sendo o tópico escolhido: "Viabilidade da implementação de algoritmos criptográficos em dispositivos para Internet das Coisas (IoT) ou Computação Vestível (Wearables)".

## A.2 Definição dos conceitos envolvidos

A partir do tópico escolhido, identificou-se três conceitos fundamentais: 1) *Information security*; 2) *Internet of Things*; e 3) *cryptographic algorithms.* A seguir encontra-se a descrição desses conceitos:

*Information security*: Segurança da informação refere-se à proteção contra o uso ou acesso não-autorizado de um conjunto de informações, no sentido de preservar a legitimidade que os dados possuem para um indivíduo ou uma organização.

*Internet of Things*: Refere-se a interconexão digital de objetos cotidianos com a internet, é uma rede de dispositivos físicos, como por exemplo veículos, relógios inteligentes e outros capaz de reunir e de transmitir dados. O termo surgiu em 1999 através de Kevin Ashton, no Laboratório de Auto-ID do MIT.

*Cryptographic algorithms*: Algoritmos criptográficos são usados para converter os dados legíveis em um texto cifrado, ou seja, um texto composto por valores ilegíveis que só podem ser decifrados com o código de acesso correto.

## A.3 Seleção das fontes bibliográficas

Realizou-se as pesquisas nas bases IEEE Xplore, ACM Digital Library, Scopus e Web of Science. Pesquisas em bibliotecas pessoais do autor e ainda, artigos indicados pelo orientador do trabalho utilizou-se em paralelo.

## A.4 Definição dos termos de busca

Através da análise das palavras-chave utilizadas em pesquisas relacionadas e por indicação do orientador, definiu-se os termos de busca. Sendo os termos utilizados: 1) *Information security*; 2) *Internet of Things*; e 3) *cryptographic algorithms*. No IEEE Xplore realizou-se a pesquisa com base no campo “Full Text and Metada”. No ACM Digital Library, baseou-se no “ACM Full-Text Collection” não houve restrições na pesquisa, sendo a busca final: [All: "information security"] AND [All: "internet of things"] AND [All: "cryptographic algorithms"]. Na Web of Science e na Scopus especificou-se somente o idioma, português e inglês.

## A.5 Definição da estratégia de pesquisa

A Tabela A.1 apresenta um exemplo de pesquisa realizada na Scopus. Observa-se que sem uma estratégia de busca com termos concatenas os resultados são, em valores, elevados, o que dificulta a análise ja que as chances de haver repostas desconexas com o objetivo do trabalho aumentam.

Tabela A.1: Exemplo de estratégia de pesquisa para Sopus e número de arquivos encontrados.

| |
|---|
| **1. Information security (29.378)**<br>TITLE-ABS-KEY (“Information security”) |
| **2. Internet of Things (152.118)**<br>TITLE-ABS-KEY (“Internet of things”) |
| **3. cryptographic algorithms (5.448)**<br>TITLE-ABS-KEY ("cryptographic algorithms") |
| **4. Pesquisa Final (30)**<br>((1.) AND (2.) AND (3.))<br>(TITLE-ABS-KEY ("Information security") AND TITLE-ABS-KEY ("Internet of Things")<br>AND TITLE-ABS-KEY ("cryptographic algorithms")) |

O gráfico da Figura A.1 mostra o número de publicações por ano da pesquisa final nas diferentes bases de dados. Identifica-se através do gráfico que o número de publicações aumentou nos últimos quatro anos e as bases que mais se destacam no volume de publicações é a Scopus em 2019 e a ACM nos anos de 2020 e 2017.

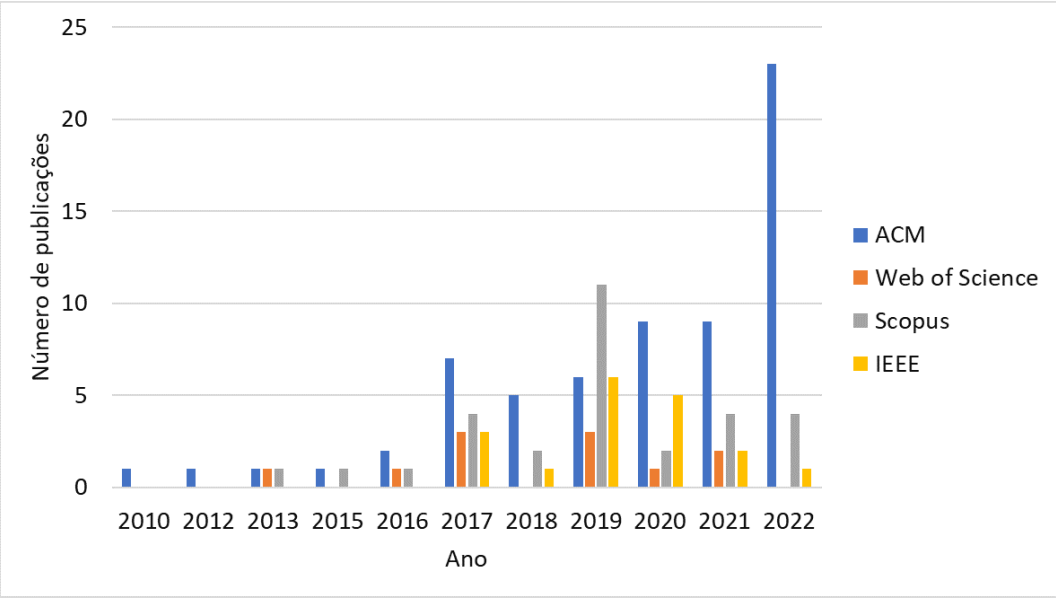



Figura A.1: Número de publicações encontradas por ano nas bases de dados.

## A.6 Critério de Seleção

Aplicou-se critérios de elegibilidade nos resultados da pesquisa, cada artigos foi submetido a uma triagem e excluídos os trabalhos que não atendiam as demandas. Critérios de seleção utilizados na análise do título inicial e resumo foram:

- Aborda algoritmos criptográficos leves e atuais;
- Está preocupado com a análise de segurança em dispositivos inteligentes;
- É uma pesquisa com um embasamento teórico bom.

Na leitura do texto completo utilizou-se os seguintes critérios:

- Análise da abordagem utilizada;
- Análise dos resultados;
- Análise das contribuições e conclusão.

Com os critérios especificados acima realizou-se um processo de triagem e análise detalhada dos artigos resultantes. Os resultados obtidos compõem a Tabela A.2 que apresenta ainda os pontos principais de cada artigo selecionado. Os detalhes do número de trabalhos selecionados em cada etapa da triagem são apresentados no tópico A.7 Fluxograma.

## A.7 Fluxograma

As primeiras pesquisas resultaram em 94 referências para consideração, das quais, como mostra o fluxograma da Figura A.2 reduziu-se para 19, uma vez que cumpriram os critérios de inclusão. Estes critérios cumprem a seguinte ordem de prioriades: 1) duplicadas, quantas referências se repetem nas outras bases de dados consideradas, 2) título, tem alguma das palavras-chaves definidas ou sinônimos, 3) resumo, aborda os critérios definidos no item A.6, 4) disponibilidade de acesso e 5) texto, obedece os critérios definidos no item A.6.

## A.8 Resultados

Na Tabela A.2 apresentam-se as características e um resumo das principais ideias das 19 referências aprovadas para compor o trabalho. Algumas limitações desta revisão fazem com

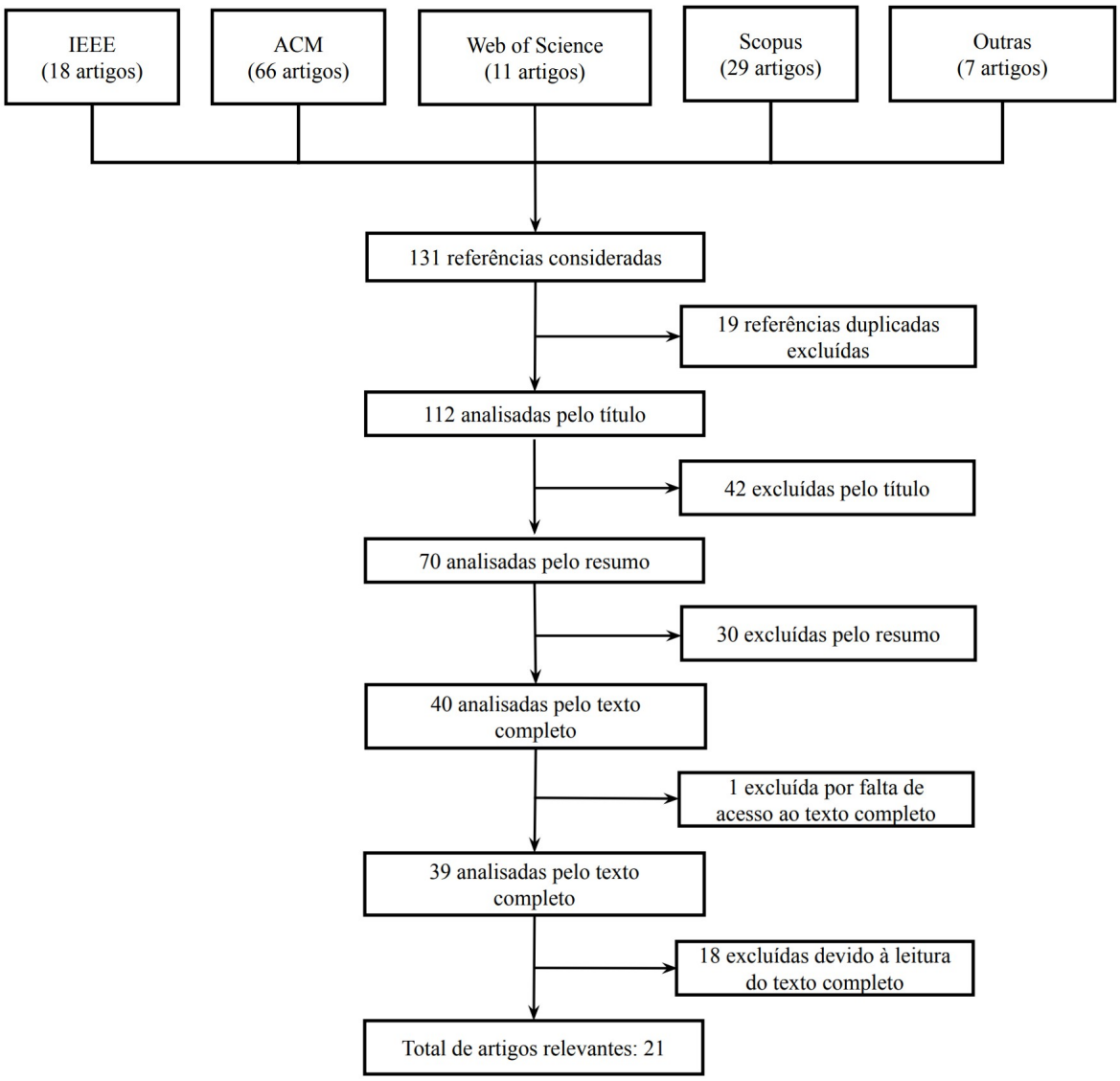



Figura A.2: Revisão sistemática

que referências relevantes passem despercebidas e por consequência sejam descartadas por não cumprimento dos critérios de elegibilidade. Todavia, pode-se afirmar que a pesquisa apresenta os principais avanços e desafios relacionados a viabilidade da implementação de algoritmos criptográficos em dispositivos para Internet das Coisas (IoT) ou Computação Vestível (Wearables).

Tabela A.2: Publicações e seus respectivos pontos-chave.

| Autor (ano) | Título | Pontos-Chave |
|---|---|---|
| (HEARTFIELD; LOUKAS, 2015) | A Taxonomy of Attacks and a Survey of Defence Mechanisms for Semantic Social Engineering Attacks | Apresenta uma taxonomia de ataques semânticos, e ainda um levantamento de defesas aplicáveis, compara o cenário de ameaças e as técnicas de mitigação associadas. |
| (BÜSCHING; WOLF, 2015) | The Rebirth of One-Time Pads—Secure Data Transmission from BAN to Sink | Utiliza pads de uso único (OTPs), comprovados matematicamente seguros e impossíveis de quebrar, para proteger a transmissão sem fio de dados pessoais. Propõe, implementa e avalia um sistema capaz de recarregar OTPs de nós WBAN. |
| | | Continua na próxima página. |

**Tabela A.2 – Continuação da lista de publicações da página anterior.**

| Referência | Título | Pontos-Chave |
|---|---|---|
| (MACHADO et al., 2016) | Methodology for Identifying the Cybersecurity Maturity Level of Smart Grids | Mostra a importância da identificação do nível de maturidade de segurança da informação em redes elétricas inteligentes, apresenta uma metodologia para avaliação do nível de maturidade de segurança cibernética em redes inteligentes. |
| (XU, M. et al., 2016) | Toward Engineering a Secure Android Ecosystem: A Survey of Existing Techniques | Faz uma análise dos problemas e soluções de segurança no software Android, identifica uma série de metodologias intensamente pesquisadas e ainda, problemas fundamentais são revelados, por fim, propõem uma abordagem holística no nível do ecossistema Android. |
| | | Continua na próxima página. |

**Tabela A.2 – Continuação da lista de publicações da página anterior.**

| Referência | Título | Pontos-Chave |
|---|---|---|
| (TAUSIF et al., 2017) | Towards Designing Efficient Lightweight Ciphers for Internet of Things | Apresenta 13 cifras de bloco de peso leve que são comparadas e ranqueadas baseadas no tamanho do código, ocupação da RAM e tempo de execução. |
| (KHAN, N. et al., 2017) | Performance Analysis of Security Algorithms for IoT devices | Mostra uma visão geral de alguns algoritmos implementados em ambiente restrito, contribuindo para a compreensão e criação de algoritmos de segurança para dispositivos de IoT comparando graficamente dentre os resultados apresentados no artigo. |
| (XU, Y. et al., 2018) | Research and Design of Reconfigurable Security Resource Pool Framework | Combina um chip reconfigurável e um serviço de segurança em nuvem, com base na plataforma de nuvem em código aberto OpenStack para prototipagem. |
| | | Continua na próxima página. |

**Tabela A.2 – Continuação da lista de publicações da página anterior.**

| Referência | Título | Pontos-Chave |
|---|---|---|
| (MUTHAVHINE; SUMBWANYAMBE, 2018) | An Analysis And A Comparative Study Of Cryptographic Algorithms Used On The Internet Of Things (IoT) Based On Avalanche Effect | Apresenta alguns algoritmos criptográficos de bloco existentes, analisa o efeito avalanche e a velocidade dos algoritmos criptográficos coletados. Através de um modelo matemático projetado cria uma melhora na velocidade e efeito avalanche. |
| (EL-HAII et al., 2018) | Analysis of Cryptographic Algorithms on IoT Hardware platforms | Analisa a atuação de diferentes algoritmos criptográficos no Raspberry Pi3 modelo B e compara com outros dispositivos de baixo custo. |
| (LOPES JUNIOR et al., 2019) | Analysis and Comparison of Cryptographic Algorithms applied to IoT | Compara cinco algoritmos criptográficos de cifra leve utilizando o ESP8266 (comumente utilizado em aplicações IoT), sendo o desempenho dos algoritmos avaliados quanto ao tempo de execução, consumo energético e vazão. |

Continua na próxima página.

**Tabela A.2 – Continuação da lista de publicações da página anterior.**

| Referência | Título | Pontos-Chave |
|---|---|---|
| (WANG et al., 2019) | Dynamically Reconfigurable Encryption and Decryption System Design for the Internet of Things Information Security | Projeta um sistema de criptografia e decodificação dinamicamente reconfigurável baseado em Field Programmable Gate Array, o que reduz o consumo de recursos lógicos e melhora a utilização e flexibilidade do sistema. Traz resultados quanto a comparação com o sistema estático de criptografia e decriptografia. |
| (SINGH, P.; ACHARYA; CHAURASIYA, 2019) | A comparative survey on lightweight block ciphers for resource constrained applications | Faz uma revisão de soluções criptográficas leves desenvolvidas recentemente para dispositivos de baixo custo, fornece informações sobre a área de aplicação das cifras leves, suas tendências de segurança e métricas de desempenho, facilitando o uso em ambientes com restrição de recursos (RCEs). |

Continua na próxima página.

**Tabela A.2 – Continuação da lista de publicações da página anterior.**

| Referência | Título | Pontos-Chave |
|---|---|---|
| (BEG; AL-KHAROBI; AL-NASSER, 2019) | Performance Evaluation and Review of Lightweight Cryptography in an Internet-of-Things Environment | Faz análises em algoritmos criptográficos leves (simétricos), observa as métricas de atuação relevantes para dispositivos de IoT, simulados no Cooja e sendo o resultado apresentado do ponto de vista de atraso, consumo de energia e tamanho de código. |
| (MARAM et al., 2019) | Intelligent security algorithm for UNICODE data privacy and security in IOT | Apresenta uma caixa de substituição (S-box) dinâmica e dependente de chave, a caixa funciona com textos UNICODE, usando a linguagem Python. |
| (AHTSHAM; YAN; ALI, 2019) | IoT based door lock surveillance system using cryptographic algorithms | Apresenta uma fechadura inteligente de alta segurança utilizando AES-128 e SHA-512, através de um aplicativo Andorid. |
| | | Continua na próxima página. |

**Tabela A.2 – Continuação da lista de publicações da página anterior.**

| Referência | Título | Pontos-Chave |
|---|---|---|
| (PINDAR et al., 2020) | A Lightweight Message Authentication Code for Virtual Work in Future Smart Cities | Apresenta o algoritmo MAC baseado na função de transformação de frase de quase-grupo (permutation-quasigroup) simulado no Raspberry Pi B+, mostrando que o algoritmo roda suficientemente bem no dispositivo, porém não se sabe o quanto. |
| (DHANDA; SINGH, B.; JINDAL, 2020) | Lightweight Cryptography: A Solution to Secure IoT | Apresenta um levantamento aprofundado e avançado de algoritmos criptográficos disponíveis até 2019. As cifras são comparadas dentro de seus grupos em termos de área do chip, energia e potência, eficiência de hardware e software, taxa de transferência, latência e figura de mérito. |

Continua na próxima página.

**Tabela A.2 – Continuação da lista de publicações da página anterior.**

| Referência | Título | Pontos-Chave |
|---|---|---|
| (PALLAVI; KUMAR; SRIKRISHNA, 2020) | Comparative Study of Various Lightweight Cryptographic Algorithms for Data Security Between IoT and Cloud | Mostra uma visão mais teórica do porquê a criptografia é necessária, quais os tipos (Simétrica e assimétrica) e os tipos de ataques, traz diferentes algoritmos criptográficos e informações que devem ser levadas em conta na escolha do melhor custo benefício para o dispositivo de IoT utilizado. |
| (KHAN, M. N.; RAO; CAMTEPE, 2020) | Lightweight Cryptographic Protocols for IoT Constrained Devices: A Survey | Analisa protocolos criptográficos leves para dispositivos inteligentes (IoT), concluindo que não existe um protocolo que aborde a assimetria de recursos dos dispositivos IoT e se adapte à capacidade de recursos desses dispositivos. Sugerindo um modelo *end-through-nodes-to-end*. |

Continua na próxima página.

**Tabela A.2 – Continuação da lista de publicações da página anterior.**

| Referência | Título | Pontos-Chave |
|---|---|---|
| (KANE et al., 2020) | Security and Performance in IoT: A Balancing Act | Apresenta uma estrutura para analisar métricas de desempenho de microcontroladores que são usadas em implantações de IoT, realiza medições nas principais áreas de consumo de energia, custo de tempo e custo de energia além de medições do pico de uso de RAM e flash. |
| (WU; DWIVEDI; SRIVASTAVA, 2021) | Security and Privacy of Patient Information in Medical Systems Based on Blockchain Technology | Mostra um método para proteger a privacidade das informações de sistemas médicos baseado na tecnologia de blockchain, analisa a superioridade desse método em aspectos como eficiência de armazenamento e transmissão de informações e ainda o desempenho do controle de segurança por meio de experimentos e simulações. |