UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Análise e Otimização do Algoritmo Criptográfico AES para Utilização em Aplicações IoT

Yuri Silva Vaz

Pelotas, 2024

Yuri Silva Vaz

Análise e Otimização do Algoritmo Criptográfico AES para Utilização em Aplicações IoT

Dissertação apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Júlio Carlos Balzano de Mattos
Coorientador: Prof. Dr. Rafael Iankowski Soares

Pelotas, 2024

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

V393a Vaz, Yuri Silva

Análise e otimização do algoritmo criptográfico AES para utilização em aplicações IoT [recurso eletrônico] / Yuri Silva Vaz ; Júlio Carlos Balzano de Mattos, orientador ; Rafael Iankowski Soares, coorientador. — Pelotas, 2024.
75 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2024.

1. Internet das coisas. 2. Segurança. 3. Criptografia leve. 4. Otimização. I. Mattos, Júlio Carlos Balzano de, orient. II. Soares, Rafael Iankowski, coorient. III. Título.

CDD 005

Elaborada por Simone Godinho Maisonave CRB: 10/1733

Yuri Silva Vaz

Análise e Otimização do Algoritmo Criptográfico AES para Utilização em Aplicações IoT

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 19 de abril de 2024

Banca Examinadora:

Prof. Dr. Júlio Carlos Balzano de Mattos (orientador)
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Rafael Iankowski Soares (co-orientador)
Doutor em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul.

Prof. Dr. Adenauer Correa Yamin
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Felipe de Souza Marques
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Marcio Seiji Oyamada
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Dedico este trabalho a todas as pessoas maravilhosas que tenho ao meu redor.

AGRADECIMENTOS

Primeiramente, agradeço imensamente a minha mãe **Carla Denise Silva da Silva** e a meu pai **Jorge Luiz Garcia Vaz** por todo o apoio nesta sequência no mundo acadêmico. Sem vocês, nada disto seria possível!

Também agradecer a meus padrinhos **Rozane Celina Silva Mattoso** e **Rudi Glênio Sias Mattoso** pelo imenso apoio e incentivo constante nesta caminhada.

A **Jair Quiroz** pela parceria, pelas diversas ajudas e, principalmente, pelo apoio e incentivo para seguir no mundo acadêmico e iniciar a caminhada na pós-graduação.

A meu irmão **Yago Caldeira** e a **Fernanda Sodré** pela parceria e por sempre trazerem palavras de apoio e incentivo.

Por fim, a meu orientador **Júlio Carlos Balzano de Mattos** e meu co-orientador **Rafael Iankowski Soares** por todas as ajudas e conselhos no desenvolvimento deste trabalho. Vocês foram peças fundamentais para a conclusão desta dissertação e também para os dois artigos que publicamos ao longo desta jornada. Mais que orientadores, vocês são grandes amigos!

A todos vocês, meu muito obrigado!

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

O sucesso é a soma de pequenos esforços repetidos dia após dia.
— ROBERT COLLIER

RESUMO

VAZ, Yuri Silva. **Análise e Otimização do Algoritmo Criptográfico AES para Utilização em Aplicações IoT**. Orientador: Júlio Carlos Balzano de Mattos. 2024. 75 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2024.

O universo de Internet das Coisas (IoT) vem se expandindo a cada ano, apresentando aplicações nos mais diversos segmentos. Contudo, esse crescimento gera uma quantidade massiva de dados que circulam pela Internet diariamente, logo, a preocupação com a segurança dessas informações é primordial. Criptografia é uma boa estratégia para a proteção dos dados, porém, os algoritmos criptográficos clássicos são muito custosos em termos de desempenho e consumo energético, o que é não é adequado para aplicações IoT, onde normalmente são implementadas em dispositivos com recursos computacionais limitados. Esta dissertação propõe otimizações no algoritmo criptográfico AES, especificamente nos estágios *SubBytes* e *MixColumns*, com o objetivo de viabilizar aplicações IoT seguras em dispositivos de pouco poder computacional, o tornando um algoritmo leve (*lightweight algorithm*). As otimizações propostas apresentam resultados significativos em termos de desempenho, sem impactar a segurança do algoritmo, e também em termos de consumo energético. Em termos de desempenho, a versão proposta obteve uma redução média de 86,71% em tempo de execução e ainda, reduções de 31,82% e 89,04% em alocação de memória de programa e memória dinâmica, respectivamente, comparado ao AES original. Já em questões de segurança, foi obtido 50,41% de efeito avalanche e uma melhor distribuição de 0's e 1's no *ciphertext* comparado a versão original, além de ser aprovado nos testes do NIST. Por fim, em questões de consumo energético, a versão otimizada proposta por esta dissertação atingiu reduções de 63,87% e 70,86% quando executada nas plataformas ESP-WROOOM-32 e Raspberry Pi Pico, respectivamente, o que o torna muito mais atrativo para aplicações IoT.

Palavras-chave: internet das coisas; segurança; criptografia leve; otimização.

ABSTRACT

VAZ, Yuri Silva. **Analysis and Optimization of the AES Cryptographic Algorithm for Use in IoT Applications**. Advisor: Júlio Carlos Balzano de Mattos. 2024. 75 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2024.

The Internet of Things (IoT) universe is expanding every year, featuring applications in various sectors. However, this growth generates a massive amount of data that flows through the Internet daily, therefore, the concern for the security of this information is of paramount importance. Cryptography is a good strategy for data protection, however, classical cryptographic algorithms are highly costly in terms of performance and energy consumption, which is particularly detrimental for IoT applications, as most of them are implemented on devices with limited computational resources. This dissertation proposes optimizations to the AES cryptographic algorithm, specifically in the SubBytes and MixColumns stages, with the aim of enabling secure IoT applications on resource constrained devices, making it a lightweight algorithm. The proposed optimizations yield significant results in terms of performance, without compromising the security of the algorithm, as well as in terms of energy consumption. In terms of performance, the proposed version achieved an average reduction of 86.71% in execution time and, furthermore, reductions of 31.82% and 89.04% in program memory and dynamic memory allocation, respectively, compared to the original AES. Regarding security, a 50.41% avalanche effect was achieved, along with a better distribution of 0's and 1's in the ciphertext compared to the original version. Moreover, it received approval in the NIST tests. Finally, in terms of energy consumption, the optimized version proposed by this dissertation achieved reductions of 63.87% and 70.86% when executed on ESP-WROOOM-32 and Raspberry Pi Pico platforms, respectively, making it much more attractive for IoT applications.

Keywords: internet of things; security; lightweight cryptography; optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AES	Advanced Encryption Standard
CoAP	Constrained Application Protocol
CSV	Comma-Separated Values
DES	Data Encryption Standard
GF	Galois Field
IoT	Internet of Things
LWC	Lightweight Cryptography
MQTT	Message Queuing Telemetry Transport
NIST	National Institute of Standards and Technology
PPK2	Power Profiler Kit II
RSA	Rivest-Shamir-Adleman
3DES	Triple Data Encryption Standard

SUMÁRIO

1 INTRODUÇÃO

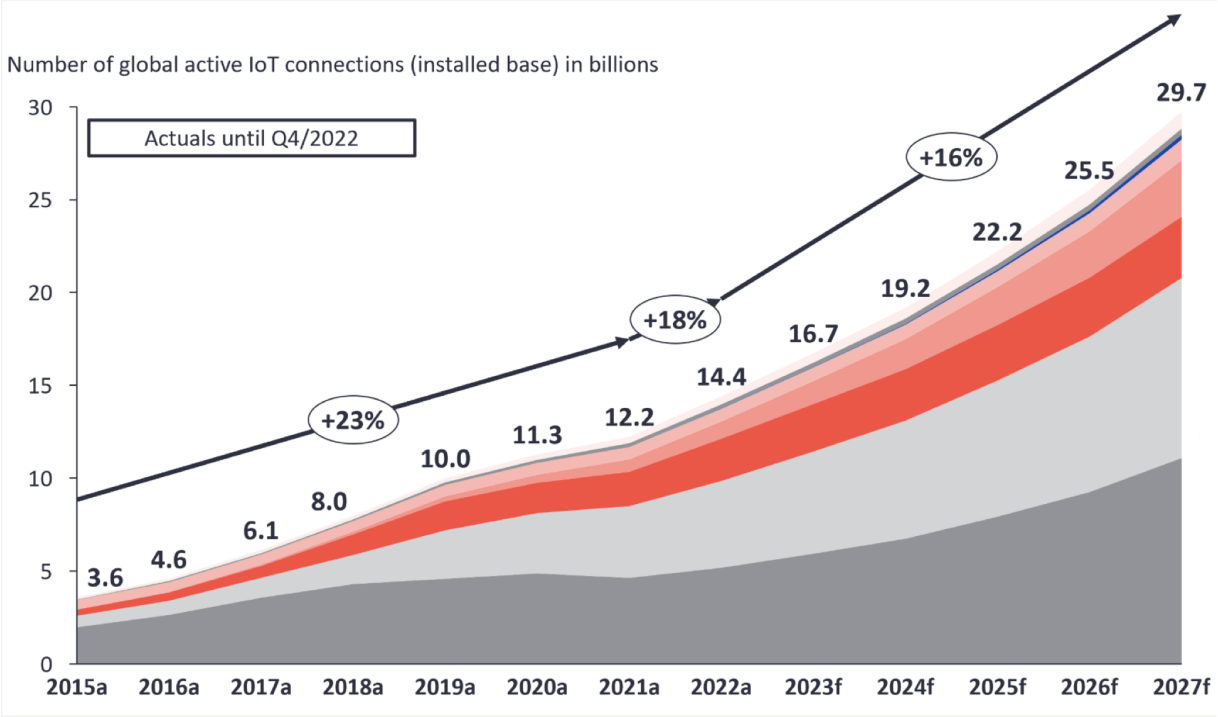
A Internet das Coisas, do inglês *Internet of Things* (IoT), pode ser definida como a presença pervasiva de uma variedade de dispositivos, tais como sensores e/ou atuadores, que têm a capacidade de interação entre si, com a motivação de atingirem um objetivo comum (Marwedel, 2021). Estes dispositivos, em geral, enviam suas informações para uma central de processamento onde, com base nos dados coletados, será tomada alguma ação. Outra definição trazida por Maccarrone (2023) é que IoT se refere a objetos do dia a dia que estão interconectados, podendo ou não estarem embarcados em objetos maiores, possibilitando a coleta e troca de informações. Dispositivos IoT estão cada vez mais presentes em nosso cotidiano, podendo eles estarem dentro de casas, indústrias ou até mesmo monitorando cidades inteiras (Sadhu; Yanambaka; Abdelgawad, 2022).

De acordo com o relatório apresentado por Mckinsey (2022), dentre as nove principais categorias em que aplicações IoT podem estar divididas estão: **serviços médicos:** dispositivos acoplados ou vestíveis em corpos humanos com o objetivo de monitorar sinais vitais; **casas inteligentes:** assistentes inteligentes comandados por voz, aspiradores inteligentes e etc; **lojas inteligentes:** dispositivos que podem ser instalados em lojas, bancos e restaurantes para facilitar *self-checkout* ou para organizar o inventário; **escritórios inteligentes:** dispositivos que gerenciam energia ou segurança; **ambientes de produção inteligentes:** aumento na eficiência de linhas de produção ou controle de estoque; **veículos Inteligentes:** controle de manutenção de carros, aviões e navios; **cidades inteligentes:** semáforos inteligentes, monitoramento do meio ambiente e gerenciamento de recursos naturais; **ambientes urbanos:** monitoramento de estradas e veículos autônomos para análise em tempo real de rotas.

Estas aplicações estão crescendo em número anualmente, visto que no ano de 2022 já haviam 14,3 bilhões de dispositivos IoT ativos e, de acordo com Sinha (2023), espera-se que esse número atinja, aproximadamente, 29 bilhões de dispositivos no ano de 2027. A Figura 1 demonstra os números já consolidados até o ano de 2022 e também mostra a previsão até o ano de 2027. Devido a este rápido crescimento

em número de aplicações IoT, mais dispositivos estão conectados a Internet, resultando em uma enorme quantidade de dados que trafegam nestas aplicações (Saraiva; Leithardt; Paula; Mendes; González; Crocker, 2019).

Figura 1 – Aplicações IoT em Números



Fonte: (Sinha, 2023).

Visto que há uma forte projeção de crescimento em número destas aplicações, é necessária a preocupação com a segurança dos dados que trafegam entre os dispositivos, pois uma aplicação IoT vulnerável pode acarretar em sérios problemas com vazamentos e manipulações dos dados (Mishra; Pandya, 2021). A grande quantidade de dados que as aplicações IoT geram e como os protegê-los é ainda um dos maiores desafios da comunidade IoT (Saraiva; Leithardt; Paula; Mendes; González; Crocker, 2019).

No primeiro semestre de 2022 haviam sido registrados 57 milhões de ataques a aplicações IoT e, no mesmo período de 2023, foram registrados 77,9 milhões de ataques, o que representa um aumento de 37% no mesmo período (Wolff, 2023). Estes números apontam que as aplicações IoT estão cada vez mais sendo visadas por cibercriminosos. No ano de 2022, o valor gasto com crimes cibernéticos a nível mundial foi de três trilhões de dólares e, de acordo com previsões, este valor atingirá a cifra de 10,5 trilhões de dólares no ano de 2025 (Mullen, 2023).

Uma possível solução para a proteção dos dados que trafegam pelas aplicações é o uso de algoritmos criptográficos, contudo, estes algoritmos podem apresentar um

impacto em termos de desempenho (Ahmad, 2018), o que é um grande problema devido a grande maioria das aplicações IoT serem desenvolvidas em dispositivos com recursos computacionais limitados, além da maior parte dos dispositivos serem alimentados por baterias (Tsai; Huang; Leu; You; Huang; Tsai, 2018).

Visto que a criptografia provoca necessidade de mais recursos em termos de desempenho e consumo energético, surge então uma classe de algoritmos criptográficos focados em aplicações IoT e baixo consumo. Estes são chamados de *lightweight cryptography*(LWC), que em tradução direta seriam algoritmos de criptografia leves. Eles são um método de encriptação que prezam por ocupar pouco espaço de armazenamento e também por utilizar pouco recurso computacional (Toshihiko, 2017).

De fato, a melhor solução encontrada pelos pesquisadores para enfrentar o problema da segurança em dispositivos com recursos computacionais limitados é utilizando LWC. Há uma imensa demanda por algoritmos de criptografia leves que sejam confiáveis e que tenham um bom custo-benefício. É muito mais eficiente aprimorar a segurança através de algoritmos criptográficos de baixo consumo do que substituir os dispositivos por algo com maior poder computacional (Ullah; Radzi; Yazdani; Alshehri; Khan, 2022).

Dos algoritmos criptográficos de chave simétrica, o AES é até hoje uma interessante solução de segurança, já tendo sido amplamente testado por agências governamentais e pela comunidade acadêmica. O AES possui um bom compromisso entre tempo de execução, simplicidade de implementação e nível de segurança comparado a outros algoritmos, apresentando uma estrutura de funções facilmente decompostas, o que o torna um dos algoritmos mais usados na literatura para análises e avaliações (Medeleanu; Racuciu; Rogobete, 2015).

Na literatura, foram encontrados alguns trabalhos que buscam otimizar o algoritmo criptográfico AES, de forma com que sua implementação seja viável em aplicações IoT. Dentre estes trabalhos, são utilizadas diferentes abordagens de otimização, como exemplo, uso de mapas caóticos, diferentes modelos de s-box e uso de tabelas pré computadas.

Exemplos destes trabalhos são os de Mohammad; Abdullah (2022), Fadhil; Farhan; Fadhil; Al-saidi (2020), Hammod (2022), Shi; Wang; Jia; Peng; Jiang; Zhu (2019) e Naif; Abdul-majeed; Farhan (2019), nos quais foram utilizados o algoritmo AES como base para otimizações, porém, estes trabalhos implementaram apenas um tipo de otimização, e é justamente o que os diferenciam da proposta desta dissertação, na qual propõe otimização em duas vertentes, que são consumo de memória e desempenho. Ademais, este trabalhou também verificou questões de consumo energético, o que não foi verificado pelos demais autores.

Este trabalho tem como objetivo propor uma versão *lightweight* do algoritmo criptográfico AES, combinando duas otimizações de modo que sua utilização seja viável

em aplicações IoT. Para isto, tem-se como objetivos específicos:

- Propor novas modificações no estágio *SubBytes*, com o intuito de reduzir consumo de memória
- Propor novas otimizações no estágio *MixColumns*, de modo a reduzir o tempo de execução do algoritmo
- Avaliar o consumo energético do algoritmo proposto em diferentes plataformas microcontroladas
- Avaliar a segurança do algoritmo proposto

1.1 Organização do Texto

Este texto está organizado da seguinte forma: o Capítulo 2 apresenta um referencial teórico sobre Internet das Coisas, criptografia e traz uma apresentação do algoritmo AES, de modo a facilitar o entendimento das otimizações realizadas ao longo do trabalho, além de apresentar as métricas de segurança que serão utilizadas nos resultados e, por fim, apresenta a revisão bibliográfica realizada; Capítulo 3 apresenta o desenvolvimento do trabalho, desde a análise inicial do algoritmo AES, passando pelas otimizações propostas e finalizando com as medições de potência e energia; Capítulo 4 traz os resultados obtidos, em termos de desempenho, segurança e energia e, por fim, Capítulo 5 apresenta as conclusões obtidas.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico necessário para o entendimento de tópicos contidos nesta dissertação. Inicialmente apresenta-se o conceito de Internet das Coisas, seguido por uma breve introdução sobre criptografia e seus principais tipos, tendo na sequência uma passagem sobre o que são algoritmos de criptografia leves. Também é apresentado o algoritmo AES, trazendo um pouco de sua história e de sua estrutura e, por último, são apresentadas algumas das métricas de avaliações utilizadas nos resultados desta dissertação.

2.1 Internet das Coisas

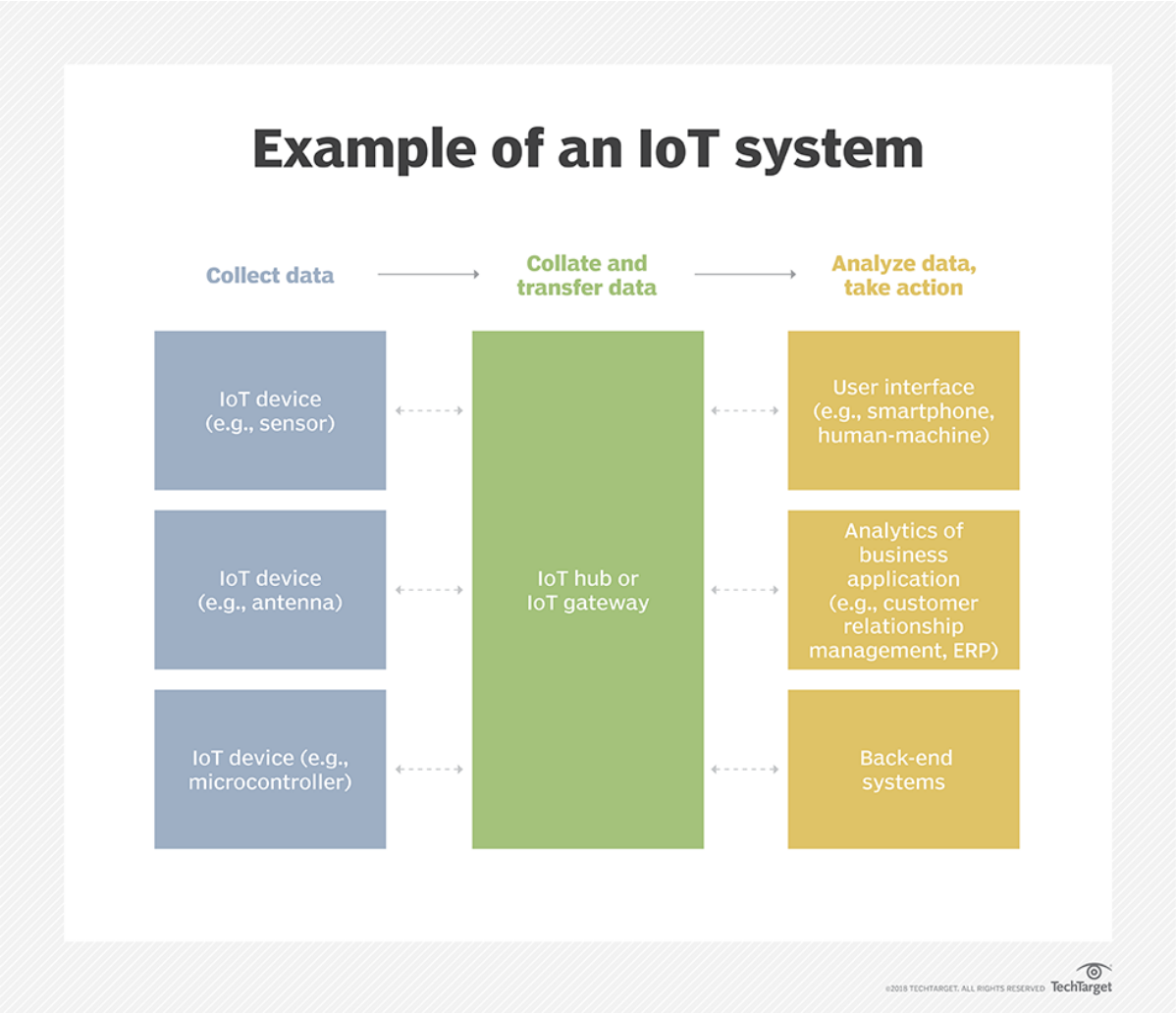
A primeira aparição do termo *Internet of Things* se deu durante uma apresentação de Kevin Ashton no ano de 1999. Ele acreditava que a maneira com que interagíamos e vivíamos com objetos físicos que estavam em nosso redor deveria ser reconsiderada, devido aos grandes avanços da computação e da Internet na época (Ashton, 2009). Uma definição atual de Internet das Coisas, conforme descrito por Gillis (2023), é uma rede de dispositivos interligados que coletam e transferem dados com outros dispositivos IoT ou até mesmo com servidores.

Pode-se tomar como exemplo uma aplicação do campo de agricultura inteligente, no qual podem-se ter diversos sensores espalhados pelos hectares de uma plantação coletando informações de temperatura, umidade e pH do solo. Estas informações podem ser transmitidas para uma central de processamento, onde, com base nestes dados, pode verificar se o solo necessita irrigação, acionando automaticamente os irrigadores da plantação ou, com base no pH do solo informar o fazendeiro por algum aplicativo que o solo necessita adubagem, por exemplo.

Uma modelo de sistema muito comum em aplicações IoT é o apresentado pela Figura 2, no qual há uma etapa de coleta de dados por um ou mais sensores, uma etapa de transmissão destes dados por um *gateway*, utilizando algum protocolo de comunicação como CoAP ou MQTT e, por último, uma etapa de armazenamento e/ou processamento destes dados para análises ou tomada de ação, por exemplo acio-

nando algum atuador.

Figura 2 – Exemplo de Modelo de Sistema Utilizado em Aplicações IoT



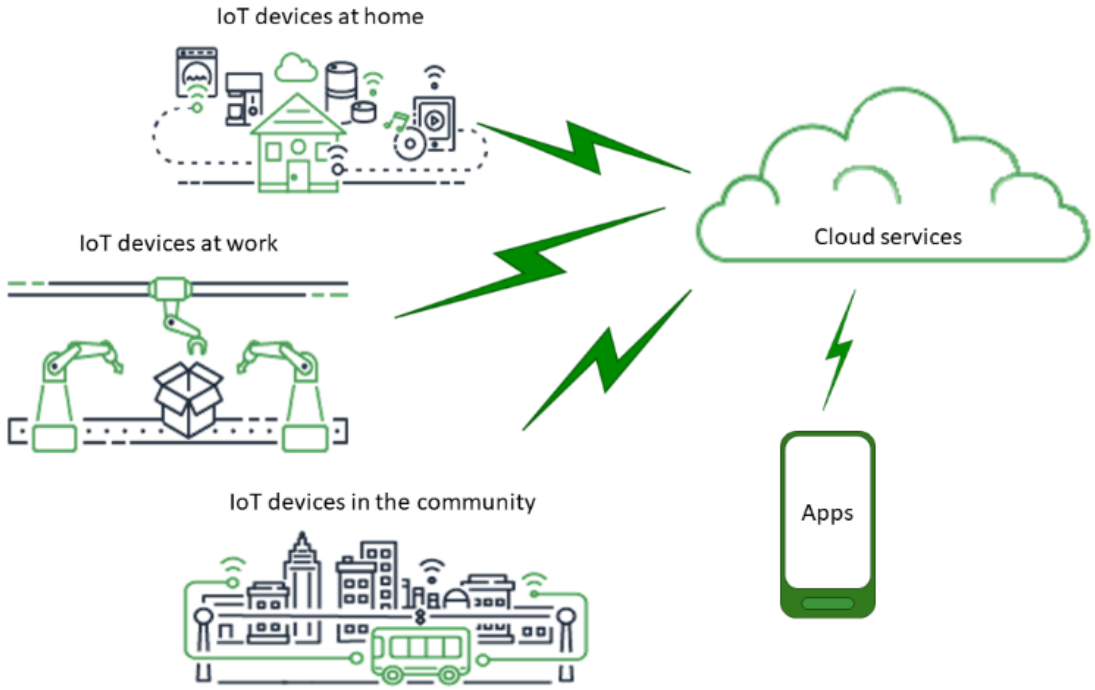
Fonte: (Gillis, 2023).

Uma ”coisa” na Internet das Coisas pode ser, por exemplo, uma pessoa com um monitor cardíaco implantado, um animal em uma fazenda com um biochip, um automóvel que tem um sensor nos pneus para alertar o motorista quando a pressão dos pneus estiver baixa ou qualquer outro objetivo que possa se conectar a Internet para coletar e/ou transmitir dados (Gillis, 2023).

A Figura 3 mostra diversos segmentos onde há possibilidade de aplicações IoT, como casas, indústrias e cidades inteligentes que enviam dados para um servidor e, por exemplo, podem ser acessadas em tempo real a partir de um *smartphone*. IoT tem um grande impacto na vida humana facilitando tarefas diárias, por exemplo, pode-se ter um despertador que, ao apertar o botão soneca, automaticamente ligue a máquina de café e abra as cortinas, ou uma geladeira inteligente que, ao detectar que os alimentos que mais são consumidos estão acabando, pode enviar uma notificação

para o *smartphone* do usuário ou até mesmo solicitar a entrega dos itens em domicílio (Amazon, 2023).

Figura 3 – Exemplos de Segmentos de Aplicações IoT



Fonte: (Amazon, 2023).

Estima-se que, no ano de 2025, cada pessoa irá se conectar com pelo menos um dispositivo inteligente a cada 18 segundos, gerando uma troca de informação entre eles. Muitas destas interações são causadas pelo rápido avanço em número de dispositivos IoT ao redor do globo, o que gerará uma criação de 90ZB de dados em 2025 (Reinsel; Gantz; Rydning, 2018).

Visto que as aplicações estão crescendo rapidamente em número, assim como a quantidade de dados que estão sendo gerados e transmitidos, é necessária a preocupação com a proteção destas informações que estão trafegando via Internet, uma vez que ataques e/ou vazamentos destes dados pode ser muito prejudicial para seus usuários. Uma possível solução para a proteção das aplicações IoT é a introdução de algoritmos criptográficos para proteção dos dados.

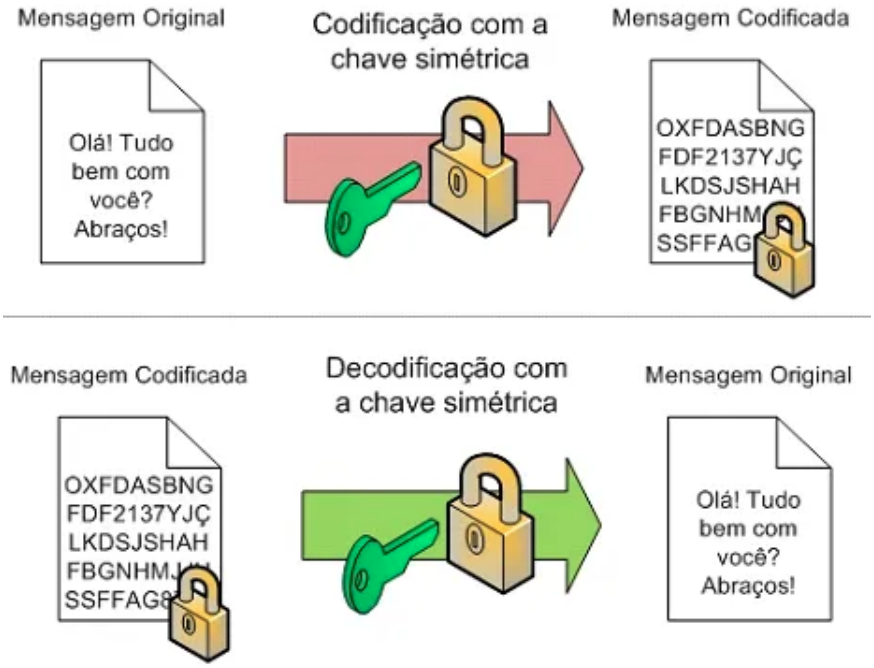
2.2 Criptografia

Criptografia é um conjunto de técnicas utilizadas para tornar ininteligível uma informação, de modo que quem não possua conhecimento de como foi feita a encriptação não consiga compreendê-la (Singh, 2000).

As duas principais técnicas de encriptação são criptografia de chave simétrica e criptografia de chave assimétrica (Srivastava; Tiwari; Srivastava, 2022). Todos os sistemas de criptografia se baseiam no conceito de chave, que pode ser definida como a base de transformação de uma mensagem comum em uma mensagem ilegível (Ibm, 2022).

Na criptografia de chave simétrica é utilizada uma única chave, tanto para encriptar quanto para decriptar a mensagem. Na Figura 4 é possível verificar que ambas as chaves são iguais em ambos os processos de cifragem e decifragem. Os principais algoritmos de chave simétrica são: *Data Encryption Standard*(DES), *Triple DES*(3DES) e *Advanced Encryption Standard*(AES).

Figura 4 – Exemplo de Criptografia de Chave Simétrica



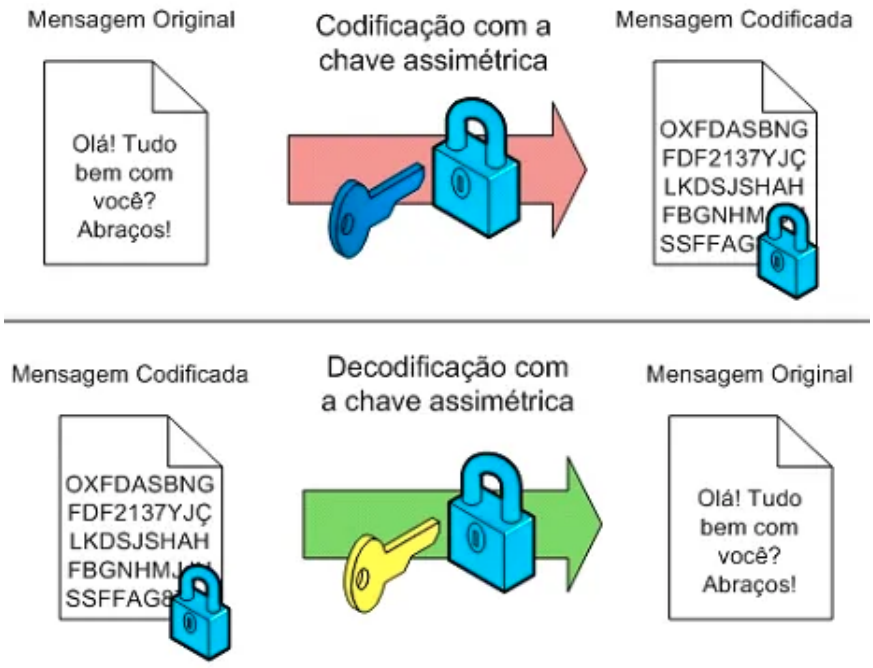
Fonte: (Luz, 2021).

A criptografia de chave simétrica é, em geral, mais rápida de executar do que a de chave assimétrica, o que é muito interessante para aplicações IoT, pois necessita menos recurso computacional para performar a encriptação. Contudo, a grande desvantagem é que se alguém tiver acesso a chave de encriptação, que neste caso é única, terá acesso a toda a informação.

Já na criptografia de chave assimétrica são utilizadas duas chaves, uma pública e uma privada. A chave pública é amplamente divulgada, todos podem ter conhecimento da mesma, enquanto a chave privada deve ser secreta de cada usuário. Logo, se uma pessoa X quer enviar uma mensagem secreta para uma pessoa Y, X deve encriptar a mensagem com a chave pública de Y e enviar a mensagem. Quando

Y recebe a mensagem, ele deve decriptá-la com sua chave privada, tendo total acesso ao conteúdo da mensagem enviada por X. Na Figura 5 é possível observar que as chaves utilizadas nos processos de codificação (encriptação) e decodificação (decriptação) são diferentes. O algoritmo de chave assimétrica mais conhecido e utilizado é o Rivest-Shamir-Adleman (RSA) (Yao, 2023).

Figura 5 – Exemplo de Criptografia de Chave Assimétrica



Fonte: (Luz, 2021).

A grande vantagem deste tipo de encriptação é que se alguém tiver acesso a chave pública do usuário, os dados não serão comprometidos pois somente a chave privada pode decifrá-los. Por outro lado, este tipo de criptografia é bem mais lenta que a de chave simétrica, demandando maior poder computacional para performá-lo.

2.2.1 Criptografia Leve

Os padrões de criptografia atuais foram desenvolvidos para computadores pessoais e servidores, no qual, em geral, não possuem recursos computacionais limitados. Além disto, a complexidade dos algoritmos e as chaves de encriptação estão cada vez maiores, o que requer mais poder computacional e disponibilidade de memória (Shaikh; Hajjej; Uslu; Yüksel; Dınçer; Alroobaea; Baqasah; Chinta, 2024). Também, na atualidade, com o surgimento da computação quântica, surgem novas possibilidades para a criptografia, tendo máquinas com poder computacional cada vez maiores (Giroti; Malhotra, 2022).

No entanto, objetos inteligentes que são conectados à Internet, em sua maioria, têm limitações em questões como memória e energia, fazendo com que a implementação dos algoritmos de criptografia tradicionais seja inviável. O rápido crescimento em número de aplicações IoT fez com que opções mais leves de criptografia fossem investigadas, assim, surgindo uma nova classe de algoritmos criptográficos chamada *lightweight cryptography*, definida por Toshihiko (2017) como um método de encriptação leve que tem como prioridade um baixo consumo de memória e/ou baixa complexidade computacional.

Algoritmos criptográficos leves não oferecem um nível mais baixo de segurança, mas sim levam em conta as limitações específicas que pequenos dispositivos utilizados em aplicações IoT utilizam na hora de desenvolver os algoritmos, assim, criando soluções mais apropriadas (Future, 2022).

Na literatura, foi possível encontrar alguns trabalhos que realizam avaliações de algoritmos criptográficos leves, como os de Gunathilake; Al-dubai; Buchana (2020) e A.mohammed; Hussein (2023), onde foram comparadas características e as estratégias de otimizações. Já o trabalho de Albarello; Oyamada; Camargo (2020) além de avaliar diferentes métricas de algoritmos criptográficos, também propõe um protocolo leve de comunicação entre dispositivos IoT, com o intuito de garantir confidencialidade e integridade na troca de mensagens.

Devido a grande maioria das aplicações IoT serem implementadas em dispositivos com recursos computacionais limitados, é que cifras de bloco convencionais, tais como o AES, são muito custosas em termos de processamento, o que torna sua implementação inviável em aplicações IoT (Zakaria; Azni; Ridzuan; Zakaria; Daud, 2023).

Cifras de bloco leves têm se tornado o padrão de segurança utilizado para proteção de aplicações IoT. O conceito de algoritmo leve tem atraído a atenção de indústrias, agências governamentais e também da comunidade acadêmica (Zakaria; Azni; Ridzuan; Zakaria; Daud, 2023). Embora o algoritmo original AES seja muito custoso para implementar em aplicações IoT, ele é muito utilizado como base para o desenvolvimento de versões leves a partir de suas funções, sendo alguns de seus componentes reaproveitados e combinados com outras técnicas (Zakaria; Azni; Ridzuan; Zakaria; Daud, 2023).

Visto que o algoritmo AES é muito utilizado como base para otimizações (Salman; Farhan; Shakir, 2022a), este será então o algoritmo analisado por este trabalho, de forma a encontrar novas oportunidades de otimização tanto em termos de desempenho quanto em termos de consumo de memória.

2.2.2 Algoritmo AES

O algoritmo *Advanced Encryption Standard*(AES) foi desenvolvido em 2001 por Vincent Rijmen e Joan Daemen e foi selecionado pelo NIST(*National Institute of Standards and Technology*) como o algoritmo padrão de criptografia. Ele é um algoritmo baseado em uma rede de substituição e permutação, que utiliza o conceito de chave simétrica, ou seja, a mesma chave é usada para encriptar e decriptar a informação. O algoritmo opera com um número determinado de rodadas, no qual este número é estabelecido de acordo com o tamanho da chave, que pode ser de 128, 192 ou 256 bits, gerando um total de 10, 12 e 14 rodadas, respectivamente.

O AES é um algoritmo de cifra de bloco, onde cada bloco possui um tamanho fixo de 128 bits (16 bytes). Estes 16 bytes de entrada são mapeados em uma matriz 4x4, chamada matriz de estado, e são ordenados por colunas. A Figura 6 exemplifica o processo de mapeamento da string de entrada, em hexadecimal, para a matriz de estado. Esta matriz percorre por todos os estágios e rodadas do algoritmo, sendo nela realizadas todas as modificações que ocorrem em cada um dos estágios. Ao final das 10 rodadas, esta matriz de estado estará contendo o texto final encriptado.

Figura 6 – Matriz de Estado

Plaintext: 66630efac432c45664098000bdebe9a7

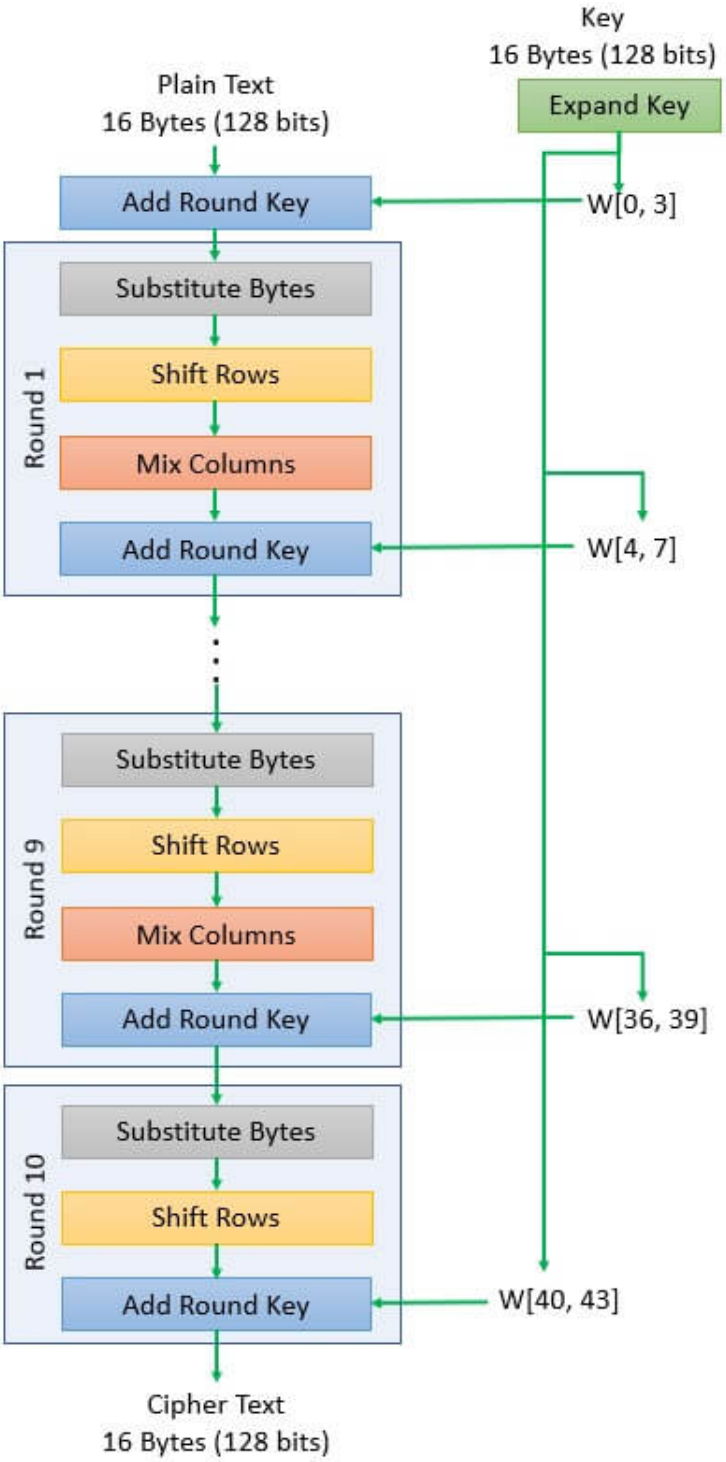
Matriz de estado:

66	c4	64	bd
63	32	09	eb
0e	c4	80	e9
fa	56	00	a7

Fonte: Elaborada pelo autor.

A entrada do algoritmo é um bloco de 16 bytes, chamado de *plaintext*. Este bloco vai passar pelas 10 rodadas do algoritmo (considerando uma chave de 128 bits), onde cada rodada consiste em 4 estágios (com exceção da última rodada, que são apenas 3 estágios) que são: *Substitute Bytes*(SubBytes), *Shift Rows*, *Mix Columns* e *Add Round Key*. Ao final das 10 rodadas, o algoritmo tem como saída um bloco encriptado, chamado de *cipher text.* A Figura 7 ilustra a estrutura do algoritmo AES em sua versão de 10 rodadas.

Figura 7 – O Algoritmo AES



Fonte: (Neha, 2020).

2.2.2.1 *Estágio SubBytes*

O estágio *SubBytes* consiste numa etapa de substituição, onde cada um dos valores armazenados nas 16 posições da matriz de estado será substituído por um novo valor com base em uma tabela de substituição, que é chamada de s-box e contêm 256 posições, normalmente expressos em hexadecimal. Esta s-box consiste em uma matriz 16x16 e pode ser visualizada na Figura 8. Cada um dos 16 valores armazenados na matriz de estado são separados em dois dígitos, onde cada digito é utilizado para endereçar a linha e a coluna correspondente na s-box.

Figura 8 – Matriz 16x16 que Representa a S-box

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fonte: (Dworkin; Barker; Nechvatal; Foti; Bassham; Roback; Dray, 2023).

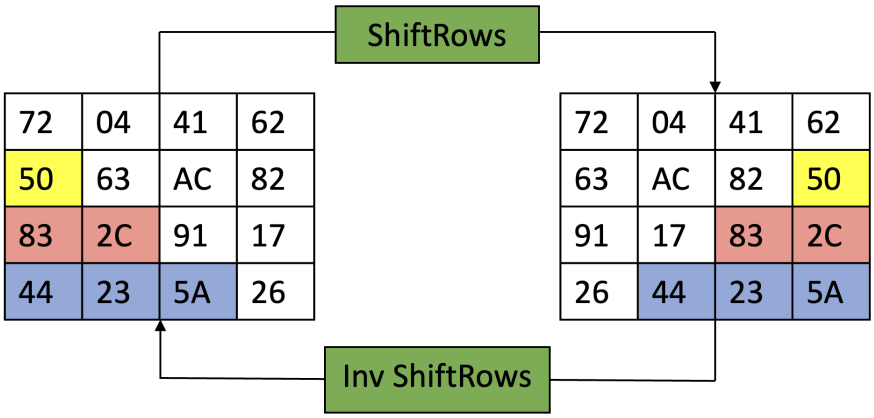
Por exemplo, se uma posição específica da matriz de estado contém o valor 75 armazenado, é então assumido o valor de x=7, que corresponde a linha da s-box, e o valor de y=5, que corresponde a coluna da s-box. Após localizar a posição corresponde na s-box, é então atualizado o valor na matriz de estado com o valor contido nesta posição da s-box, que no caso do exemplo utilizado seria o valor 9d. Este processo de substituição é realizado em cada uma das 16 posições da matriz de estado e, então, segue-se para o próximo estado do algoritmo.

2.2.2.2 *Estágio ShiftRows*

O estágio *ShiftRows* é uma etapa bem simples do algoritmo, que consiste em deslocamentos circulares na matriz de estado, onde cada uma das quatro linhas da matriz

se deslocam para a esquerda com um determinado valor. As linhas 0, 1, 2 e 3 da matriz se deslocam para a esquerda 0, 1, 2 e 3 vezes, respectivamente. A Figura 9 exemplifica este processo, mostrando a matriz de entrada antes e depois da operação de *ShiftRows*.

Figura 9 – Exemplo da Operação ShiftRows



Fonte: (A. mehdi; Jabbar; Fadhil; Abbood, 2018).

2.2.2.3 Estágio MixColumns

O estágio *MixColumns* opera sobre cada uma das colunas de matriz de entrada de forma individual, realizando operações XOR e multiplicação de matrizes para gerar novos valores. A Figura 10 exemplifica como a matriz resultante é gerada a partir das operações realizadas.

Figura 10 – Operação MixColumns

$$\begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \begin{vmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{vmatrix} = \begin{vmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{vmatrix}$$

Fonte: (Stallings, 2015).

Cada posição da matriz resultante é a soma dos produtos dos elementos de uma linha e uma coluna. Estas somas e multiplicações utilizam o conceito de Campos de Galois (GF)(Stallings, 2015), especificamente em GF(2^8). A multiplicação de cada coluna da matriz de estado por ser visualizada na Figura 11.

Figura 11 – Exemplo de Multiplicação de Cada Coluna

$$s'_{0,j} = (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{1,j} = s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j}$$
$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j})$$
$$s'_{3,j} = (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j})$$

Fonte: (Stallings, 2015).

2.2.2.4 Estágio AddRoundKey

O estágio *AddRoundKey* é essencialmente uma operação XOR entre cada posição da matriz de estado e a chave da rodada. A Figura 12 exemplifica a operação *AddRoundKey*, onde é possível ver a matriz de entrada à esquerda realizando uma operação XOR com a matriz que contém a chave secreta correspondente a rodada do algoritmo, resultando em uma nova matriz de estado.

Figura 12 – Exemplo da Operação AddRoundKey

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

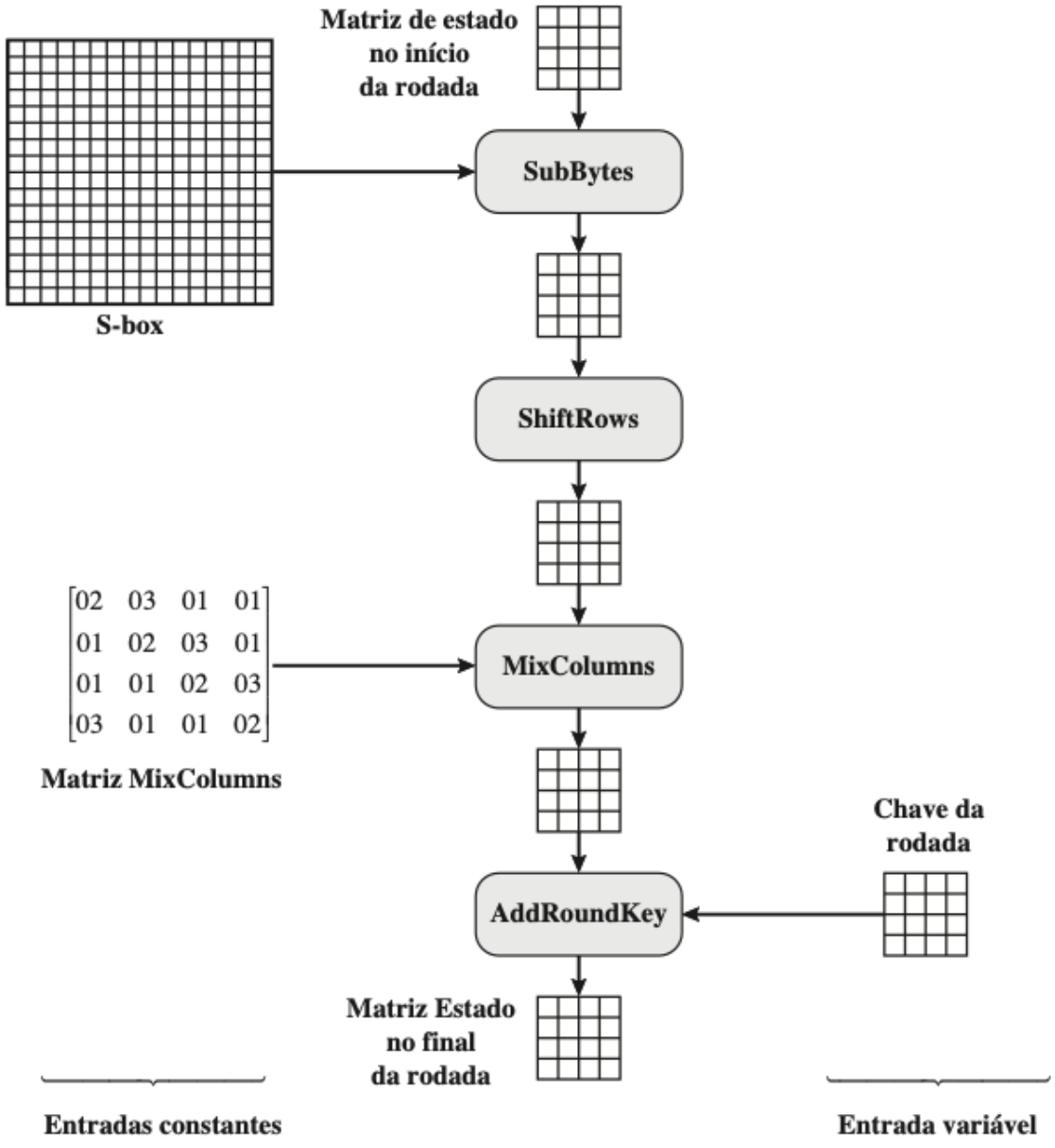
$=$

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

Fonte: (Stallings, 2015).

O algoritmo AES-128, originalmente, é composto por 10 rodadas, onde da 1ª a 9ª rodada são executadas as 4 operação acima descritas exatamente nesta ordem e, por último, na 10ª rodada, são executadas somente 3 destas operações, sendo a operação *MixColumns* não utilizada nesta rodada. A Figura 13 apresenta a visão de uma rodada do algoritmo AES, enfatizando quais entradas, variáveis ou constantes, são utilizadas em cada um dos estágios.

Figura 13 – Visão de uma Rodada do AES



Fonte: (Stallings, 2015).

2.3 Métricas de Avaliação de Segurança

Com o objetivo de verificar a qualidade da encriptação gerada pelo algoritmo proposto, foram utilizadas três análises que serão descritas a seguir.

2.3.1 Efeito Avalanche

O efeito avalanche é uma medida de segurança altamente desejável em criptografia de boa qualidade. Isto significa que uma ligeira alteração no *plaintext* ou na chave de encriptação, mesmo que seja de apenas um bit, deve acarretar em uma grande mudança no *ciphertext* (Majumdar; Biswas; Baishnab; Sood, 2019). Se a mudança afetar pelo menos metade dos bits, o que significa 50% do texto cifrado, então o algoritmo criptográfico é classificado como de boa qualidade (Biswas, 2023).

Para calcular o valor de efeito avalanche que o algoritmo criptográfico produz, é utilizado o conceito de distância de Hamming, no qual a distância consiste no número de posições que divergem entre duas strings de mesmo tamanho (Mohammad; Abdullah, 2022). Por exemplo, as strings carro e cerro teriam distância de Hamming igual a 1, pois divergem seu conteúdo em apenas uma posição. O cálculo de fato do efeito avalanche se dá pela Equação 1.

$$\text{Efeito Avalanche} = \left(\frac{\text{nº de bits alterados no ciphertext}}{\text{nº de bits no ciphertext}}\right) * 100 \quad (1)$$

2.3.2 Critério de Balanceamento

Um teste muito importante para verificar se uma s-box está bem projetada é o critério de balanceamento, que consiste em verificar se a s-box consegue gerar uma quantidade balanceada de 0's e 1's no *ciphertext* (Fadhil; Farhan; Fadhil; Al-saidi, 2020). Em outras palavras, o *ciphertext* deve conter o mais próximo possível de 50% de 0's e 50% de 1's.

2.3.3 NIST Randomness Test

Os testes do NIST consistem num compilado de 14 testes especialmente desenvolvidos para analisar sequências binárias geradas por algoritmos criptográficos. Estes testes examinam a sequência binária utilizando estatísticas de bits ou blocos de bits, no qual, ao final da análise, são capazes de concluir se a sequência binária possuí aleatoriedade ou não (Sýs; Riha; Matyas; Marton; Suciu, 2015).

Para verificar se o *ciphertext* gerado por um algoritmo criptográfico proposto é aleatório o suficiente, é necessário gerar um *ciphertext* a partir de um *plaintext* e

uma chave. Esse *ciphertext* deve ser convertido em uma sequência binária e, então, levado para os testes do NIST de modo a avaliar se a sequência é randômica o suficiente. A saída dos testes do NIST são sempre aleatória ou não aleatória o que significa, na prática, aprovada ou reprovada, respectivamente.

2.4 Revisão Bibliográfica

De modo a investigar o estado de arte na área de criptografia e algoritmos criptográficos leves, foi realizado um mapeamento sistemático da literatura. Este mapeamento buscou apresentar as diferentes estratégias já existente para se obter um LWC. Ao final, foi realizado um comparativo entre as estratégias encontradas e também é destacada qual a contribuição desta dissertação frente aos demais trabalhos encontrados na literatura.

2.4.1 Mapeamento Sistemático da Literatura

Com o objetivo de verificar o estado da arte no tema relacionado a segurança da informação em aplicações IoT, foi realizado um mapeamento sistemático da literatura. Esta técnica é muito utilizada por pesquisadores para obter uma visão geral de área, com o objetivo de encontrar lacunas e possíveis espaços para atuar em determinado tema (Wohlin; Runeson; da Mota Silveira Neto; Engström; do Carmo Machado; de Almeida, 2013).

Inicialmente, foram realizadas buscas em duas diferentes bibliotecas digitais: IEEE Xplore[1] e ACM Digital Library[2]. As palavras-chaves escolhidas foram "internet of things", "iot", "security", "cryptography"e "lightweight". A partir das palavras-chave escolhidas, foi então criada a seguinte string de busca: **("internet of things"OR "iot") AND ("security"OR "cryptography") AND ("lightweight")**.

A busca inicial retornou 2039 artigos na ACM e 1625 artigo na IEEE, totalizando 3664 artigos. Logo após, foi aplicado um filtro por ano nas buscas, para buscar artigos publicados nos últimos cinco anos, ou seja, de 2019 a 2023. Após o primeiro filtro, foram obtidos 1187 artigos na ACM e 1139 artigos na IEEE, totalizando 2326 artigos. A segunda filtragem consistiu em selecionar artigos que trabalhavam com otimizações de algoritmos criptográficos, ficando com um total de 152 artigos. A terceira filtragem consistiu em artigos que tinham implementações e avaliações de algoritmos no contexto de aplicações IoT, o que reduziu para 23 artigos.

A Tabela 1 apresenta a quantidade de artigos selecionados resultando de cada etapa de filtragem realizada.

[1] https://ieeexplore.ieee.org/Xplore/home.jsp
[2] https://dl.acm.org/

Tabela 1 – Número de Artigos em Cada Etapa

Biblioteca	1ª Etapa	2ª Etapa	3ª Etapa	4ª Etapa
ACM	2039	1187	28	3
IEEE	1625	1139	124	20
Total	3664	2326	152	23

Fonte: Elaborada pelo autor.

2.4.2 Resultados do Mapeamento Sistemático da Literatura

O mapeamento realizado buscou responder as seguintes questões de pesquisa:

- **Q1:** Quais são as tendências de pesquisa na área?
- **Q2:** Estão sendo utilizados algoritmos *lightweight*?
- **Q3:** Quais algoritmos criptográficos estão sendo usados para otimizações?
- **Q4:** Estão sendo realizados comparativos de desempenho e/ou consumo energético entre as propostas?
- **Q5:** Quais plataformas estão sendo utilizadas?

A Tabela 2 reúne as informações extraídas dos 23 artigos selecionados no mapeamento. Esta tabela busca responder as questões de pesquisa de forma resumida, trazendo quais algoritmos estão sendo apresentados, quais plataformas estão sendo utilizadas nos testes e também apresentando as avaliações realizadas por cada autor.

Como podemos observar, há muita pesquisa no tema de algoritmos criptográficos leves e existe uma tendência de continuidade da pesquisa por ser um assunto muito importante principalmente na área de Internet das Coisas. Foi possível constatar que há uma grande quantidade de algoritmos sendo desenvolvidos para dispositivos computacionais com recursos limitados, além de se ter estudado várias maneiras de reduzir o consumo energético dos algoritmos de criptografia tradicionais, como o AES. Também foi possível observar que não há apenas estudos em algoritmos criptográficos de chave simétrica, mas também algoritmos de chave assimétrica, vários baseados em curvas elípticas, e algoritmos baseados em funções hash.

As métricas de avaliação mais utilizadas pelos trabalhos foram: atraso(tempo em que um pacote leva para chegar até seu destino), *throughput*, uso de memória, efeito avalanche, segurança da técnica, tamanho do código, consumo energético, tempo de encriptação e decriptação, área utilizada, número de ciclos, tamanho da chave, tamanho do bloco, tempo de geração da chave e tamanho da chave.

Tabela 2 – Comparativo Entre os Trabalhos Apresentados

Trabalho/Ano	**Algoritmos Utilizados**	**Plataformas Utilizadas**	**Avaliações**
Nuha (2022)	Simon e Speck	-	Atraso(ms), *throughput*(Mbps), uso de memória(MB) e efeito avalanche(%)
Buchana (2020)	Diversos LWC separados por categorias	-	Segurança, tamanho do código consumo de RAM e *throughput*
Al-nasser (2019)	PRESENT, RC5, CLEFIA, Mysterion, Grain, MICKEY 2, Trivium PHOTON, Spongent, Quark, Lesamnta Chaskey, SipHash, RSA e ECC	Texas Instruments MSP430F1611	Consumo energético (mJ), tempo de encriptação e decriptação (ms) e tamanho de código (kB)
Hasan (2019)	G-TBSA	Arduino Uno e ESP8266	Efeito avalanche e consumo energético (μJ/byte)
Kumar (2021)	KECCAK, PHOTON, QUARK, SPONGENT, GLUON e HASH-ONE	ASIC 0,13 μm e 0,18 μm	Segurança, área, *throughput,* potência e número de ciclos
Sharath (2020)	A4	-	Segurança
Jianwei (2020)	Novo LWC	MATLAB (software) e Xilinx Virtex-6 (hardware)	Segurança, efeito avalanche e tempo de encriptação
Afzal awan (2020)	Elixir	-	Segurança e *throughput*
Sadio; Ngom; Lishou (2019)	ChaCha20 em união com o Poly1305	Arduino Uno e ESP8266	Tempo de geração da chave, encriptação e decriptação e consumo de memória
Salman; Farhan; Shakir (2022b)	12 diferentes LWCs	-	Tamanho da chave, tamanho do bloco, número de rodadas tempo de encriptação, *throughput* e efeito avalanche
Wahid (2021)	32 diferentes LWC	Raspberry Pi 3B, iMX233 e Raspberry Pi Zero W	Potência, uso de RAM e tempo de execução

Continua na próxima página

Trabalho/Ano	Algoritmos Utilizados	Plataformas Utilizadas	Avaliações
Guo; Li; Liu (2021)	Shadow	Xilinx Virtex-5 e ASIC 0,18 μm	Efeito Avalanche, área utilizada, energia e *throughput*
Al-shargabi; Assi (2022)	LWC baseado em DNA	-	Tempo de encriptação, segurança e efeito avalanche
Dandjinou (2022)	R-LAES	-	Segurança e consumo energético
Ghorashi; Zia; Jiang (2020)	Variação do algoritmo Klein	PC com Windows de 2.4 GHz e Raspberry Pi 3	Uso de CPU, consumo de memória, tempo de execução e quantidade de dados processados
Yelamarthi (2019)	XTEA	PIC18F27K40 e MSP432	Consumo de memória, potência e tempo de execução
Sultan; Mir; Banday (2020)	AES com diferentes modos de operação	Sky e Z1	Consumo energético
Abdallah; Kuang; Huang (2020)	AES em Hardware	Intel Core i5-8250U com Linux v19.04 Intel Core i5-8250U com Win 10, Intel Silver J5005 com Linux v18.10, Intel Silver J5005 com Win 10 e ARM v8 com JetPack 4	Desempenho e consumo energético
Mohammad; Abdullah (2022)	AES com Otimizações	Intel Core i5 com Windows 10	Efeito Avalanche e Desempenho
Fadhil (2020)	AES com Otimizações	-	Efeito Avalanche, Desempenho Criterio de Balanceamento e NIST
Hammod (2022)	AES com Otimizações	-	Desempenho, Efeito Avalanche e NIST
Shi (2019)	AES com Otimizações	MKL36Z64VLH4	Desempenho e Efeito Avalanche
Naif (2019)	AES com Otimizações	Raspberry Pi B	Desempenho e NIST

Fonte: Elaborada pelo autor.

Em questão das plataformas utilizadas nos testes, alguns trabalhos não apresentaram esta informação, não sendo possível saber se foram simulações ou simplesmente omitiram esta informação. Porém, apareceram muitas implementações em ASIC, Arduino Uno, diferentes modelos de Xilinx e diferentes modelos de Raspberry. Também em alguns trabalhos foi utilizado o módulo Wi-Fi ESP8266.

2.4.3 Trabalhos Relacionados

Considerando que esta dissertação possui como objetivo implementar otimizações no algoritmo AES, foram selecionados os 5 artigos(dos 23 artigos da última etapa de revisão da literatura), que se baseiam no AES, como trabalhos relacionados. Apesar de haver 6 artigos na revisão da literatura que utilizam o AES, um deles foi excluído por apresentar otimizações em hardware, sendo assim, os 5 artigos selecionados trabalham apenas com a versão em software do AES.

Com o objetivo de reduzir tempo de execução, Mohammad; Abdullah (2022) propuseram uma versão otimizada do algoritmo AES. Para isto, foi proposta a substituição do estágio *MixColumns* pelo uso de um modelo matemático de frações contínuas, que consiste em uma maneira de representar um número inteiro somado a uma série de funções aninhadas. Esta versão proposta utiliza o mesmo tamanho de bloco e chave do algoritmo original, além de utilizar o mesmo número de rodadas. O algoritmo proposto passa por 10 rodadas contendo as operações *SubBytes*, *ShiftRows* e *AddRoundKey* e, somente após as 10 rodadas, o algoritmo entra no último estágio que consiste na aplicação do método de frações contínuas. Para uma entrada de 20 bytes, o algoritmo atingiu uma redução de 41,38% em tempo de execução. Em questões de segurança, foi verificado o percentual de efeito avalanche alcançado, atingindo valores esperados para esta métrica.

Uma versão *lightweight* do algoritmo AES foi proposta por Fadhil; Farhan; Fadhil; Al-saidi (2020), no qual todas as etapas do algoritmo são dependentes entre si utilizando sistemas caóticos. A teoria do caos é focada no comportamento dinâmico de sistema, no qual têm alta sensibilidade a condições iniciais, ou seja, uma pequena mudança nos valores de entrada deve acarretar numa grande mudança nos valores de saída. Neste algoritmo é utilizado um sistema caótico 3D para geração de chaves e também para gerar os valores dinâmicos de deslocamentos utilizados no estágio *ShiftRows*. Já para geração da s-box é utilizado um sistema caótico 1D. Nas etapas de permutação, um sistema caótico 2D é utilizado. Como resultados, são apresentados os valores de efeito avalanche obtidos, critério de balanceamento, teste do NIST e tempos de encriptação. Foi concluído que o algoritmo proposto é mais rápido que a versão original do AES, além de atender as questões de segurança necessárias.

Hammod (2022) propôs otimizações em dois estágios do AES, *SubBytes* e *Shif-*

tRows, com o intuito de torná-lo mais leve. Além disto, Hammod (2022) utiliza apenas 4 rodadas nos processos de encriptação e decriptação. A modificação proposta para o estágio *SubBytes* consiste no uso de uma s-box menor, contendo 16 bytes. Esta versão utiliza duas s-boxes de 16 bytes cada, uma para encriptar e outra para decriptar o dado. Já a modificação proposta para o estágio *ShiftRows* consiste no uso de um sistema caótico 1D para gerar a quantidade de deslocamentos de cada linha da matriz de entrada. O artigo traz como resultados a aprovação nos testes do NIST, tempos de encriptação, consumo de memória e percentuais de efeito avalanche. Foi concluído pelo autor que foi atingida uma redução de 10% em tempo de execução comparado ao algoritmo original.

Com o intuito de reduzir a complexidade do algoritmo AES, Shi; Wang; Jia; Peng; Jiang; Zhu (2019) propuseram uma versão leve chamada LCHAOSAES. Esta versão utilizada 7 rodadas e utiliza tabelas pré-calculadas para acelerar a execução dos estágios. Como resultados, foi verificado o percentual de efeito avalanche calculado tempos de execução. A versão proposta atingiu melhores valores de tempo de execução e valores esperados de efeito avalanche, porém, não foi verificado critério de balanceamento nem executado o teste do NIST.

Naif; Abdul-majeed; Farhan (2019) propuseram uma versão leve do algoritmo AES, utilizando um sistema caótico 5D, que combina os mapas caóticos Logistic e Lorenz, com o objetivo de reduzir tempo de execução. Esta versão proposta utiliza as mesmas funções do AES original, com exceção do estágio *MixColumns*, que é substituído por estágios multi-XOR. Foram avaliadas versões do algoritmo com 4, 6 e 8 rodadas. Como resultados, foram extraídas métricas de tempo de execução e, em questões de segurança, foi utilizado o teste do NIST. Foi concluído que a versão proposta é segura e atraente para aplicações IoT.

A Tabela 3 apresenta um resumo das principais estratégias utilizadas pelos demais autores. Desta revisão bibliográfica, foi possível constatar que o algoritmo AES é bastante utilizado para gerar novos algoritmos leves a partir de suas funções.

Os trabalhos de Fadhil; Farhan; Fadhil; Al-saidi (2020), Hammod (2022) e Naif; Abdul-majeed; Farhan (2019) têm em comum a utilização de sistemas caóticos em estágios do AES. Já o trabalho de Mohammad; Abdullah (2022) optou por substituir o estágio *MixColumns* pela utilização de um método de frações contínuas. Por fim, o trabalho de Shi; Wang; Jia; Peng; Jiang; Zhu (2019) utilizou tabelas pré-calculadas para acelerar a execução do algoritmo.

Os cinco trabalhos apresentados obtiveram reduções de algum tipo de métrica, porém, todos trabalhos implementaram apenas um tipo de otimização, se preocupando em otimizar apenas um tipo de métrica. Além disto, nenhum dos cinco trabalhos apresentados realizou uma análise de consumo energético, que é uma das principais preocupações quando se está preocupado com aplicações IoT.

Tabela 3 – Resumo das Estratégias Propostas pelos Trabalhos Relacionados

Artigo	**Estratégia Utilizada**
Mohammad; Abdullah (2022)	Substitui o *MixColumns* por Frações Contínuas
Fadhil(2020)	Faz uso de sistemas caóticos 1D, 2D e 3D em todos os estágios do AES
Hammod (2022)	Utiliza s-box de 32 bytes e um sistema caótico 1D no estágio *ShiftRows*
Shi(2019)	Utiliza tabelas pré-calculadas que dão suporte a operação de *MixColumns*
Naif(2019)	Utiliza sistemas caóticos Logistic e Lorenz em todos os estágios do AES
Este Trabalho	Utiliza somas e multiplicações mais simples no estágio *MixColumns*, além de utilizar uma s-box de apenas 16 bytes

Fonte: Elaborada pelo autor.

Com isto, esta dissertação se diferencia dos demais trabalhos justamente por implementar dois tipos de otimizações, uma delas levando a redução de tempo de execução e outra levando a redução de consumo de memória. Além disto, esta dissertação também avaliou a redução de consumo energético que foi obtida a partir das otimizações implementadas.

2.5 Considerações Finais Sobre o Capítulo

Este capítulo teve como objetivo trazer uma introdução aos conceitos utilizados ao longo desta dissertação. Inicialmente apresentou o conceito de Internet das Coisas, seguido pelo conceito geral de criptografia, criptografia leve e abordando o algoritmo AES, apresentando sua estrutura, de modo a facilitar a explicação de quais estágios dele serão implementadas as otimizações propostas. Também foram apresentadas as métricas de segurança que serão utilizadas no capítulo de resultados desta dissertação, de modo a facilitar a compreensão dos resultados obtidos. Além disto, ao final do capítulo, foi apresentada a revisão bibliográfica realizada, apresentando os demais artigos que têm similaridade ao tema e que serão comparados com os resultados desta dissertação.

3 DESENVOLVIMENTO

Este capítulo apresenta as otimizações propostas ao algoritmo AES de modo a torná-lo um LWC. Inicialmente apresenta a análise inicial realizada no algoritmo AES, seguido pelas otimizações propostas nos estágios *SubBytes* e *MixColumns* e, por fim, apresenta como foram realizadas as medições de potência e energia.

3.1 Análise Inicial do Algoritmo AES

Com o intuito de verificar quais estágios do AES eram mais custosos em termos de tempo de execução e consumo de memória, foi realizada uma análise inicial no algoritmo para encontrar possíveis otimizações que poderiam ser implementadas. Esta análise inicial foi conduzida em um MacBook Air com processador Apple M2, 8GB de memória RAM, sistema operacional macOS Ventura 13.4.1 e SSD de 256GB. A versão original utilizada do AES foi disponibilizada no GitHub e foi desenvolvida por Meysam Parvizi[1]. Esta versão foi selecionada por estar muito bem documentada e seguindo claramente a implementação original do algoritmo AES, de acordo com sua descrição encontrada na literatura.

Em termos de tempo de execução, a análise inicial consistiu na execução de diversos testes para verificar qual estágio dominava o tempo total de execução. Para isto, foram utilizados diferentes tamanhos de entrada, todas do tipo string, e verificado o comportamento do algoritmo, especificamente da função de encriptação. Nestes testes, o algoritmo foi executado 100 vezes em sequência e, após, foi extraída a média aritmética destes tempos de execução. O algoritmo AES foi compilado com GCC sem qualquer tipo de *flag* de otimização ativada e executado diretamente via terminal. Para verificar os tempos de execução consumidos por cada estágio, foi utilizada a função clock() antes e depois da chamada de cada função. A Tabela 4 apresenta os resultados da análise inicial, mostrando os tempos que cada estágio consome e seus respectivos percentuais.

[1] Disponível em: https://github.com/m3y54m/aes-in-c/tree/main

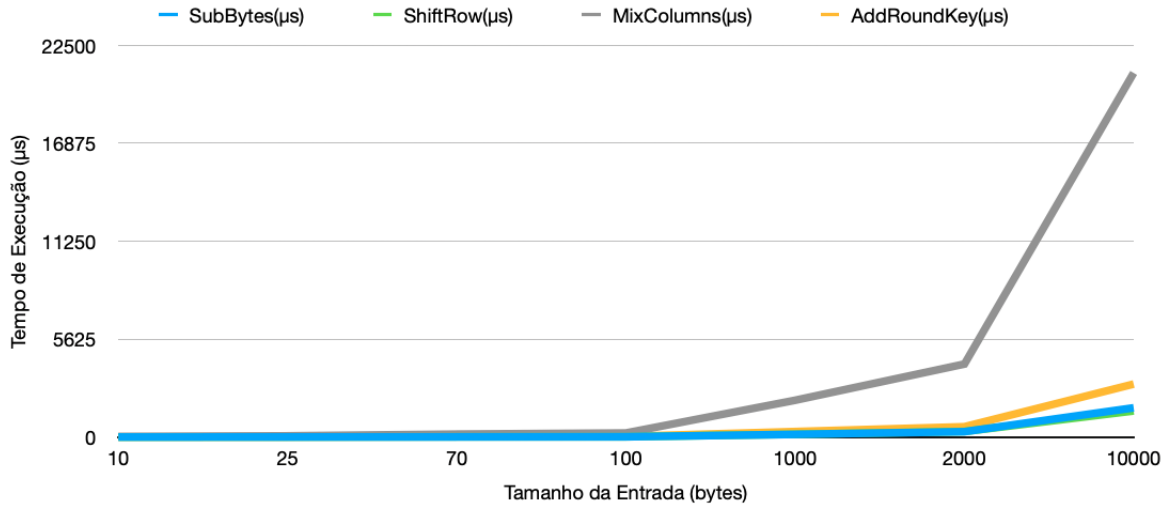
Tabela 4 – Tempos de Execução(µs) do AES Original - Função de Encriptação.

Tamanho da Entrada(bytes)	SubBytes (µs)	%	ShiftRow (µs)	%	MixColumns (µs)	%	AddRoundKey (µs)	%	Tempo Total (µs)
10	4,03	8,02%	3,87	7,70%	34,92	**69,48%**	7,44	14,80%	50,26
25	6,29	6,51%	6,25	6,47%	72,15	**74,66%**	11,95	12,37%	96,64
70	13,3	5,89%	14,24	6,30%	171,61	**75,95%**	26,79	11,86%	225,94
100	18,54	5,98%	18,14	5,85%	237,36	**76,51%**	36,19	11,67%	310,23
1000	157,45	5,71%	160,42	5,82%	2117,90	**76,87%**	319,54	11,60%	2755,31
2000	311,55	5,74%	322,01	5,93%	4197,33	**77,32%**	597,34	11,00%	5428,23
10000	1683,24	6,20%	1503,92	5,54%	20916,06	**77,04%**	3046,92	11,22%	27150,14

Fonte: Elaborada pelo autor.

Como é possível perceber, o tempo total da função de encriptação do AES é grande parte consumido pelo estágio *MixColumns*, consumindo de 69,48% até 77,04% do tempo total de execução para entradas de 10 bytes e 10000 bytes, respectivamente. A Figura 14 demonstra o crescimento dos tempos de execução.

Figura 14 – Crescimento dos Tempos de Execução de Cada Estágio



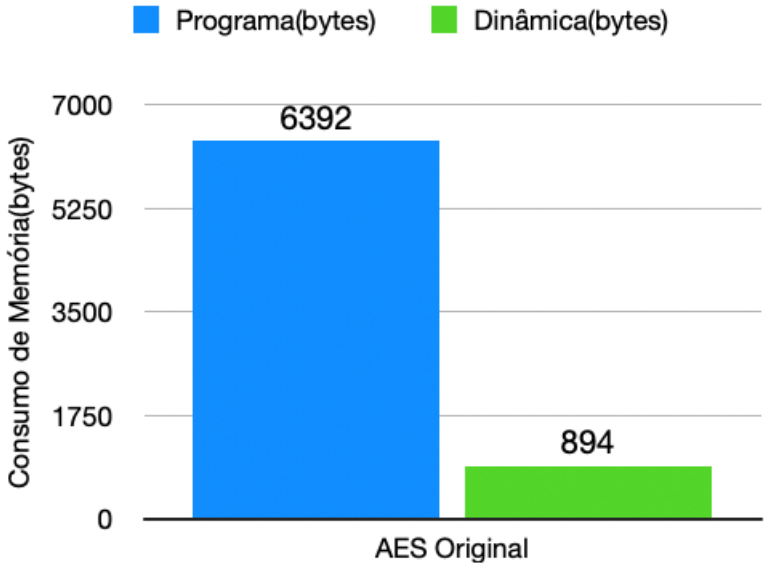
Fonte: Elaborada pelo autor.

Visto que o estágio *MixColumns* representa, em média, 75,4% do tempo total de execução da função de encriptação, buscar oportunidades de otimização neste estágio do AES seria muito benéfico para reduzir o tempo de execução do algoritmo.

Já em termos de consumo de memória, foi verificado que o algoritmo AES utiliza duas s-box de 256 bytes cada, totalizando 512 bytes de armazenamento em memória dinâmica. Logo, diminuir o tamanho destas s-box, que são diretamente relacionadas ao estágio *SubBytes*, seria uma interessante estratégia para reduzir o consumo de memória do algoritmo, o que é muito interessante para aplicações IoT, onde normalmente são implementadas em dispositivos com restrições de memória. Com o intuito de verificar quantos o tamanho em bytes de memória, tanto dinâmica quanto de programa, que o algoritmo AES utilizou, foi utilizada a IDE do Arduino para extrair esta

informação, para, após a implementação das otimizações propostas, poder comparar qual foi o percentual de redução alcançado. A Figura 15 apresenta o consumo de memória do algoritmo AES.

Figura 15 – Consumo de Memória do Algoritmo AES



Fonte: Elaborada pelo autor.

3.2 Otimizações no Estágio SubBytes

A otimização proposta para o estágio de *SubBytes* baseia-se na utilização de uma s-box menor, contendo apenas 16 entradas. Esta s-box é modelada como um vetor com índice de 0 a 15 e seu conteúdo indo de F a 0. A Figura 16 exemplifica a ideia da tabela desenvolvida.

Figura 16 – S-box Desenvolvida com 16 Bytes

Índice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Valor	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Fonte: Elaborada pelo autor.

A matriz de estado é composta por 16 entradas, dispostas entre as 4 linhas e as 4 colunas da matriz. Cada byte, representado em hexadecimal, é dividido em dois dígitos, onde estes dois dígitos são utilizados como índices da s-box proposta. Quando há um dígito de A a F, em hexadecimal, este é utilizado como decimal, variando de 10 a 15, respectivamente. A Figura 17 exemplifica o processo de aplicação da operação

SubBytes, utilizando a nova s-box proposta, mostrando a matriz de estado de entrada e, posteriormente, a matriz com a informação atualizada após a etapa de substituição. Utilizando este método, não há necessidade de uma s-box para encriptação e uma s-box para decriptação(s-box reversa), pois a mesma tabela é utilizada para ambas operações.

Figura 17 – Exemplo do Estágio *SubBytes* Utilizando a Nova S-box

Matriz de Entrada			
66	c4	64	bd
63	32	09	eb
0e	c4	80	e9
fa	56	00	a7

SubBytes

Matriz pós SubBytes			
99	3b	9b	42
9c	cd	f6	14
f1	3b	7f	16
05	a9	ff	58

Fonte: Elaborada pelo autor.

Utilizando o hexadecimal 3B como exemplo, ao passar pelo estágio *SubBytes*, ele se tornaria C4 e, ao passar pelo processo de decriptação, utilizando a mesma s-box, ele voltaria ao seu valor 3B original.

O algoritmo AES original armazena duas s-boxes de 256 bytes cada, uma para o processo de encriptação e outra para o processo de decriptação, totalizando 512 bytes de consumo em armazenamento. Já a alternativa proposta utiliza apenas 16 bytes, pois utiliza uma única tabela tanto para o processo de cifragem quanto para o processo de decifragem, resultando numa redução de 96,87% em bytes.

3.3 Otimizações no Estágio MixColumns

A otimização proposta para o estágio *MixColumns* se baseia na utilização de operações mais simples e menos custosas em termos de recursos computacionais. Para isto, foi desenvolvida uma função que realiza multiplicações em Campos da Galois pela constante 2, exemplificado pelo Algoritmo 3.1, que serve de base para todas as outras multiplicações, ou seja, todas as outras multiplicações são feitas utilizando esta função e utilizando funções XOR para somas. As multiplicações são sempre pelas constantes 2 e 3, no processo de encriptação, e pelas constantes 9, 11, 13 e 14 no processo de decriptação.

```
unsigned char gmult2(unsigned char value){
    if(value & 0x80)
        return (value << 1) ^ 0x1B;
    else
        return value << 1;
}
```

Algoritmo 3.1 – Função que Realiza Multiplicação por 2

O Algoritmo 3.2 apresenta a operação *MixColumns* proposta, onde originalmente era um laço de repetição, agora são expandidas as operações de soma e multiplicação, que vão compondo as novas colunas da matriz de estado. Na versão original do AES, é criado um vetor temporário de quatro posições, onde vão sendo performadas as operações do estágio *MixColumns* e, após, este vetor temporário é copiado de volta para as posições correspondentes da matriz de estado. Já na versão otimizada, ao invés de criar um vetor temporário, já é criada uma matriz 4x4 temporária que realizará todas as operações de *MixColumns* nela para então, após, copiar os valores atualizados para a matriz de estado. Sendo assim, ao invés de 4 chamadas de função são utilizadas apenas 1. Estão sendo apresentadas as operações envolvendo as colunas 0 e 3, estando suprimido o código das colunas 1 e 2.

```
void MixColumnsLite(unsigned char *state){
    unsigned char tmp[4][4];
    tmp[0][0] = (unsigned char)(gmul2(state[0]) ^ gmul3(state[4]) ^ state[8] ^ state[12]);
    tmp[1][0] = (unsigned char)(state[0] ^ gmul2(state[4]) ^ gmul3(state[8]) ^ state[12]);
    tmp[2][0] = (unsigned char)(state[0] ^ state[4] ^ gmul2(state[8]) ^ gmul3(state[12]));
    tmp[3][0] = (unsigned char)(gmul3(state[0]) ^ state[4] ^ state[8] ^ gmul2(state[12]));

                                                .
                                                .
                                                .

    tmp[0][3] = (unsigned char)(gmul2(state[3]) ^ gmul3(state[7]) ^ state[11] ^ state[15]);
    tmp[1][3] = (unsigned char)(state[3] ^ gmul2(state[7]) ^ gmul3(state[11]) ^ state[15]);
    tmp[2][3] = (unsigned char)(state[3] ^ state[7] ^ gmul2(state[11]) ^ gmul3(state[15]));
    tmp[3][3] = (unsigned char)(gmul3(state[3]) ^ state[7] ^ state[11] ^ gmul2(state[15]));
}
```

Algoritmo 3.2 – Função que Realiza a Operação *MixColumns*

Esta otimização proposta para o estágio *MixColumns* apresentou bons ganhos em termos de tempo de execução, tornando o algoritmo AES mais rápido comparado a sua versão original, atingindo uma redução média de 86,71% em tempo de execução.

3.4 Plataformas Embarcadas Utilizadas

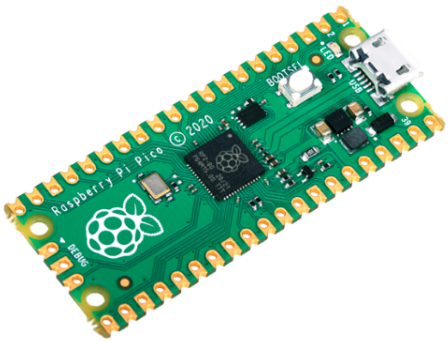
Foram conduzidos testes em duas plataformas muito utilizadas no desenvolvimento de aplicações IoT, que são a ESP-WROOM-32 Devkit V1, desenvolvida por Espressif (Systems, 2024) e a Raspberry Pi Pico, desenvolvida por Raspberry Pi (Pi, 2024). Ambas plataformas estão sendo apresentadas nas Figuras 18 e 19, respectivamente.

A ESP-WROOOM-32 é uma plataforma de baixo custo para desenvolver aplicações IoT(Hercog; Lerher; Truntič; Težak, 2023), contendo um processador Xtensa Dual-Core de 32 de bits, 520kB de memória RAM e 4MB de memória flash, podendo atingir uma frequência máxima de 240MHz.

Figura 18 – Plataforma ESP-WROOM-32

Fonte: (Systems, 2024).

Figura 19 – Plataforma Raspberry Pi Pico



Fonte: (Pi, 2024).

A Raspberry Pi Pico é uma plataforma utilizada em vários projetos IoT, na qual é equipada com um processador de baixo custo bastante atraente(Loker, 2023). Ela é um microcontrolador RP2040 desenvolvido pela própria Raspberry Pi e conta com um processador Arm Cortex-M0+, que atinge uma frequência máxima de 133MHz. Possuí 264kB de memória RAM e 2MB de memória flash.

Ambas plataformas são utilizadas nos testes executados nesta dissertação, com o objetivo de avaliar o tempo de execução necessário para encriptação de dados, tanto na versão original do AES quanto na versão otimizada, além de analisar seus respectivos consumos energéticos.

3.5 Medições de Potência e Energia

Grande parte das aplicações IoT são alimentadas por baterias, logo, a preocupação com consumo energético é uma questão muito importante. Com o objetivo de comparar os consumos energéticos que as versões original e otimizada do AES apresentam, foram realizados experimentos tanto com a ESP-WROOOM-32 quanto com a Raspberry Pi Pico. Para realizar as medições de consumo energético, foi utilizada a Power Profiler Kit II(PPK2), que é uma ferramenta para medições de corrente elétrica muito utilizada em análises de consumo energético em soluções embarcadas, devido a sua alta taxa de amostragem (Semiconductor, 2023). A Figura 20 apresenta a PPK2.

Figura 20 – Power Profiler Kit II(PPK2)

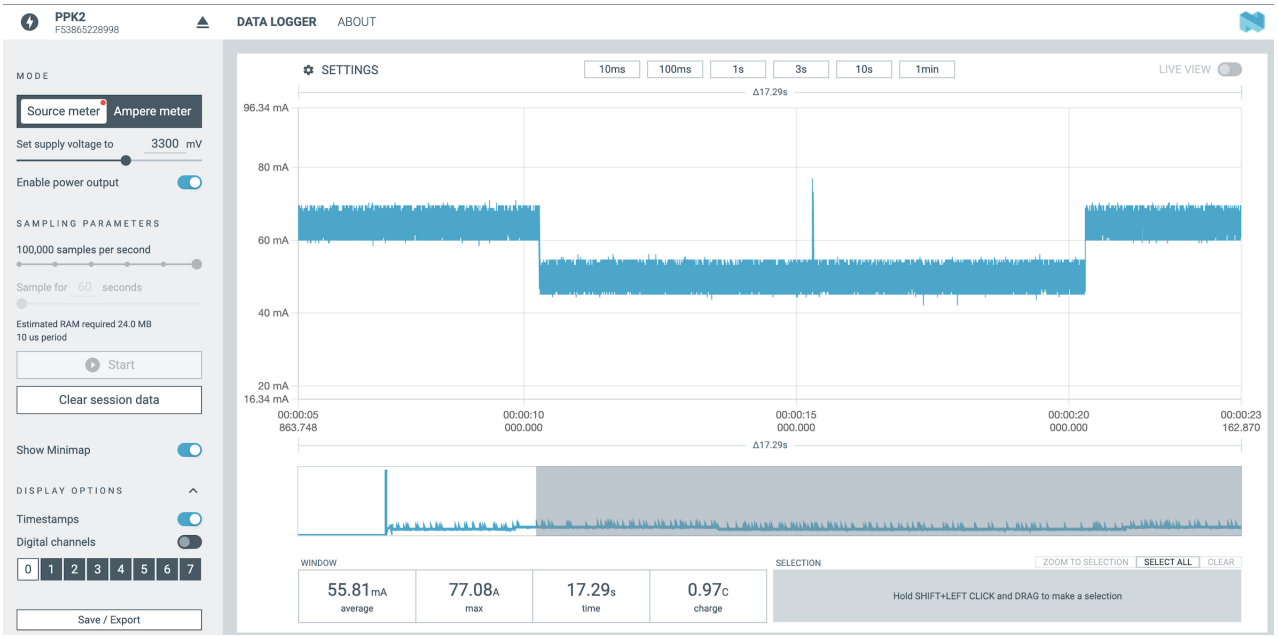
Fonte: (Semiconductor, 2023).

A PPK2 possuí uma faixa de medição de corrente elétrica que varia de 200nA até 1A, com uma resolução de 100nA. No modo fonte, é possível configurar a saída de tensão para valores entre 0,8V e 5V. A PPK2 possuí uma taxa de amostragem de 100k amostras por segundo, ou seja, é possível obter uma amostra de corrente a cada 10us.

A PPK2 possui interface USB, onde se faz a conexão da mesma com o software fornecido pelo próprio fabricante. Neste software é possível realizar configurações de

parâmetros e também visualizar as amostras em tempo real durante a amostragem. A PPK2 pode ser utilizada de dois modos, que são modo fonte e modo amperímetro. Para os experimentos desta dissertação, a PPK2 foi configurada em modo fonte, pois utilizaremos as plataformas ESP-WROOOM-32 e Raspberry Pi Pico ligadas em série com a PPK2, de modo a extrair amostras de corrente elétrica. A Figura 21 apresenta a interface do software utilizado.

Figura 21 – Interface Fornecida pelo Fabricante da PPK2

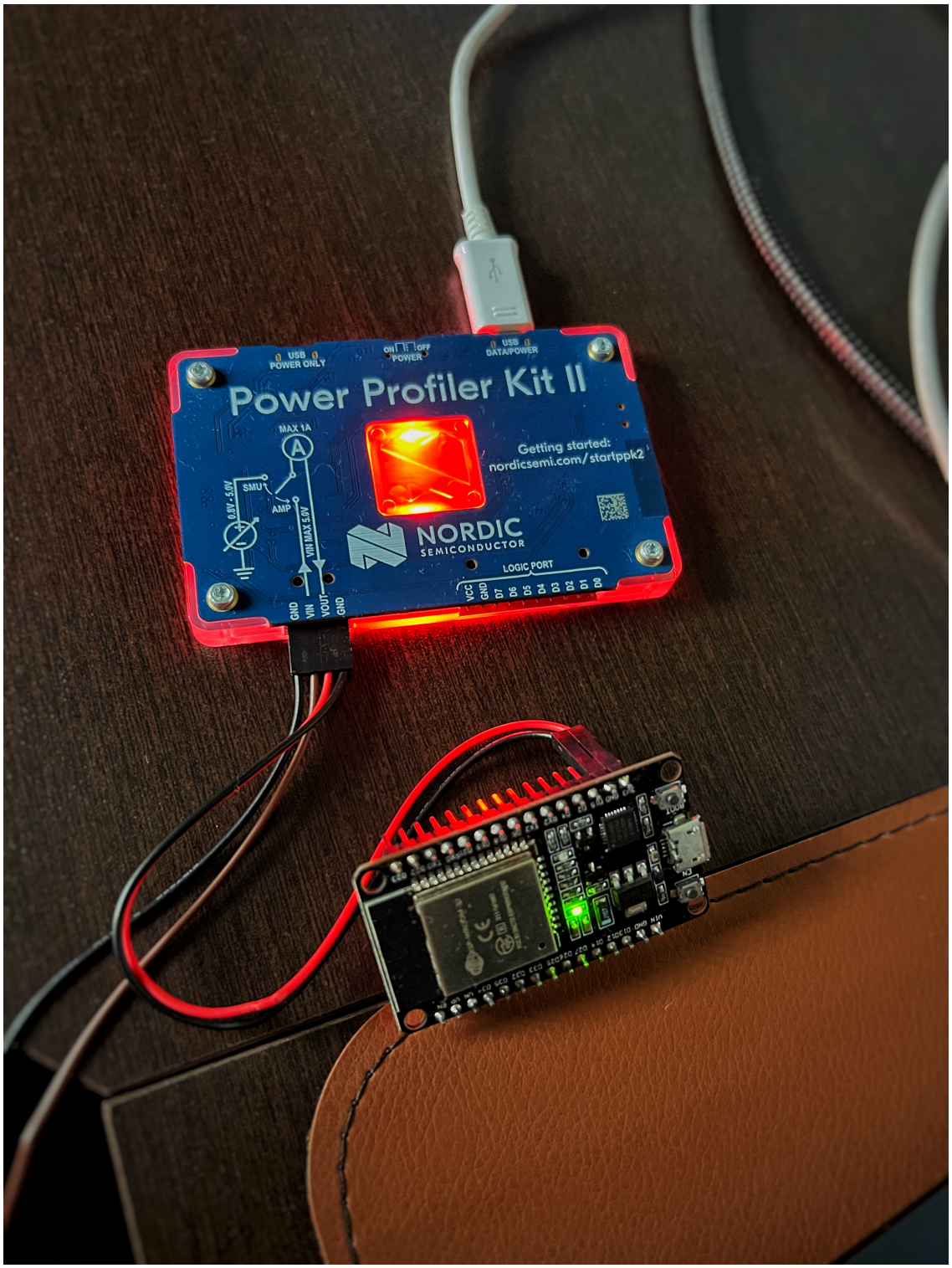


Fonte: (Semiconductor, 2023).

Como é possível verificar na esquerda da Figura 21, existe uma interface com o usuário onde é possível configurar a tensão de saída por meio de um menu de configurações, assim como ativar ou desativar a saída de tensão e também configurar a taxa de amostragem. Na direita da Figura 21 é possível ver em tempo real as amostras de corrente que vão sendo adquiridas ao longo do tempo. Nesta interface, é possível extrair o conjunto de amostras realizadas em um arquivo no formato CSV (do inglês, *Comma-Separated Values*). Este arquivo CSV extraído é composto por duas colunas, onde em uma delas há o *timestamp* da amostra e na outra coluna o valor da amostra em uA.

Para os experimentos realizados, ambas plataformas utilizadas nos testes foram ligadas em série com a PPK2 para que fosse possível coletar as amostras de corrente elétrica. A saída de tensão foi configurada em 3,3V e a taxa de amostragem em 100k amostras por segundo. A Figura 22 apresenta uma imagem real protótipo utilizado, sendo possível visualizar a ESP-WROOOM-32 ligada em série com a PPK2.

Figura 22 – Protótipo Utilizado Para Aquisição de Amostras de Corrente



Fonte: Elaborada pelo autor.

A saída de tensão de PPK2 (VOUT) foi ligada na entrada de tensão da ESP-WROOOM-32, na sua entrada de 3,3V que foi a tensão utilizada nos experimentos. Também o GND da PPK2 foi ligado ao GND da ESP-WROOOM-32. Do mesmo modo, a Raspberry Pi Pico foi posteriormente utilizada neste protótipo para extrair seu conjunto de amostras. Para isto, a Raspberry Pi Pico foi ligada também em série com a PPK2 utilizando a mesma configuração de alimentação utilizada para a ESP-WROOOM-32.

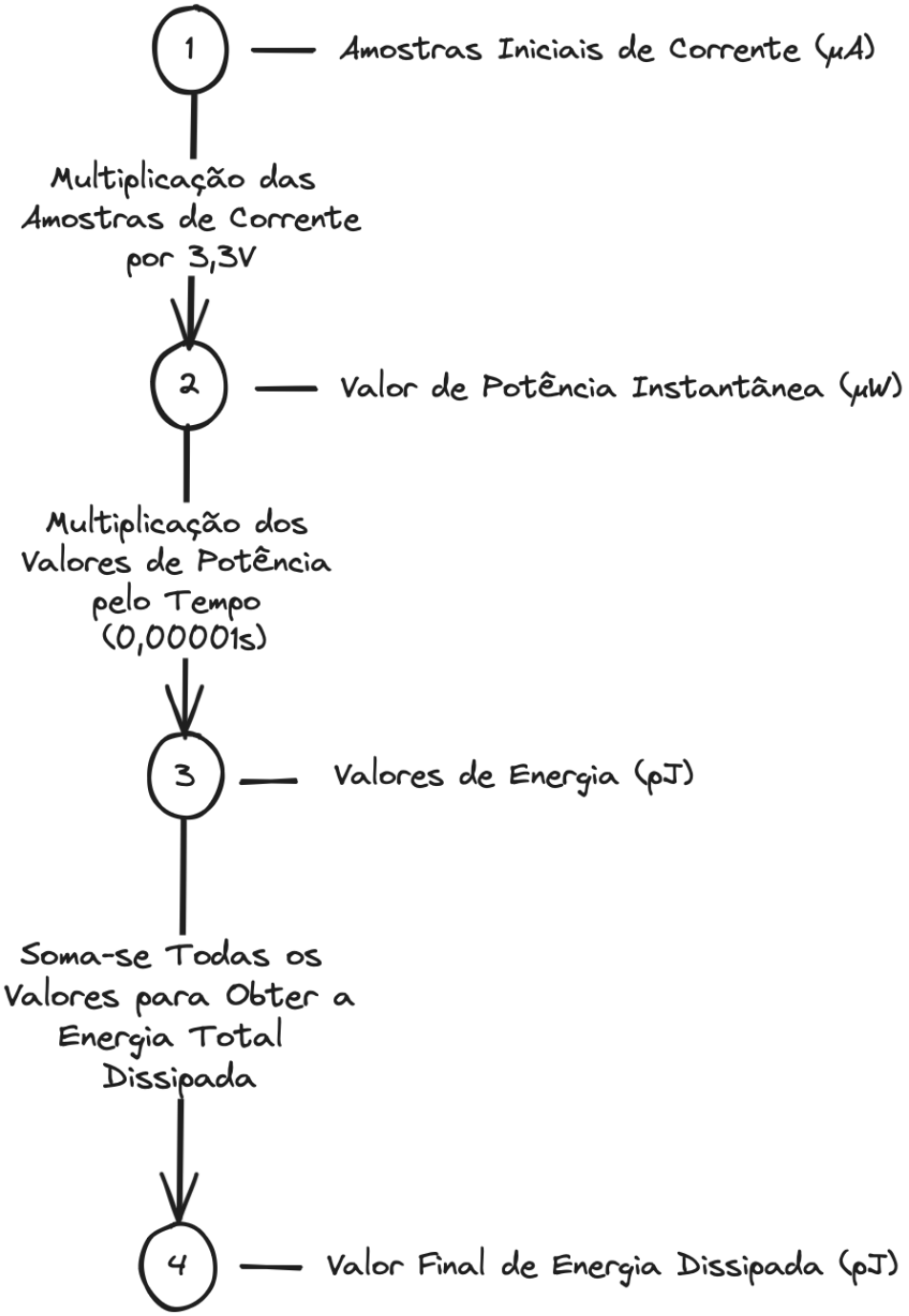
Os experimentos realizados foram:

- Executar um laço com 100 repetições do algoritmo AES original na ESP-WROOOM-32
- Executar um laço com 100 repetições do algoritmo AES otimizado na ESP-WROOOM-32
- Executar um laço com 100 repetições do algoritmo AES original na Raspberry Pi Pico
- Executar um laço com 100 repetições do algoritmo AES otimizado na Raspberry Pi Pico

Para cada um dos experimentos realizados, foi extraído o arquivo CSV com suas respectivas amostras de corrente. Como o arquivo CSV gerado contém amostras de corrente elétrica, foi preciso aplicar uma equação matemática para converter as amostras de corrente elétrica em energia dissipada. De acordo com a equação da potência, $P = V * i$, temos que a potência(W) é calculada a partir da tensão(V) multiplicada pela corrente(A). Logo, para ajustar o arquivo CSV de modo que contenha valores de potência instantânea, foi preciso multiplicar os valores das amostras de corrente pela tensão utilizada, que foi 3,3V. Após, para chegar no valor de fato de energia dissipada, pela equação da energia, $E = P * t$, onde a energia(J) é calculada pela potência(W) multiplicada pelo tempo(s), foi preciso multiplicar os valores de potência instantânea pelo intervalo de tempo de cada amostra, definido em 0,00001s(10us). Por fim, para chegar no valor da energia total dissipada, foi necessário somar todos os valores do arquivo CSV que, após os cálculos acima descritos, são valores de energia instantânea, em Joules.

O fluxograma exibido na Figura 23 busca detalhar a sequência de cálculos realizada para conversão das amostras de corrente obtidas a partir do arquivo CSV gerado pela PPK até encontrar a energia total dissipada. Este fluxograma é aplicado em cada um dos experimentos realizados.

Figura 23 – Fluxo de Cálculos Realizados no Arquivo CSV



Fonte: Elaborada pelo autor.

3.6 Considerações Finais Sobre o Capítulo

Este capítulo teve como foco apresentar as otimizações propostas ao algoritmo AES a fim de torná-lo um algoritmo mais adequado aos requisitos de aplicações IoT. Inicialmente é apresentado a análise inicial realizada no algoritmo criptográfico AES, de modo a encontrar possibilidades de otimizações. Logo após, são então apresentadas as otimizações implementadas no AES, especificamente nos estágios *SubBytes* e *MixColumns*, com o objetivo de reduzir tempo de execução e consumo de memória, respectivamente. Também, ao final do capítulo, foram apresentadas as plataformas utilizadas nos testes de avaliação, que são a ESP-WROOOM-32 e a Raspberry Pi Pico. Por último, com o objetivo de avaliar a potência dissipada e consumo energético, foi apresentada a ferramenta Power Profiler Kit II(PPK2), sua interface de configuração e monitoramento, assim como a metodologia para extração e cálculos realizados nos dados obtidos.

4 RESULTADOS

As otimizações implementadas foram avaliadas em termos de desempenho, utilizando como métricas o tempo de encriptação e o consumo de memória, e em termos de segurança, avaliando efeito avalanche e critério de balanceamento, além de serem submetidas aos testes do NIST. Estes resultados iniciais foram conduzidos em um MacBook Air com processador Apple M2, 8GB de memória RAM, sistema operacional macOS Ventura 13.4.1 e SSD de 256GB, além do auxílio da IDE do Arduino na versão 2.1.0 e compilador GCC, executado diretamente via terminal. Após, são apresentados os resultados obtidos para a execução na ESP-WROOOM-32 e na Raspberry Pi Pico, juntamente dos resultados de consumo energético em ambas as plataformas. Por fim, são comparados os resultados obtidos com outros 5 trabalhos encontrados na literatura que possuem temas afins a este.

4.1 Análise de Desempenho

A análise de desempenho consistiu em extrair métricas de tempo de execução e consumo de memória de ambas as versões do algoritmo AES avaliadas no decorrer desta dissertação, com o intuito de verificar quais os percentuais de redução foram obtidos.

4.1.1 Tempo de Execução

Para obtenção dos valores dos tempos de execução da função de encriptação, cada teste contendo um tamanho diferente de entrada foi executado no Macbook Air em um laço de repetição com 1000 execuções. Ao final de cada teste, foram extraídas as médias aritméticas simples destes valores. Como dados de entrada do algoritmo foram utilizadas strings com os seguintes tamanhos: 10, 25, 70, 100, 1000, 2000 e 10000 bytes. A Tabela 5 traz o comparativo dos tempos de execução do algoritmo AES originalmente proposto (AES Original) e do algoritmo AES com as otimizações propostas neste trabalho (AES Modificado).

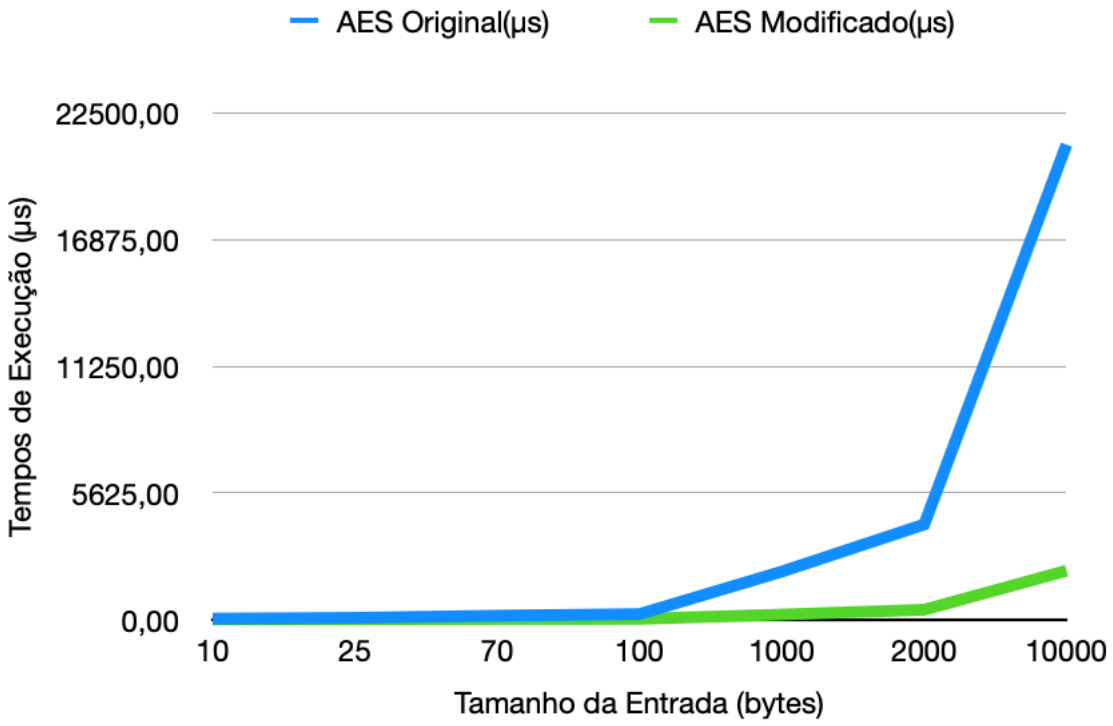
Tabela 5 – Tempos de Encriptação do AES Original e Modificado

Entrada(bytes)	AES Original(µs)	AES Modif.(µs)	Redução(%)
10	40,96	8,23	79,92
25	74,60	12,92	82,68
70	172,95	21,96	87,30
100	243,57	29,25	87,99
1000	2130,79	220,75	89,64
2000	4222,14	434,43	89,71
10000	21091,54	2167,31	89,72

Fonte: Elaborada pelo autor.

Pode-se observar uma redução significativa em termos de tempo de encriptação, atingindo uma redução média de 86,71% com o uso das otimizações propostas. Estas reduções se devem principalmente ao fato da versão otimizada proposta por esta dissertação utilizar operações de somas e multiplicações mais leves no estágio *MixColumns*, acarretando em uma execução bem mais rápida do algoritmo. Na Figura 24 é possível verificar a diferença de crescimento das curvas.

Figura 24 – Tempos de Execução do AES Original e Otimizado

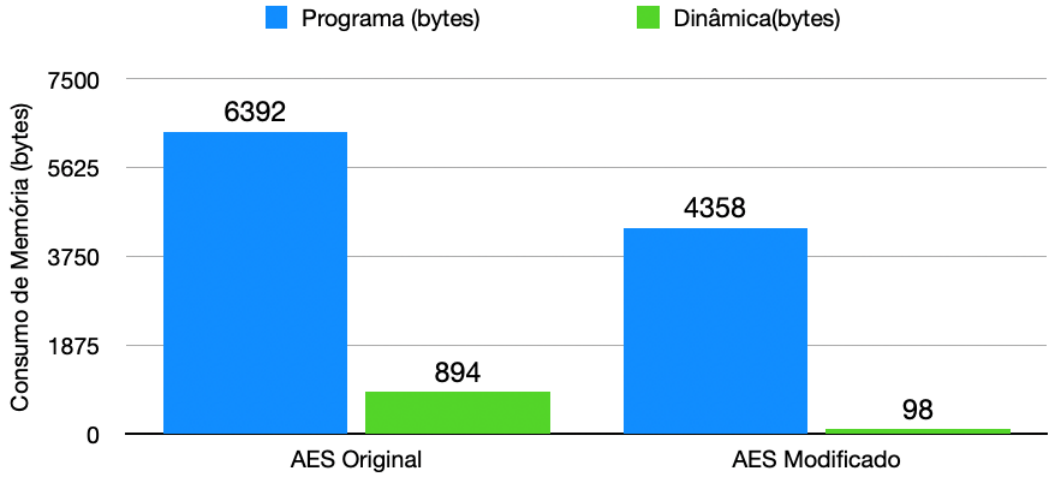


Fonte: Elaborada pelo autor.

4.1.2 Consumo de Memória

Para extração destes resultados, foram compiladas ambas versões dos algoritmos na IDE do Arduino utilizando o Macbook Air. A Figura 25 apresenta o consumo de memória, em bytes, das duas versões do algoritmo, tanto da memória dinâmica quanto da memória de programas. A memória dinâmica(dados) é onde são armazenadas as variáveis e ponteiros utilizados durante a execução do programa. Já a memória de programas é onde de fato fica armazenado o código-fonte da aplicação.

Figura 25 – Consumo de Memória do AES Original e Modificado



Fonte: Elaborada pelo autor.

Em questão da memória de programas foi possível obter uma redução de 31,82%. Grande parte desta redução se deve ao fato da versão otimizada de *MixColumns* ser mais simples, utilizando um código bem menor para realizar as operações necessárias, também contribuindo para uma diminuição no tempo de execução. Já em memória dinâmica foi possível obter uma redução de 89,04%, o que é um número bem expressivo, e se deve ao fato da modificação implementada no estágio *SubBytes*, onde foi reduzido de duas s-box que totalizavam 512 bytes para uma única s-box que ocupa apenas 16 bytes.

4.2 Análise de Segurança

A análise de segurança consistiu em extrair métricas de efeito avalanche e critério de balanceamento de modo a verificar a qualidade da encriptação obtida pelo algoritmo proposto. Além disto, a versão otimizada também foi submetida aos testes do NIST, para verificar se a sequência binária por ela gerada era considerada randômica. A Análise de segurança foi conduzida no Macbook Air, porém, como

as estruturas arquiteturais dos embarcados são compatíveis com o processador M2 e seu compilador, seus resultados podem ser considerados de igual maneira nas plataformas embarcadas utilizadas nos testes.

4.2.1 Efeito Avalanche

A análise de efeito avalanche é primordial para avaliar a segurança de um algoritmo criptográfico. O efeito avalanche deve garantir que se uma mínima mudança no *plaintext* ou na chave, mesmo que de apenas 1 bit, isto deve causar uma mudança significativa no *ciphertext*(Mohammad; Abdullah, 2022). Nos testes realizados neste trabalho foram propostas mudanças na chave criptográfica, utilizando o conceito de distância Hamming entre duas strings, cuja distância corresponde ao número de caracteres diferentes entre elas. O cálculo efetivo do efeito avalanche se deu de acordo com a Equação 1.

A análise utiliza distâncias Hamming variando de 1 a 5. Em seguida, é calculada a média aritmética destes 5 resultados. Para este teste, as entradas utilizadas foram do tipo string e tinham tamanho de 1000 bytes. A Tabela 6 apresenta os resultados obtidos.

Tabela 6 – Valores de Efeito Avalanche para Diferentes Distâncias de Hamming

Algoritmo	**Distância de Hamming**					
	1	2	3	4	5	Média
AES Original	49,94%	51,10%	51,03%	50,47%	50,20%	**50,55%**
AES Modificado	50,55%	50,47%	51,37%	49,80%	49,86%	**50,41%**

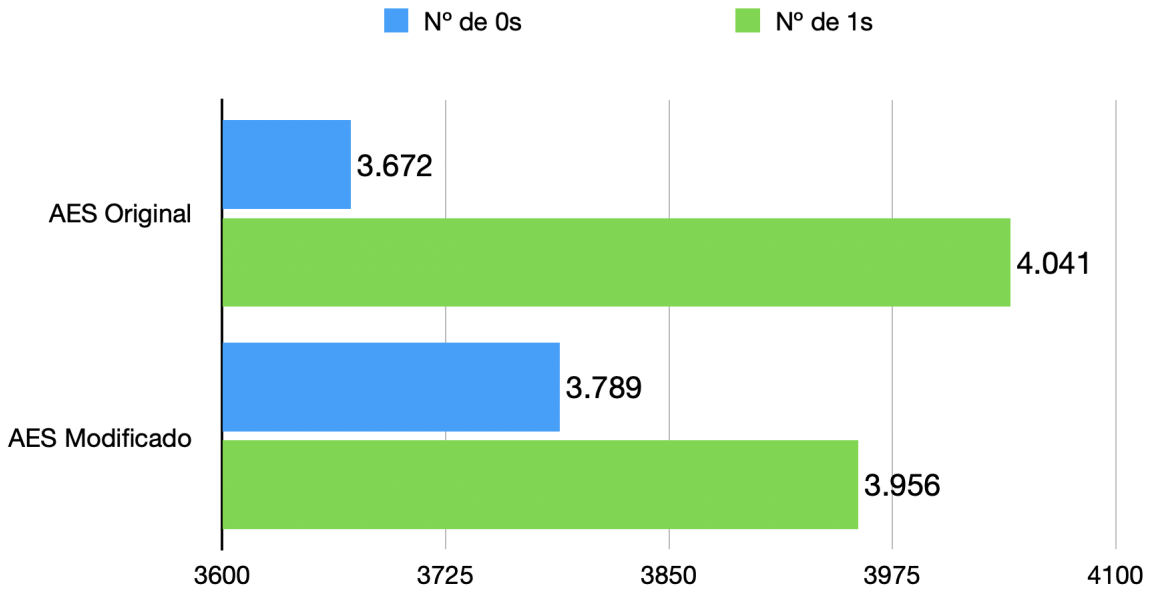
Fonte: Elaborada pelo autor.

O algoritmo AES original e modificado apresentaram um efeito avalanche médio de 50,55% e 50,41%, respectivamente, o que está de acordo com o valor esperado para este teste, onde devem-se atingir valores superiores a 50% (Biswas, 2023).

4.2.2 Critério de Balanceamento

Um dos principais critérios para testar se uma s-box está bem planejada é o cálculo de balanceamento. Uma s-box deve distribuir de forma balanceada a quantidade de 0's e 1's no *ciphertext*. A Figura 26 apresenta a distribuição de 0's e 1's de ambas versões do AES, quando uma entrada de 1000 bytes do tipo string é utilizada. Como é possível observar, o algoritmo modificado apresentou uma distribuição um pouco melhor na quantidade de 0's e 1's.

Figura 26 – Resultados do Critério de Balanceamento para Ambas Versões do AES



Fonte: Elaborada pelo autor.

4.2.3 NIST

Os testes do NIST são utilizados para verificar a aleatoriedade do *ciphertext* gerado por um algoritmo criptográfico, ou seja, verificar se a sequência binária produzida pelo algoritmo é randômica o suficiente(Rukhin; Soto; Nechvatal; Smid; Barker; Leigh; Levenson; Vangel; Banks; Heckert; Dray; Vo, 2010).

A saída destes testes para cada sequência binária é sempre aleatória ou não aleatória, o que significa, na prática, aprovada ou reprovada, respectivamente. Foram geradas 6 strings de forma aleatória, cada uma com 1000 bytes, e logo foram analisadas as sequências binárias geradas pelo algoritmo proposto. Ao executar os testes nas 6 sequências, foi possível verificar que todas foram aprovadas em cada um dos testes disponíveis.

4.3 Análise de Consumo Energético

Neste ponto do texto, os testes deixaram de ser conduzidor no Macbook Air e passaram a ser conduzidos nas plataformas embarcadas ESP-WROOOM-32 e Raspberry Pi Pico.

Para a análise de consumo energético, inicialmente verificou-se o tempo de execução de ambas versões do algoritmo AES nas duas plataformas. A função micros(), fornecida pela própria IDE do Arduino, foi utilizada para verificar o tempo transcorrido pela função de encriptação dos algoritmos. Nestes testes, os algoritmos foram exe-

cutados 100 vezes em série e logo foi calculada a média aritmética simples destes valores. A entrada utilizada foi de tipo string e teve o tamanho de um bloco, ou seja, 16 bytes. Os resultados obtidos são apresentados na Tabela 7, onde é possível verificar o tempo de execução de ambas versões, original e otimizada, do AES e o percentual de redução atingido.

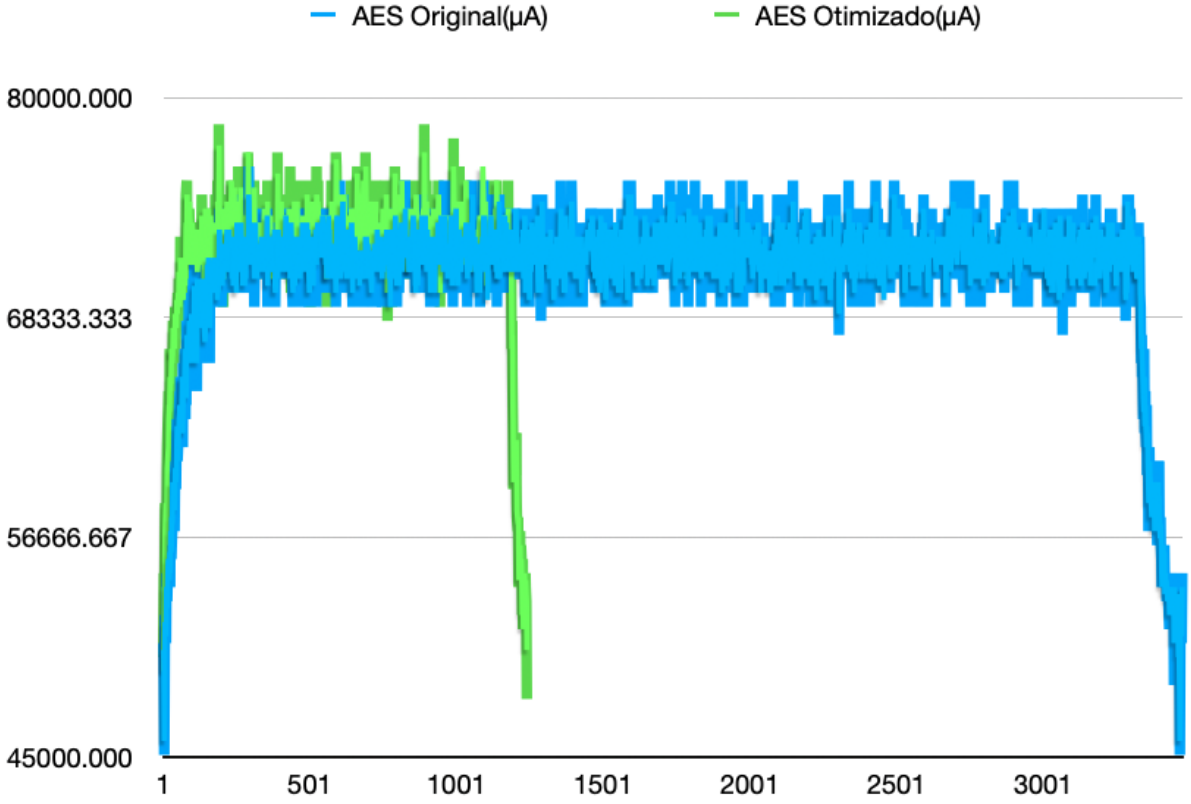
Tabela 7 – Tempos de Execução dos Algoritmos nas Plataformas

	AES Original(µs)	**AES Modif.(µs)**	**Redução(%)**
Raspberry Pi Pico	677	178	73,71%
ESP-WROOOM-32	312	103	66,99%

Fonte: Elaborada pelo autor.

Após, foram realizados os experimentos para verificar o consumo energético que as versões original e otimizada do AES apresentavam em ambas plataformas. A Figura 27 apresenta o gráfico de corrente, em uA, da execução de ambas versões, original e otimizada, do algoritmo AES na ESP-WROOOM-32.

Figura 27 – Valores de Corrente(uA) da Execução dos Algoritmos na ESP-WROOOM-32



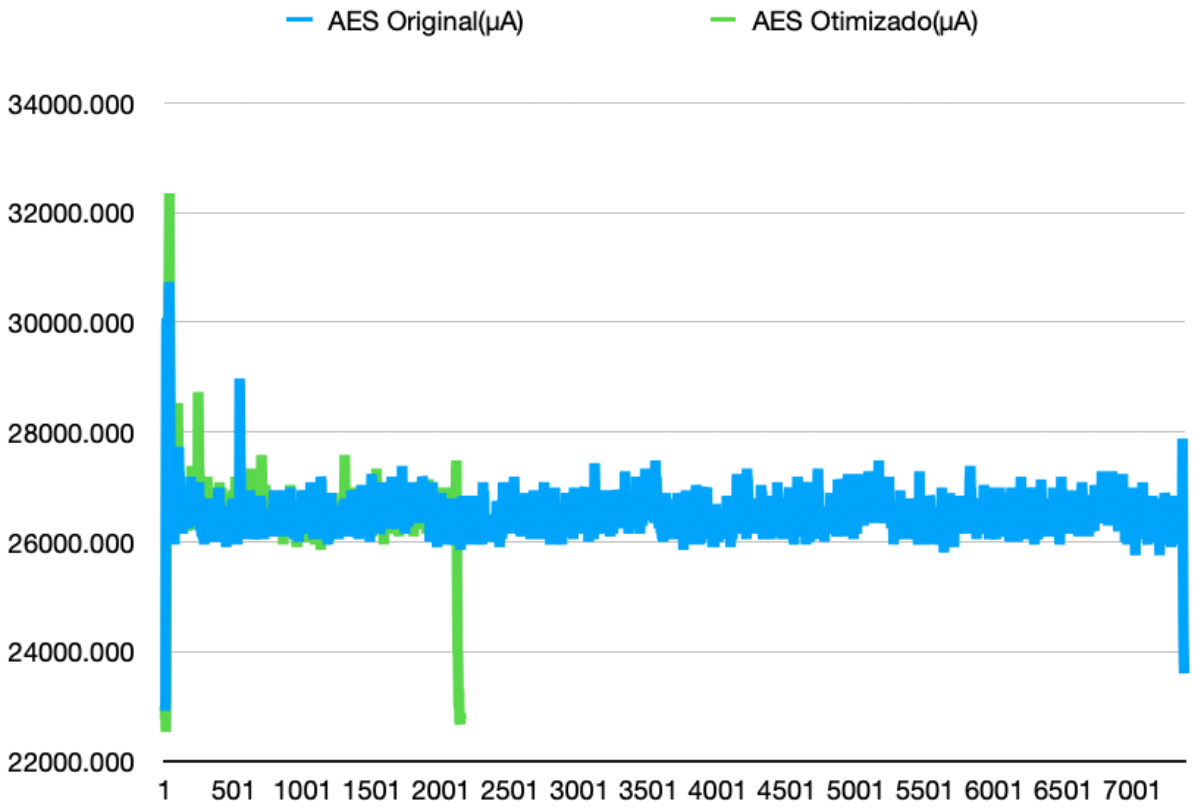
Fonte: Elaborada pelo autor.

É possível observar que o algoritmo AES original teve uma variação de corrente entre 45087uA e 76337uA. Já a versão otimizada teve uma variação de corrente entre

48055uA e 78576uA. É possível verificar que, para a ESP-WROOOM-32, a versão otimizada apresenta uma maior dissipação de potência, porém, como o algoritmo modificado executa em um tempo consideravelmente mais rápido, faz com que o o consumo de energia seja menor.

A Figura 28 apresenta o gráfico de corrente, em uA, de ambas versões, original e otimizada, do algoritmo AES na Raspberry Pi Pico.

Figura 28 – Valores de Corrente(uA) da Execução dos Algoritmos na Raspberry Pi Pico

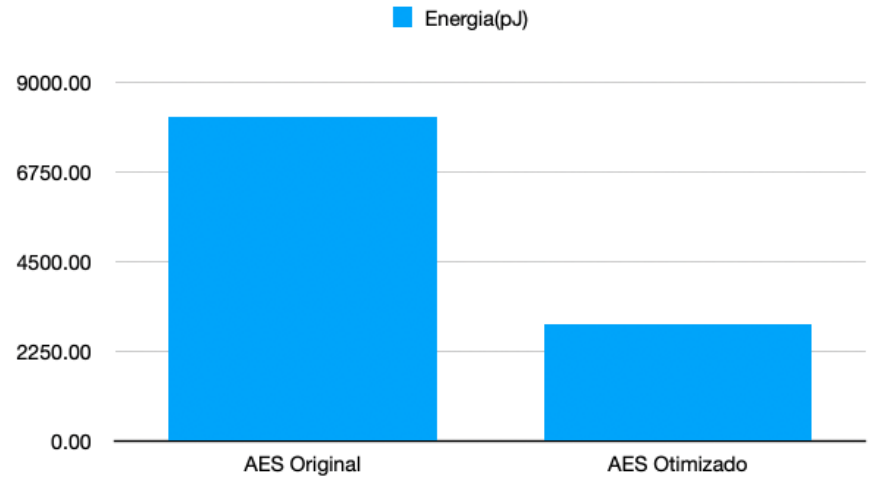


Fonte: Elaborada pelo autor.

É possível observar que o algoritmo AES original teve uma variação de corrente entre 22679uA e 30835uA, enquanto a versão otimizada teve uma variação de corrente entre 22485uA e 32354uA. É possível verificar que, para a Raspberry Pi Pico, a versão otimizada apresenta uma dissipação de potência ligeiramente maior, porém, o consumo energético final é bem menor, devido ao fato da versão otimizada ter um tempo de execução mais rápido.

Já em termos de energia total dissipada, as Figuras 29 e 30 apresentam os valores, em pJ, de cada plataforma para cada versão do algoritmo. Nelas, é possível verificar que o consumo energético da versão otimizada do AES proposta por esta dissertação é realmente menor, fazendo com que esta versão seja mais atraente para dispositivos com recursos computacionais limitados e/ou alimentados por baterias.

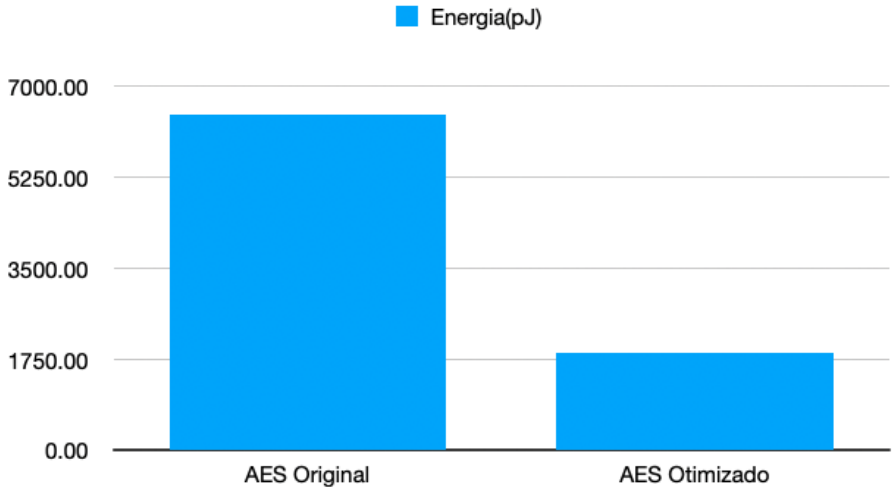
Figura 29 – Valores de Energia Dissipada(pJ) da Execução dos Algoritmos na ESP-WROOOM-32



Fonte: Elaborada pelo autor.

A versão proposta otimizada do AES atingiu uma redução de 63,87% em consumo de energia para a plataforma ESP-WROOOM-32. Isso ocorre principalmente pelo fato de a versão otimizada executar em um tempo consideravelmente menor, assim, mesmo que a potência dissipada na versão otimizada seja maior, leva a um consumo de energia menor.

Figura 30 – Valores de Energia Dissipada(pJ) da Execução dos Algoritmos na Raspberry Pi Pico



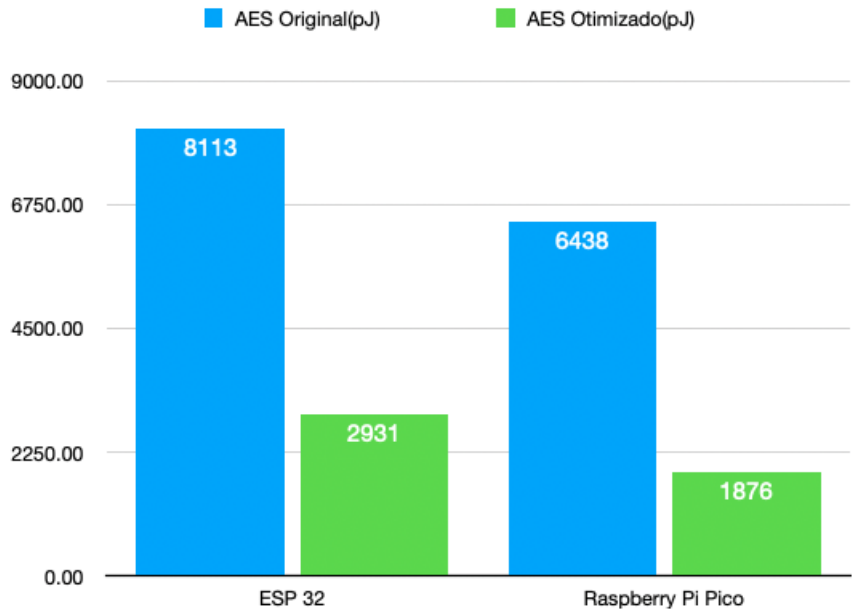
Fonte: Elaborada pelo autor.

Já pra a plataforma Raspberry Pi Pico, a versão proposta otimizada do AES atingiu uma redução de 70,86% em consumo de energia. De mesmo modo que para a plataforma ESP-WROOOM-32, esta redução esta fortemente relacionada a redução obtida

em tempo de execução.

A Figura 31 apresenta o comparativo de consumo energético de ambas as versões do algoritmo entre as duas plataformas utilizadas.

Figura 31 – Comparativo de Consumo Energético Entre as Plataformas



Fonte: Elaborada pelo autor.

É possível verificar que a Raspberry Pi Pico apresenta um menor consumo energético em ambas as versões do algoritmo AES. Porém, na ESP-WROOOM-32 ambas versões do AES apresentam um tempo de execução menor. Logo, dependendo do tipo de projeto que se tenha interesse em desenvolver, poderia se optar por utilizar a Raspberry Pi Pico se a maior necessidade seja consumo energético e utilizar a ESP-WROOOM-32 se haja necessidade de um tempo de execução menor.

4.4 Comparação dos Resultados com a Literatura

Com o objetivo de comparar a solução proposta neste trabalho com os outros 5 trabalhos relacionados da literatura, é possível concluir o seguinte:

- De Mohammad; Abdullah (2022) é possível comparar os tempos de execução com uma entrada de tamanho 20 bytes, onde foi por eles atingida uma redução de 41,38% comparado ao AES original, enquanto a redução atingida pelo algoritmo proposto por esta dissertação foi de 81,63%. Ambas otimizações apresentaram bom resultado de efeito avalanche, porém, a otimização desenvolvida por Mohammad; Abdullah (2022) não foi submetida aos testes de aleatoriedade de NIST, nem foi realizada avaliação de critério de balanceamento. Além disto,

questões energéticas também não foram avaliadas por Mohammad; Abdullah (2022).

- De Fadhil; Farhan; Fadhil; Al-saidi (2020) pode-se comparar os tempos de execução para entrada de tamanho 10000 bytes, onde eles atingiram uma redução de 91,11% comparado ao AES original, enquanto a modificação proposta por esta dissertação atingiu uma redução de 89,72% Apesar de ter uma redução ligeiramente menor, a modificação proposta por esta dissertação consome uma quantidade de memória bem menor, visto que a proposta de Fadhil; Farhan; Fadhil; Al-saidi (2020) utiliza 2 s-boxes de 256 bytes cada, enquanto a proposta desta dissertação utiliza apenas 16 bytes na s-box. Ambas otimizações avaliaram critério de balanceamento, efeito avalanche e NIST, porém, Fadhil; Farhan; Fadhil; Al-saidi (2020) não avaliaram o consumo energético.

- De Hammod (2022) é possível comparar os tempos de execução, para entradas de tamanho 20 e 200 bytes, onde foi por eles obtido uma redução de 13,6% e 4,23%, respectivamente, comparados ao AES original, enquanto a otimização proposta por esta dissertação atingiu uma redução de 81,63% e 88,7% para entradas de mesmo tamanho. Em questão de consumo de memória, Hammod (2022) utiliza 64 bytes em suas s-box, enquanto a modificação proposta utiliza uma única s-box de apenas 16 bytes. Ambas modificações foram avaliadas no teste do NIST e também em efeito avalanche, porém, Hammod (2022) não realizou avaliação de critério de balanceamento nem avaliou questões de consumo energético.

- De Shi; Wang; Jia; Peng; Jiang; Zhu (2019), em questões de tempo de encriptação, é possível comparar as proposta com a entrada de tamanho 1000 bytes, onde foi encontrado por Shi; Wang; Jia; Peng; Jiang; Zhu (2019) uma redução de 35% em comparação ao AES original, enquanto a proposta desta dissertação atingiu uma redução de 88,93%. Ambas propostas avaliaram efeito avalanche, tendo resultados aceitáveis, porém, a proposta de Shi; Wang; Jia; Peng; Jiang; Zhu (2019) não foi submetida aos testes do NIST nem realizou avaliação em termos de consumo de memória e consumo energético.

- De Naif; Abdul-majeed; Farhan (2019) pode-se comparar o tempo de execução com entradas de tamanho 1000 e 10000 bytes, onde a otimização proposta por Naif; Abdul-majeed; Farhan (2019) atingiu uma redução de 30,19% e 17,7%, respectivamente, comparado ao AES original, enquanto a proposta deste artigo atingiu uma redução de 88,93% e 89,72%, respectivamente. Naif; Abdul-majeed; Farhan (2019) validou sua otimização com os testes do NIST, porém, não realizou testes de efeito avalanche nem observou questões de consumo de memória,

além de não ter avaliado questões de energia.

A Figura 32 apresenta um resumo das comparações realizadas entre os artigos. Os comparativos realizados em termos de encriptação foram valores obtidos na execução do algoritmo no Macbook Air.

Figura 32 – Resumo das Comparações Entre as Propostas

Artigo	Tamanho da Entrada(B) Tempo de Encriptação: Redução em %				Efeito Avalanche	Critério de Balanceamento	NIST	Consumo Energético
	20	200	1000	10000				
Mohammad; Abdullah (2022)	41.38%	-	-	-	✅	❌	❌	❌
Fadhil et al. (2020)	-	-	-	91.11%	✅	✅	✅	❌
Hammod (2022)	13.6%	4.23%	-	-	✅	❌	✅	❌
Shi et al. (2019)	-	-	35%	-	✅	❌	❌	❌
Naif; Abdul-majeed; Farhan (2019)	-	-	30.19%	17.7%	❌	❌	✅	❌
Este Trabalho	**81.63%**	**88.7%**	**88.93%**	**89.72%**	✅	✅	✅	✅

Fonte: Elaborada pelo autor.

Os cinco trabalhos relacionados apresentados, de mesmo modo que esta dissertação, apresentaram formas de otimizar o algoritmo criptográfico AES para melhorar alguma métrica. Porém, somente foi possível realizar o comparativo em termos de tempo de execução, visto que os demais autores não se preocuparam em avaliar consumo energético, para de fato saber se a versão proposta otimizada trouxe benefícios em redução de consumo que é, de fato, uma das maiores preocupações no desenvolvimento de aplicações IoT.

4.5 Considerações Finais Sobre o Capítulo

Este capítulo buscou apresentar os resultados obtidos em três principais grupos: análise de desempenho, análise de segurança e análise de consumo energético. A análise de desempenho apresentou resultados de tempo de execução para diferentes tamanho de entrada e também apresentou questões de consumo de memória extraídas com auxílio da IDE do Arduino. Ambas as versões, original e proposta, foram utilizadas nos testes para poder comparar seus resultados e calcular o percentual de redução obtido.

Já em questões de segurança, foram utilizados três diferentes testes para assegurar a qualidade da encriptação, que foram os testes de efeito avalanche, critério de balanceamento e NIST. De acordo com os resultados obtidos nestes testes, pode-se afirmar que a versão proposta é segura para utilização.

Em questão de consumo energético, foram utilizadas as plataformas ESP-WROOOM-32 e Raspberry Pi Pico nos testes, onde foi possível verificar grandes reduções tanto em tempo de execução quanto em consumo energético que é, de fato, uma das maiores aplicações quando se trata de aplicações IoT. Embora a potência dissipada pela versão otimizada do algoritmo AES, tanto na ESP-WROOOM-32 quanto na Raspberry Pi Pico, tenha sido maior, o consumo energético final é menor na versão otimizada. Isso se deve ao fato do consumo energético estar muito relacionado com o tempo de execução do algoritmo, que, na versão otimizada, é consideravelmente menor. Ao final do capítulo, foi realizado um comparativo dos resultados obtidos por esta dissertação com outros cinco artigos que tinha temas afins a este.

5 CONCLUSÃO

Este trabalho teve como objetivo propôr uma versão *lightweight* do algoritmo AES, de modo que sua utilização seja viável em aplicações IoT, no qual são usualmente implementadas em dispositivos com recursos computacionais limitados. Para isto, foram propostas modificações nos estágios *SubBytes* e *MixColumns* do algoritmo criptográfico AES, onde se buscou redução de consumo de memória e redução de tempo de execução, respectivamente, o que acarretou em uma redução de consumo energético.

A modificação realizada no estágio *SubBytes* consistiu na utilização de uma s-box menor, saindo de duas s-boxes de 256 bytes cada, totalizando 512B, do algoritmo AES original, para apenas uma s-box de 16 bytes. Já no estágio *MixColumns*, a modificação se concentrou em utilizar operações de adição e multiplicação mais leves, fazendo com que o algoritmo tivesse uma execução mais rápida.

A média de redução de tempo de execução entre os diferentes tamanhos de entrada avaliados, quando comparados ao AES original, foi de 86,71%, levando o algoritmo a ter um tempo de execução bem menor. Já em termos de consumo de memória, a versão otimizada apresentou bons resultados tornando o algoritmo bem mais leve, atingindo reduções de 31,82% e 89,04% em alocação de memória de programa e memória dinâmica, respectivamente, fazendo com que seja mais viável sua utilização em dispositivos com recursos computacionais limitados.

Em questões de segurança, o algoritmo proposto foi validado com o teste de efeito avalanche e critério de balanceamento, além de ser aprovado nos testes de aleatoriedade do NIST.

Por último, foi possível verificar que as otimizações implementadas acarretaram em uma diminuição no consumo energético, atingindo reduções de 63,87% e 70,86% nas plataformas ESP-WROOOM-32 e Raspberry Pi Pico, respectivamente.

Concluindo, é possível afirmar que a versão otimizada proposta por esta dissertação é apropriada para utilização em aplicações IoT, pois foram atingidas grandes reduções tanto em desempenho quanto em consumo energético, sendo assim uma versão atraente para ser implementada em dispositivos com recursos computacionais limitados.

Como trabalhos futuros, sugerem-se alguns possíveis caminhos:

- Implementar variações no estágio *ShiftRows* para trabalhar com valores dinâmicos de deslocamentos com o intuito de verificar se geraria valores ainda melhores de efeito avalanche;
- Testar variações no número de rodadas do algoritmo, considerando valores menores que 10 rodadas, para avaliar o impacto que isso traria nas métricas de segurança;
- Analisar a remoção do estágio *MixColumns* combinado com um aumento no número de rodadas do algoritmo e verificar como ficariam as métricas de segurança do algoritmo;
- Realizar testes com outras plataformas embarcadas;
- Realizar testes em aplicações reais que façam uso da criptografia com objetivo de avaliar o impacto no cenário real;
- Avaliar o comportamento do consumo energético alimentando as plataformas embarcadas com bateria.

REFERÊNCIAS

A. MEHDI, S.; JABBAR, K.; FADHIL, H.; ABBOOD, F. H. Image Encryption Based on the Novel 5d Hyper-Chaotic System Via Improved Aes Algorithm. **International Journal of Civil Engineering and Technology (IJCIET)**, [S.l.], v.9, n.10, p.1841–1855, 2018.

ABDALLAH, E. G.; KUANG, Y. R.; HUANG, C. Advanced Encryption Standard New Instructions (AES-NI) Analysis: Security, Performance, and Power Consumption. In: INTERNATIONAL CONFERENCE ON COMPUTER AND AUTOMATION ENGINEERING, 2020., 2020, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2020. p.167–172. (ICCAE 2020).

AFZAL AWAN, A. Elixir: A 128-bit Stream Cipher Protocol for Lightweight IoT Devices. In: INTERNATIONAL CONFERENCE ON CYBER WARFARE AND SECURITY (ICCWS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

AHMAD, A. A New Security Method for the Internet of Things Based on Ciphering and Deciphering Algorithms. **Kirkuk Journal of Science**, [S.l.], v.13, n.3, p.154–174, 2018.

AL-NASSER, A. Performance Evaluation and Review of Lightweight Cryptography in an Internet-of-Things Environment. In: INTERNATIONAL CONFERENCE ON COMPUTER APPLICATIONS & INFORMATION SECURITY (ICCAIS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–6.

AL-SHARGABI, B.; ASSI, A. D. An Improved DNA based Encryption Algorithm for Internet of Things Devices. In: INTERNATIONAL CONFERENCE ON ENGINEERING (ICEMIS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–5.

ALBARELLO, R.; OYAMADA, M.; CAMARGO, E. de. Avaliação de Algoritmos de Criptografia e Implementação de um Protocolo Leve para Comunicação entre Dispositivos IoT. In: ESTENDIDOS DO X SIMPóSIO BRASILEIRO DE ENGENHARIA DE SISTEMAS COMPUTACIONAIS, 2020, Porto Alegre, RS, Brasil. **Anais...** SBC, 2020. p.65–72.

AMAZON, A. **O que é a Internet das Coisas (IoT)?** Disponível em: <https://aws.amazon.com/pt/what-is/iot/>. Acesso em: 2023-12-13.

A.MOHAMMED, Z.; HUSSEIN, K. A. Lightweight Cryptography Concepts and Algorithms: A Survey. In: SECOND INTERNATIONAL CONFERENCE ON ADVANCED COMPUTER APPLICATIONS (ACA), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.1–7.

ASHTON, K. That "internet of things"things. **RFID Journal**, [S.l.], v.22, n.7, p.97–114, 2009.

BISWAS, A. e. a. LRBC: a lightweight block cipher design for resource constrained IoT devices. **Journal of Ambient Intelligence and Humanized Computing**, [S.l.], v.14, p.5773–5787, 05 2023.

BUCHANA, W. J. Recent Advances and Trends in Lightweight Cryptography for IoT Security. In: INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT (CNSM), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–5.

DANDJINOU, T. M. An improvement of the AES protocol to optimize energy consumption in IoT. In: IEEE MULTI-CONFERENCE ON NATURAL AND ENGINEERING SCIENCES FOR SAHEL'S SUSTAINABLE DEVELOPMENT (MNE3SD), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–5.

DWORKIN, M.; BARKER, E.; NECHVATAL, J.; FOTI, J.; BASSHAM, L.; ROBACK, E.; DRAY, J. **Advanced Encryption Standard (AES)**. Disponível em: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>. Acesso em: 2023-12-09.

FADHIL, M. S.; FARHAN, A. K.; FADHIL, M. N.; AL-SAIDI, N. M. A New Lightweight AES Using a Combination of Chaotic Systems. In: INFORMATION TECHNOLOGY TO ENHANCE E-LEARNING AND OTHER APPLICATION (IT-ELA, 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.82–88.

FUTURE, H. **Lightweight cryptography for strong security of the Internet of Things**. Disponível em: <https://hellofuture.orange.com/en/lightweight-cryptography-for-strong-security-of-the-internet-of-things/>. Acesso em: 2023-12-09.

GHORASHI, S. R.; ZIA, T.; JIANG, Y. Optimisation of Lightweight Klein Encryption Algorithm With 3 S-box. In: IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS (PERCOM WORKSHOPS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–5.

GILLIS, A. **What is the internet of things (IoT)?** Disponível em: <https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>. Acesso em: 2023-12-13.

GIROTI, I.; MALHOTRA, M. Quantum Cryptography: A Pathway to Secure Communication. In: INTERNATIONAL CONFERENCE ON COMPUTATION SYSTEM AND INFORMATION TECHNOLOGY FOR SUSTAINABLE SOLUTIONS (CSITSS), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–6.

GUNATHILAKE, N. A.; AL-DUBAI, A.; BUCHANA, W. J. Recent Advances and Trends in Lightweight Cryptography for IoT Security. In: INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT (CNSM), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–5.

GUO, Y.; LI, L.; LIU, B. Shadow: A Lightweight Block Cipher for IoT Nodes. **IEEE Internet of Things Journal**, [S.l.], v.8, n.16, p.13014–13023, 2021.

HAMMOD, D. N. Modified Lightweight AES based on Replacement Table and Chaotic System. In: INTERNATIONAL CONGRESS ON HUMAN-COMPUTER INTERACTION, OPTIMIZATION AND ROBOTIC APPLICATIONS (HORA), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.1–5.

HASAN, A. S. M. T. G-TBSA: A Generalized Lightweight Security Algorithm for IoT. In: INTERNATIONAL CONFERENCE ON ELECTRICAL INFORMATION AND COMMUNICATION TECHNOLOGY (EICT), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–6.

HERCOG, D.; LERHER, T.; TRUNTIč, M.; TEžAK, O. Design and Implementation of ESP32-Based IoT Devices. **Sensors**, [S.l.], v.23, n.15, 2023.

IBM. **Criptografia de Chave Pública**. Disponível em: <https://www.ibm.com/docs/pt-br/integration-bus/10.0?topic=overview-public-key-cryptography>. Acesso em: 2024-02-12.

JIANWEI, L. Lightweight Encryption Algorithm Implementation for Internet of Thing Application. In: INTERNATIONAL CONFERENCE ON CYBER WARFARE AND SECURITY (ICCWS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

KUMAR, R. Sponge based Lightweight Cryptographic Hash Functions for IoT Applications. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TECHNOLOGIES (CONIT), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.1–5.

LOKER, D. R. Raspberry Pi Pico as an IoT Device. In: ASEE ANNUAL CONFERENCE & EXPOSITION, 2023., 2023, Baltimore , Maryland. **Anais...** ASEE Conferences, 2023. https://peer.asee.org/44016.

LUZ, B. **CRIPTOGRAFIA SIMÉTRICA E ASSIMÉTRICA**: VANTAGENS E DESVANTAGENS. Disponível em: <https://brunaluz-11690.medium.com/criptografia-simétrica-e-assimétrica-vantagens-e-desvantagens-38e614d201eb>. Acesso em: 2023-12-09.

MACCARRONE, D. **10 relevant IoT trends in 2023**. Disponível em: <https://www.loriot.io/blog/IoT-trends-2023.html>. Acesso em: 2023-12-11.

MAJUMDAR, A.; BISWAS, A.; BAISHNAB, K.; SOOD, S. DNA Based Cloud Storage Security Framework Using Fuzzy Decision Making Technique. **KSII Transactions on Internet and Information Systems**, [S.l.], v.13, p.3794–3820, 07 2019.

MARWEDEL, P. **Embedded System Design**. Dortmund, Germany: Springer, 2021. 433p.

MCKINSEY. **What is the Internet of Things?** Disponível em: <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-the-internet-of-things>. Acesso em: 2024-01-10.

MEDELEANU, F.; RACUCIU, C.; ROGOBETE, M. CONSIDERATIONS ABOUT THE POSSIBILITIES TO IMPROVE AES S-BOX CRYPTOGRAPHIC PROPERTIES BY MULTIPLICATION. **PROCEEDINGS OF THE ROMANIAN ACADEMY**, [S.l.], v.16, p.339–344, 2015.

MISHRA, N.; PANDYA, S. Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review. **IEEE Access**, [S.l.], v.PP, 04 2021.

MOHAMMAD, H.; ABDULLAH, A. Enhancement process of AES: a lightweight cryptography algorithm-AES for constrained devices. **TELKOMNIKA (Telecommunication Computing Electronics and Control)**, [S.l.], v.20, p.551, 06 2022.

MULLEN, J. **What is the US Cyber Trust Mark?** Disponível em: <https://www.keysight.com/blogs/en/tech/nwvs/2023/10/25/what-is-us-cyber-trust-mark>. Acesso em: 2024-01-10.

NAIF, J. R.; ABDUL-MAJEED, G. H.; FARHAN, A. K. Secure IOT System Based on Chaos-Modified Lightweight AES. In: INTERNATIONAL CONFERENCE ON ADVANCED SCIENCE AND ENGINEERING (ICOASE), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–6.

NEHA, T. **Advanced Encryption Standard (AES)**. Disponível em: <https://binaryterms.com/advanced-encryption-standard-aes.html>. Acesso em: 2023-12-08.

NUHA, H. H. A Comparative Method for Securing Internet of Things (IoT) Devices: AES vs Simon-Speck Encryptions. In: INTERNATIONAL CONFERENCE ON INFORMATION SYSTEM & INFORMATION TECHNOLOGY (ICISIT), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.392–396.

PI, R. **Raspberry Pi Pico**. Disponível em: <https://www.raspberrypi.com/products/raspberry-pi-pico/>. Acesso em: 2024-02-19.

REINSEL, D.; GANTZ, J.; RYDNING, J. **The Digitization of the World**: From Edge to Core. Disponível em: <https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf>. Acesso em: 2023-12-13.

RUKHIN, A.; SOTO, J.; NECHVATAL, J.; SMID, M.; BARKER, E.; LEIGH, S.; LEVENSON, M.; VANGEL, M.; BANKS, D.; HECKERT, A.; DRAY, J.; VO, S. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Application. **NIST Special Publication 800-22**, [S.l.], 2010.

SADHU, P. K.; YANAMBAKA, V. P.; ABDELGAWAD, A. Internet of Things: Security and Solutions Survey. **Sensors**, [S.l.], v.22, n.19, 2022.

SADIO, O.; NGOM, I.; LISHOU, C. Lightweight Security Scheme for MQTT/MQTT-SN Protocol. In: SIXTH INTERNATIONAL CONFERENCE ON INTERNET OF THINGS: SYSTEMS, MANAGEMENT AND SECURITY (IOTSMS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.119–123.

SALMAN, R. S.; FARHAN, A. K.; SHAKIR, A. Lightweight Modifications in the Advanced Encryption Standard (AES) for IoT Applications: A Comparative Survey. In: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND SOFTWARE ENGINEERING (CSASE), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.325–330.

SALMAN, R. S.; FARHAN, A. K.; SHAKIR, A. Lightweight Modifications in the Advanced Encryption Standard (AES) for IoT Applications: A Comparative Survey. In: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND SOFTWARE ENGINEERING (CSASE), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.325–330.

SARAIVA, D. A. F.; LEITHARDT, V. R. Q.; PAULA, D.; MENDES, A. S.; GONZáLEZ, G. V.; CROCKER, P. PRISEC: Comparison of Symmetric Key Algorithms for IoT Devices. **Sensors (Basel)**, [S.l.], v.19, p.4312, 2019.

SEMICONDUCTOR, N. **Power Profiler Kit II - documentation**. Disponível em: <https://nsscprodmedia.blob.core.windows.net/prod/software-and-other-downloads/product-briefs/power-profiler-kit-ii-pbv10.pdf>. Acesso em: 2023-12-18.

SHAIKH, Z. A.; HAJJEJ, F.; USLU, Y. D.; YüKSEL, S.; DıNçER, H.; ALROOBAEA, R.; BAQASAH, A. M.; CHINTA, U. A New Trend in Cryptographic Information Security for Industry 5.0: A Systematic Review. **IEEE Access**, [S.l.], v.12, p.7156–7169, 2024.

SHARATH, G. A4: A Lightweight Stream Cipher. In: INTERNATIONAL CONFERENCE ON COMMUNICATION AND ELECTRONICS SYSTEMS (ICCES), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.573–577.

SHI, L.; WANG, Y.; JIA, R.; PENG, T.; JIANG, J.; ZHU, S. Research of Lightweight Encryption Algorithm Based on AES and Chaotic Sequences for Narrow-Band Internet of Things. In: MACHINE LEARNING AND INTELLIGENT COMMUNICATIONS, 2019. **Anais...** [S.l.: s.n.], 2019. v.295, p.267–280.

SINGH, S. **The Code Book**. London, UK: Clays Ltd, 2000. 402p.

SINHA, S. **State of IoT 2023**: Number of connected IoT devices growing 16% to 16.7 billion globally. Disponível em: <https://iot-analytics.com/number-connected-iot-devices/>. Acesso em: 2023-12-06.

SRIVASTAVA, S.; TIWARI, A.; SRIVASTAVA, P. K. Review on quantum safe algorithms based on Symmetric Key and Asymmetric Key Encryption methods. In: INTERNATIONAL CONFERENCE ON ADVANCE COMPUTING AND INNOVATIVE TECHNOLOGIES IN ENGINEERING (ICACITE), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. p.905–908.

STALLINGS, W. **Criptografia e segurança de redes**: princípios e práticas. São Paulo, Brasil: Pearson Education do Brasil, 2015. 558p.

SULTAN, I.; MIR, B. J.; BANDAY, M. T. Analysis and Optimization of Advanced Encryption Standard for the Internet of Things. In: INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING AND INTEGRATED NETWORKS (SPIN), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.571–575.

SYSTEMS, E. **ESP32**. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 2024-02-19.

SýS, M.; RIHA, Z.; MATYAS, V.; MARTON, K.; SUCIU, A. On the interpretation of results from the NIST statistical test suite. **ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY**, [S.l.], v.18, p.18–32, 01 2015.

TOSHIHIKO, O. **Lightweight Cryptography Applicable to Various IoT Devices**. Disponível em: <https://www.nec.com/en/global/techrep/journal/g17/n01/170114.html>. Acesso em: 2023-12-06.

TSAI, K.-L.; HUANG, Y.-L.; LEU, F.-Y.; YOU, I.; HUANG, Y.-L.; TSAI, C.-H. AES-128 based Secure Low Power Communication for LoRaWAN IoT Environments. **IEEE Access**, [S.l.], v.PP, p.1–1, 07 2018.

ULLAH, S.; RADZI, R. Z.; YAZDANI, T. M.; ALSHEHRI, A.; KHAN, I. Types of Lightweight Cryptographies in Current Developments for Resource Constrained Machine Type Communication Devices: Challenges and Opportunities. **IEEE Access**, [S.l.], v.10, p.35589–35604, 2022.

WAHID, K. A. Comparative Performance Analysis of Lightweight Cryptography Algorithms for IoT Sensor Nodes. **IEEE Internet of Things Journal**, [S.l.], v.8, n.10, p.8279–8290, 2021.

WOHLIN, C.; RUNESON, P.; da Mota Silveira Neto, P. A.; ENGSTRöM, E.; do Carmo Machado, I.; de Almeida, E. S. On the reliability of mapping studies in software engineering. **Journal of Systems and Software**, [S.l.], v.86, n.10, p.2594–2610, 2013.

WOLFF, A. **Cryptojacking Continues Crushing Records**. Disponível em: <https://blog.sonicwall.com/en-us/2023/08/cryptojacking-continues-crushing-records>. Acesso em: 2023-12-11.

YAO, X. Design of School Assets Management Data Encryption Security System Based on RSA Algorithm. In: INTERNATIONAL CONFERENCE ON AMBIENT INTELLIGENCE, KNOWLEDGE INFORMATICS AND INDUSTRIAL ELECTRONICS (AIKIIE), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.1–5.

YELAMARTHI, K. Performance Evaluation of IoT Encryption Algorithms: Memory, Timing, and Energy. In: IEEE SENSORS APPLICATIONS SYMPOSIUM (SAS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–6.

ZAKARIA, A. A.; AZNI, A.; RIDZUAN, F.; ZAKARIA, N. H.; DAUD, M. Systematic literature review: Trend analysis on the design of lightweight block cipher. **Journal of King Saud University - Computer and Information Sciences**, [S.l.], v.35, n.5, p.101550, 2023.

Apêndices

Apêndice A: Cálculos Complementares

Com o intuito de ampliar as estatística apresentadas na Tabela 5, os valores mínimos, máximos e de desvio padrão são apresentados na Tabela 8. Os valores apresentados nesta tabela são da análise incial do AES, executados no Macbook Air.

Tabela 8 – Valores Mínimos, Máximos e de Desvio Padrão de Ambas Versões do AES

Entrada(bytes)	AES Original			AES Otimizado		
	Mínimo(µs)	Máximo(µs)	Desvio Padrão(µs)	Mínimo(µs)	Máximo(µs)	Desvio Padrão(µs)
10	37,12	43,78	2,60	7,96	8,35	0,15
25	73,29	75,50	0,99	11,99	13,62	0,72
70	169,15	174,14	1,89	18,94	23,89	2,26
100	240,10	246,13	2,26	27,80	30,20	1,10
1000	2126,92	2133,98	4,57	219,27	221,71	1,30
2000	4213,10	4228,94	8,15	433,94	434,91	0,49
10000	21080,51	21102,57	15,60	2166,34	2168,12	0,89

Fonte: Elaborada pelo autor.

Apêndice B: Publicações

VAZ, YURI SILVA; MATTOS, JÚLIO C. B; SOARES, RAFAEL IANKOWSKI. **AES Otimizado para Uso em Aplicações IoT**. In: Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC), 2023, Porto Alegre, Brasil. Anais Estendidos do XIII Simpósio Brasileiro de Engenharia de Sistemas Computacionais, 2023.

VAZ, YURI SILVA; MATTOS, JÚLIO C. B.; SOARES, RAFAEL IANKOWSKI. **Improving an Ultra Lightweight AES for IoT Applications**. In: IEEE 9th World Forum on Internet of Things (WF-IoT), 2023, Aveiro, Portugal. Proceedings of IEEE 9th World Forum on Internet of Things, 2023.