UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
MESTRADO EM ENGENHARIA ELÉTRICA

Diego Pereira Targino

Criptografia de baixo consumo em sistemas embarcados aplicados à Internet das coisas

MOSSORÓ
2024

Diego Pereira Targino

Criptografia de baixo consumo em sistemas embarcados aplicados à Internet das coisas

Dissertação de mestrado apresentado ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal Rural do Semi-Árido como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Linha de Pesquisa: Telecomunicações e Eletromagnetismo Aplicado.

Orientador: Prof. Dr. Francisco de Assis Brito Filho

Co-orientador: Prof. Dr. Paulo Henrique Lopes Silva

MOSSORÓ

2024

© Todos os direitos estão reservados a Universidade Federal Rural do Semi-Árido. O conteúdo desta obra é de inteira responsabilidade do (a) autor (a), sendo o mesmo, passível de sanções administrativas ou penais, caso sejam infringidas as leis que regulamentam a Propriedade Intelectual, respectivamente, Patentes: Lei n° 9.279/1996 e Direitos Autorais: Lei n° 9.610/1998. O conteúdo desta obra tomar-se-á de domínio público após a data de defesa e homologação da sua respectiva ata. A mesma poderá servir de base literária para novas pesquisas, desde que a obra e seu (a) respectivo (a) autor (a) sejam devidamente citados e mencionados os seus créditos bibliográficos.

TT185 Targino, Diego Pereira.
c Criptografia de baixo consumo em sistemas embarcados aplicados à Internet das coisas / Diego Pereira Targino. - 2024.
90 f. : il.

Orientador: Francisco de Assis Brito.
Coorientador: Paulo Henrique Lopes Silva.
Dissertação (Mestrado) - Universidade Federal Rural do Semi-árido, Programa de Pós-graduação em Engenharia Elétrica, 2024.

1. Internet das Coisas. 2. , Baixo Consumo. 3. Criptografia Leve. 4. Segurança da Informação. I. Brito, Francisco de Assis , orient. II. Silva, Paulo Henrique Lopes , co-orient. III. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Mossoró / Setor de Informação e Referência
Bibliotecária: Keina Cristina Santos Sousa e Silva
CRB: 15/120

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

Diego Pereira Targino

Criptografia de baixo consumo em sistemas embarcados aplicados à Internet das coisas

Dissertação de mestrado apresentado ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal Rural do Semi-Árido como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Defendido em 25/07/2024

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br
FRANCISCO DE ASSIS BRITO FILHO
Data: 29/07/2024 10:29:32-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Francisco de Assis Brito Filho
Orientador (Presidente) - UFERSA

Documento assinado digitalmente
gov.br
PAULO HENRIQUE LOPES SILVA
Data: 29/07/2024 09:44:41-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Paulo Henrique Lopes Silva
Examinador Interno - UFERSA

Documento assinado digitalmente
gov.br
JOAO BATISTA BORGES NETO
Data: 29/07/2024 09:50:37-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Joao Batista Borges Neto
Examinador Externo - UFRN

AGRADECIMENTOS

Gostaria de expressar minha gratidão ao meu orientador, o Prof. Dr. Francisco de Assis Brito Filho. O seu apoio e orientação foram fundamentais durante o desenvolvimento deste trabalho. Sua paciência e dedicação não apenas enriqueceram meus conhecimentos sobre o tema, mas também me inspiraram para a minha trajetória acadêmica. Agradeço por acreditar no meu potencial e por dedicar o seu tempo e conhecimento para o meu crescimento.

Meu agradecimento ao meu coorientador, Prof. Dr. Paulo Henrique Lopes Silva por ter aceitado o convite e nos auxiliar neste caminho com o seu conhecimento.

RESUMO

A Internet das Coisas (IoT, do inglês *Internet of Things*) é um conceito que se refere à conexão via Internet de objetos do cotidiano, como dispositivos vestíveis, eletrodomésticos, veículos e edifícios. Esses elementos estão equipados com sensores, softwares e outros recursos tecnológicos que possibilitam a coleta e a troca de dados sem a intervenção humana. Esta rede de dispositivos conectados tem o objetivo de tornar o mundo mais inteligente e eficiente, oferecendo às empresas e aos indivíduos a capacidade de monitorar e controlar seus ambientes e ativos em tempo real. Contudo, a IoT também traz consigo diversos desafios de segurança. Pois sendo composta por uma vasta gama de dispositivos interconectados, muitos deles são vulneráveis a ataques cibernéticos, tornando a segurança dos dados e da rede uma preocupação significativa. Para garantir a segurança e a privacidade dos dados transmitidos na rede IoT, é fundamental a utilização de técnicas de criptografia, que visam tornar os dados ininteligíveis para terceiros não autorizados. No entanto, os dispositivos de IoT possuem dimensões reduzidas e limitações de capacidade de processamento, armazenamento de dados e fornecimento de energia. Nesse contexto, surge a criptografia leve, com o objetivo de assegurar a privacidade dos dados mesmo em dispositivos com recursos limitados. Existem diversas técnicas de criptografia leve que podem ser aplicadas na IoT, incluindo a criptografia simétrica e assimétrica. A criptografia simétrica usa uma chave privada compartilhada para cifrar e decifrar dados, enquanto a assimétrica utiliza duas chaves diferentes, uma pública e outra privada. Neste trabalho, analisaremos apenas o desempenho das criptografias simétricas em dispositivos de capacidade limitada. Serão abordadas as criptografias de bloco AES, Camellia, XXETA, Simon e Speck, as criptografias de fluxo Chacha20 e Salsa20, e as funções *hash* BLAKE-S, SHA-256 e SHA3-256. Além disso, apresentaremos os modos de criptografia implementados e testados com o auxílio de dois kits de desenvolvimento diferentes: o ESP32 DevKit V1 e o Arduino Nano V3. Nos testes, avaliaremos o desempenho, o consumo de energia e a latência das cifras mencionadas, com o objetivo de demonstrar os impactos no consumo e desempenho nesses dispositivos.

Palavras-chave: Internet das Coisas, Baixo Consumo, Criptografia Leve, Segurança da Informação

ABSTRACTS

The Internet of Things (IoT) is a concept that refers to the connection of everyday objects to the Internet, such as wearable devices, household appliances, vehicles, and buildings. These objects are equipped with sensors, software, and other technological resources that enable the collection and exchange of data without human intervention. This network of connected devices aims to make the world smarter and more efficient, providing businesses and individuals with the ability to monitor and control their environments and assets in real-time. However, IoT also brings with it several security challenges. As it is composed of a vast range of interconnected devices, many of them are vulnerable to cyberattacks, making data and network security a significant concern. To ensure the security and privacy of data transmitted across the IoT network, it is essential to employ encryption techniques, which aim to render the data unintelligible to unauthorized third parties. However, reduced dimensions and limitations in processing power, data storage, and energy supply characterize IoT devices. In this context, lightweight cryptography emerges with the goal of ensuring data privacy even in devices with limited resources. There are various lightweight cryptographic techniques that can be applied in IoT, including symmetric and asymmetric cryptography. Symmetric cryptography uses a shared private key to encrypt and decrypt data, while asymmetric cryptography employs two different keys, one public and the other private. In this paper, we will analyze the performance of symmetric encryptions on low-capacity devices. The block ciphers AES, Camellia, XXETA, Simon, and Speck, the stream ciphers Chacha20 and Salsa20, and the hash functions BLAKE-S, SHA-256, and SHA3-256 will be addressed. Additionally, we will present the encryption modes implemented and tested using two different development kits: the ESP32 DevKit V1 and the Arduino Nano V3. In our tests, we will evaluate the performance, power consumption, and latency of the mentioned ciphers to demonstrate the impacts on consumption and performance in these devices.

Keywords: Internet of Things, Low Consumption, Lightweight Cryptografhy, Information Security.

LISTA DE FIGURAS

Figura 1 – Rede IoT 19
Figura 2 – Camadas da Internet das Coisas. 21
Figura 3 – Arquitetura dos Dispositivos IoT 23
Figura 4 – Hierarquia subdivisão da criptografia 28
Figura 5 – Propagação do padrão de atividade (em cinza) em uma única rodada 32
Figura 6 – Rede Feistel 33
Figura 7 – Processo de encriptação da cifra Camellia para chaves de 128 bits 35
Figura 8 – Etapa de Round Feistel da Cifra Simon 37
Figura 9 – Rodada da função Speck 39
Figura 10 – Rodada da função Speck decomposta nos passos utilizando Feistel. 39
Figura 11 – Uma Rodada da XXTEA 41
Figura 12 – Quarter Round Function Salsa20 43
Figura 13 – Quarter Round Function Chacha20 45
Figura 14 - Fluxograma do processo executado para medição do consumo. 53
Figura 15 – Layout de medição 53
Figura 16 – Layout Montado 54
Figura 17 – Utilização de memória ROM 56
Figura 18 – Utilização de memória RAM 57
Figura 19 – Tempo de criptografia em microssegundos 61
Figura 20 – Tempo de descriptografia em microssegundos 61
Figura 21 – Latência de Criptografia por bloco 63
Figura 22 – Latência de Descriptografia por Bloco 63
Figura 23 – Throughput de Criptografia 65
Figura 24 – Throughput de Descriptografia 65
Figura 25 – Custo de Energia por bit no ESP32 68
Figura 26 – Custo de Energia por bit no Arduino Nano 68
Figura 27 – Eficiência na Criptografia 70
Figura 28 – Eficiência na Descriptografia 71
Figura 29 – Consumo Medido na Cifra AES Implementado no ESP-32 73
Figura 30 – Amostra de 5 Segundos do Consumo da Cifra Simon no ESP-32 74
Figura 31 – Amostra de 5 Segundos da Potência Demandada pela Cifra Simon no ESP-32 75
Figura 32 – Amostra de 5 Segundos do Consumo da Hash SHA3-256 no ESP-32 75
Figura 33 – Amostra de 5 Segundos da Potência Demandada pela Hash SHA3-256 no ESP-32 76
Figura 34 – Amostra de 5 Segundos da Potência Demandada pela Cifra Speck no Arduino Nano 76
Figura 35 – Amostra de 5 Segundos da Potência Demandada pela Cifra Speck no Arduino Nano 77
Figura 36 – Amostra de 5 Segundos do Consumo da Hash SHA256 no Arduino Nano 77
Figura 37 – Amostra de 5 Segundos do Consumo da Hash SHA256 no Arduino Nano 78
Figura 38 – Consumo das Cifras e Hashs no ESP-32 79
Figura 39 – Consumo das Cifras e Hashs no Arduino 79
Figura 40 – Consumo Total no Decorrer de 5 Minutos no ESP-32 80
Figura 41 – Consumo Total no Decorrer de 5 Minutos no Arduino 80

LISTA DE TABELAS

Tabela 1 – Criptografias e Hash abordadas 30
Tabela 2 – Parâmetros da Cifra Simon 36
Tabela 3 – Parâmetros da cifra Speck 38
Tabela 4 – Descrição do modulo ESP32 52
Tabela 5 – Descrição do modulo Arduino Nano 52
Tabela 6 – Configuração das Cifras e Hashs 54
Tabela 7 – Taxa de ocupação das memórias 59

LISTA DE SIGLAS E ABREVIATURAS

5G	Quinta geração
AES	*Advanced Encryption Standard*
AES	*Advanced Encryption Standard*
ARPA	*Advanced Research Projects Agency*
ARPANET	*Advanced Research Projects Agency Network*
DES	*Data Encryption Standard*
FIPS PUB	*Federal Information Processing Standards Publication*
IoT	*Internet of Things*
IPv6	*Internet Protocol version 6*
ISO/IEC	*International Organization for Standardisation / International Electrotechnical Commission*
LAN	*Local Area Network*
LTE	*Long-Term Evolution*
NESSIE	*New European Schemes for Signatures, Integrity, and Encryption*
NIST	*National Institute of Standards and Technology*
NIST	*National Institute of Standards and Technology*
NSA	*National Security Agency*
NTT	*Nippon Telegraph and Telephone Corporation*
P&G	*Procter & Gamble*
PC	*Personal Computer*
RFID	*Radio Frequency Identification*
RSA	*Ron Rivest, Adi Shamir e Len Adleman*
RSSF	Rede de Sensores Sem Fio
SHA	*Secure Hash Algorithm*
Wi-Fi	*Wireless Fidelity*
XXTEA	*Corretend Tiny Encryption*
SSL/TLS	*Secure Sockets Layer / Transport Layer Security*
TCP/IP	*Transmission Control Protocol / Internet Protocol*

LISTA DE SIMBOLOS

%	Porcentagem
&	*And* bit a bit
S^n	Deslocamento bit a bit para esquerda
S^{-n}	Deslocamento bit a bit para direita
<<	Deslocamento bit a bit para esquerda
>>	Deslocamento bit a bit para direita
	Concatenação
	Exclusive OR

SUMÁRIO

1. INTRODUÇÃO 13
1.1 Justificativas 15
1.2 Objetivos gerais e específicos 16
2. REVISÃO DA LITERATURA 16
2.1 Internet das Coisas 17
2.1.1 História da Internet das Coisas 18
2.1.2 Arquitetura 21
2.1.3 Dispositivos com Recursos Limitados e IoT 22
2.1.4 Desafios e Futuro 24
2.2 Segurança da Informação 25
2.3 Criptografia e Função *Hash* Leve 27
2.3.1 AES 30
2.3.2 Camellia 32
2.3.3 Simon 35
2.3.4 Speck 38
2.3.5 XXTEA 40
2.3.6 Salsa 20 42
2.3.7 Chacha 20 44
2.3.8 Blake2s 45
2.3.9 SHA – 256 46
2.3.10 SHA3 – 256 48
3. METODOLOGIA 49
4. ANÁLISE E RESULTADOS 55
4.1 Consumo de Memória 55
4.2 Latência de criptografia 60
4.3 *Throughput* 64
4.4 Custo Energético por bit 67
4.5 Eficiência do Algoritmo Criptográfico 69
4.6 Consumo Medido 72
5. CONSIDERAÇÕES FINAIS 81
BIBLIOGRAFIA 84

1. INTRODUÇÃO

A Internet das Coisas (IoT, do inglês *Internet of Things*) representa uma revolução tecnológica que está transformando a maneira como interagimos com o mundo ao nosso redor. Ao conectar dispositivos físicos à Internet, permitindo a troca de dados entre objetos anteriormente isolados, criando um ecossistema interconectado que oferece uma vasta gama de benefícios e oportunidades. Desde casas inteligentes e cidades conectadas até sistemas industriais avançados, a IoT está moldando o futuro de diversos setores. Representando uma das evoluções tecnológicas mais significativas da era digital, prometendo transformar diversos aspectos do cotidiano, a IoT conecta objetos físicos à Internet, permitindo que se comuniquem e colaborem para desempenhar tarefas de maneira mais eficiente e automatizada. Assim desde o momento em que acordamos até a hora de dormir, inúmeros dispositivos conectados estão presentes em nosso cotidiano, proporcionando conveniência, eficiência e uma maior integração entre o mundo físico e digital.

A ideia de conectar objetos físicos à Internet não é completamente nova. O conceito remonta às décadas de 1980 e 1990, quando os primeiros dispositivos com capacidade de comunicação começaram a ser desenvolvidos. Mouha (2021) cita em seu artigo que termo "Internet das Coisas" foi cunhado por Kevin Ashton em uma apresentação à Proctor & Gamble em 1999. Ele vislumbrou um mundo onde os dispositivos poderiam ser identificados e gerenciados automaticamente através da Internet, criando um sistema interconectado capaz de otimizar processos de maneira inédita. Já nos anos 2000, com o avanço tecnológico da comunicação sem fio, miniaturização dos sensores e aumento da capacidade de processamento, que a IoT começou a ganhar tração real. Desde então, a IoT evoluiu rapidamente, impulsionada pelo desenvolvimento de tecnologias como RFID, IPv6, e redes de sensores sem fio.

Atualmente, de acordo com Vailshery (2024) a quantidade de dispositivos IoT ativos no mundo ultrapassa a marca de 17 bilhões e continua crescendo rapidamente. Essa proliferação é impulsionada por avanços em tecnologia de rede, como 5G, e a redução dos custos de hardware. Cada vez mais objetos estão sendo equipados com capacidades de comunicação, permitindo uma integração e automação sem precedentes. Essa tendência está transformando setores inteiros, desde a saúde, onde dispositivos vestíveis monitoram sinais vitais em tempo real, até a agricultura, com sensores que otimizam a irrigação e o uso de fertilizantes.

No entanto, juntamente com os inúmeros benefícios, a IoT também apresenta desafios significativos, especialmente em termos de segurança. Devido à natureza dos dispositivos IoT, que muitas vezes são limitados em termos de recursos computacionais, energia e armazenamento, implementar medidas de segurança robustas pode ser um desafio. Neste contexto, o uso de criptografia leve surge como uma solução crucial para proteger dados e garantir a segurança das comunicações em ambientes IoT. A criptografia leve refere-se a técnicas e algoritmos de criptografia que são projetados para funcionar de maneira eficiente em dispositivos com recursos limitados. Isso é essencial na IoT, onde muitos dispositivos, como sensores e atuadores, possuem capacidade de processamento e energia reduzida. Julian Okello WLiu QAli Siddiqui F et al. (2017), Fotovvat ARahman GVedaei S et al. (2021) e Hatzivasilis GFysarakis KPapaefstathiou I et al. (2018) fornece uma análise detalhada do estado da criptografia leve para a IoT. Os autores exploram diversas técnicas e algoritmos que foram desenvolvidos para atender às necessidades específicas de segurança em ambientes IoT. Oferecendo uma visão abrangente do estado da arte das cifras. Apresentando analises detalhada das características, desempenho e segurança de diversas cifras, além de fornecer recomendações para sua aplicação em dispositivos embarcados.

O artigo é um recurso valioso para pesquisadores e desenvolvedores que trabalham com criptografia leve, oferecendo informações cruciais para a escolha da cifra mais adequada às suas necessidades. Ao destacar o desafio de equilibrar segurança robusta e eficiência de recursos em um ambiente heterogêneo como a IoT, os autores realizam uma análise comparativa do desempenho de diversos algoritmos de criptografia leve aplicados a nós sensores. A avaliação abrange consumo de energia, tempo de execução e uso de memória, fatores críticos para a viabilidade de soluções de segurança em dispositivos IoT. Os resultados demonstram que a escolha do algoritmo ideal depende do tipo de dispositivo e da aplicação específica.

Com o objetivo de analisar o desempenho de cifras em ambientes com recursos limitados, este estudo propõe uma abordagem metodológica abrangente e aprofundada. Serão empregadas diversas técnicas de avaliação, incluindo análises teóricas de complexidade, principalmente, medições práticas em plataformas reais.

As medições de desempenho serão realizadas em duas arquiteturas distintas: microcontroladores de 8 bits, representativos de dispositivos IoT de baixo custo e

consumo reduzido, e microcontroladores de 32 bits, que oferecem maior poder de processamento para aplicações mais exigentes. Esta abordagem permitirá uma análise comparativa detalhada, evidenciando o impacto da arquitetura na eficiência das cifras. Além do desempenho computacional, serão realizadas medições precisas do consumo de energia, um fator crítico em dispositivos alimentados por bateria ou com restrições energéticas. A análise do consumo energético auxiliará na identificação das cifras mais eficientes em termos de energia, contribuindo para o desenvolvimento de soluções de segurança otimizadas para dispositivos IoT. As cifras selecionadas para este estudo abrangem uma variedade de algoritmos leves, projetados especificamente para ambientes com recursos limitados. A análise comparativa dessas cifras, sob diferentes métricas de desempenho e em arquiteturas distintas, fornecerá um panorama abrangente e relevante para a comunidade científica e para desenvolvedores de aplicações IoT. Em suma, no decorrer este estudo visa-se apenas avaliar o desempenho de cifras de bloco simétricas, mas também fornecer *insights* valiosos sobre a escolha da cifra mais adequada para cada cenário de aplicação, considerando as características e restrições específicas de cada dispositivo IoT.

1.1 Justificativas

A criptografia é uma das medidas de segurança da informação mais importante para proteger a privacidade e a segurança dos dados na IoT. A criptografia é um processo que converte dados legíveis em um formato codificado que só pode ser lido com a chave de decodificação correta. Ao usar criptografia, mesmo que um invasor acesse os dados, ele não será capaz de lê-los sem a chave de decodificação.

Com o uso de criptografia em IoT temos:

- Privacidade: Dispositivos IoT são nós sensores que coletam e transmite dados pessoais e confidenciais, podendo portar dados como informações de saúde, dados financeiros, informações de localização entre outro. A criptografia pode proteger esses dados de acesso não autorizado a terceiros.
- Segurança: A criptografia pode proteger dispositivos IoT contra ataques maliciosos, como interceptação de dados, injeção de código ou manipulação de dados.
- Integridade e confiança dos dados: A criptografia garante que os dados não sejam modificados ou adulterados durante a transmissão ou armazenamento, garantindo

que seus dados sejam protegidos e seguros, mantendo assim a integridade e aumentar a confiança dos dados.

Em resumo, a criptografia é uma medida de segurança essencial para proteger os dispositivos IoT e as informações confidenciais que eles contêm. A criptografia garante a privacidade, integridade e segurança aos dados, além de ajudar as organizações a cumprir as regulamentações de segurança de dados.

1.2 Objetivos gerais e específicos

O objetivo geral deste trabalho é demonstrar a possibilidade de utilizar criptografia em dispositivos IoT para garantir segurança e privacidade dos dados.

Objetivos Específicos:

- Implementar criptografias leves em dispositivos com recursos limitados;
- Testar o impacto do desempenho nos dispositivos;
- Comparar o consumo de energia e desempenho entre os métodos criptográficos;
- Comparar o consumo de energia e desempenho entre os dispositivos.

2. REVISÃO DA LITERATURA

Neste capítulo, abordaremos a revisão da literatura, com foco em três tópicos fundamentais: a IoT, a segurança da informação e as criptografias leves. Sendo esses cruciais para o entendimento e desenvolvimento do trabalho apresentado.

Primeiramente, faremos uma breve introdução à Internet das Coisas, discutindo sua definição, evolução histórica, arquiteturas, aplicações e impacto na vida cotidiana e nos negócios. Em seguida, exploraremos a segurança da informação, destacando os principais desafios e ameaças que surgem com a crescente conectividade e troca de dados no ambiente da IoT. Por fim, faremos um resumo sobre as criptografias leves, suas características, vantagens e desvantagens de cada uma das 10 cifras que serão trabalhadas.

Através desta revisão da literatura, pretendemos fornecer uma base sólida de conhecimento para compreender a intersecção entre a Internet das Coisas, segurança da informação e criptografias leves, facilitando a compreensão dos conceitos e aplicações discutidas nos capítulos posteriores deste trabalho.

2.1 Internet das Coisas

A IoT é amplamente reconhecida como uma das tecnologias mais promissoras do século 21. Ela permite a interconexão entre dispositivos, possibilitando a comunicação sem a intervenção humana. E, de acordo com Radouan Ait Mouha (2021), a definição de IoT ainda não é completamente estabelecida ou consensual, o que significa que não há uma definição geral amplamente aceita pela maioria ou pela comunidade global de usuários. A "coisa" na IoT pode ser qualquer dispositivo com sensores incorporados capazes de coletar dados e transmiti-los pela rede sem intervenção manual. Consequentemente, a IoT está em constante amadurecimento e continua a ser um dos conceitos mais novos e populares no mundo da tecnologia da informação.

A Iot está remodelando nosso cotidiano e impulsionando a inovação em diversos setores. Com bilhões de dispositivos conectados em operação, estamos apenas começando a explorar as possibilidades e benefícios que essa tecnologia pode oferecer. A evolução da IoT promete um futuro ainda mais conectado e eficiente, onde a interação entre o mundo físico e digital será cada vez mais harmoniosa e integrada. A capacidade de coletar, analisar e agir sobre dados em tempo real está transformando o mundo em que vivemos, trazendo novas oportunidades e desafios para a sociedade.

Tendo o potencial de transformar a maneira como vivemos, trabalhamos e nos relacionamos uns com os outros. A IoT oferece uma enorme oportunidade para melhorar a eficiência, a produtividade e a segurança através da conexão com setores da sociedade como demonstrado na Figura 1. Alguns exemplos incluem:

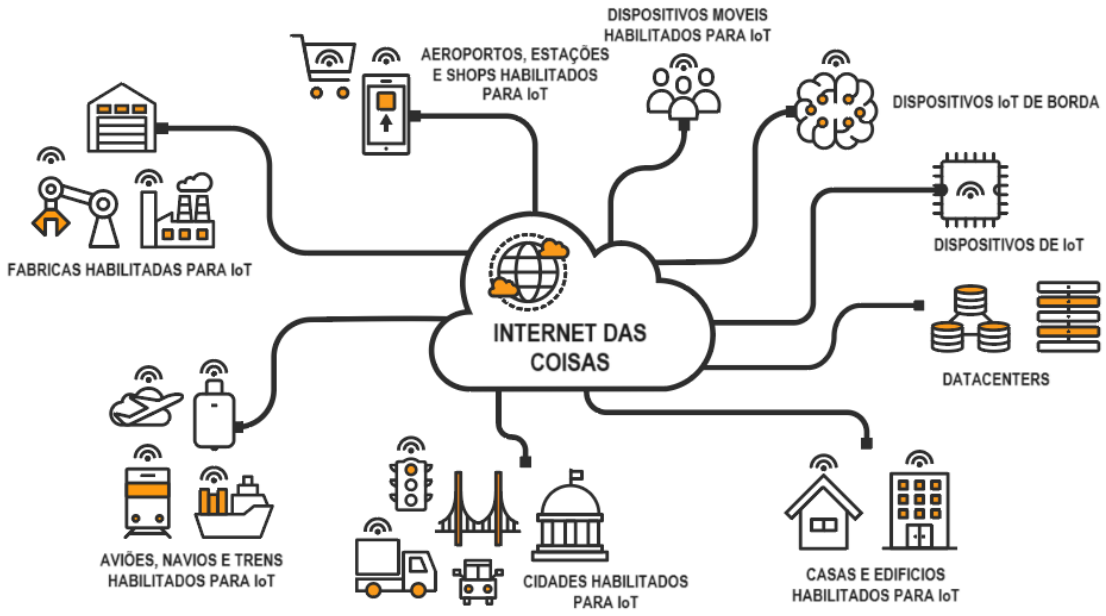
- Cidades inteligentes, como demonstrado por Zanella et al. (2018), podem ajudar a torná-las mais eficientes e sustentáveis. Sensores podem ser usados para monitorar o tráfego, a poluição do ar e a qualidade da água. Isso pode ajudar a identificar áreas problemáticas e tomar medidas para melhorar a qualidade de vida dos moradores.
- Na área da saúde, Rodrigues et al. (2018) destacam que o uso de IoT pode contribuir significativamente para melhorar a qualidade do atendimento médico e reduzir custos. Sensores inteligentes, por exemplo, podem monitorar a saúde dos pacientes em tempo real, alertando os médicos sobre potenciais problemas. Essa abordagem ajuda a prevenir internações desnecessárias e garante que os pacientes recebam o tratamento adequado no momento certo.

- Na Indústria Perera et al. (2015) cita que ajudam a aumentar a eficiência e a produtividade em fábricas e outras instalações industriais. Sensores podem ser usados para monitorar o desempenho de máquinas e alertar os operadores sobre quaisquer problemas. Assim diminuindo ou evitando o número de falhas e consequentemente reduzindo o tempo de inatividade das mesmas.
- Na agricultura, Elijah et al. (2018) demonstra que sensores podem ser usados para monitorar as condições do solo e do clima, ajudando os agricultores a aumentar a sua produtividade e reduzir os custos, pois pode-se determinar os melhores momentos para plantar e colher, reduzindo assim o desperdício e aumentar a eficiência das lavouras.

2.1.1 História da Internet das Coisas

De acordo com Greengard (2023), os primeiros indícios de dispositivos conectados remontam ao início do século XIX, com o desenvolvimento do telégrafo e sua habilidade de enviar informações codificadas a longas distâncias. Mas, apenas no final dos anos 60, que se situam as raízes da IoT. Neste período, um conjunto de pesquisadores começou a investigar maneiras de interligar computadores e sistemas entre si. Sendo criada nesse período a ARPANET pela *Advanced Research Projects Agency* (ARPA) do Departamento de Defesa dos EUA, a ARPANET veio se torna posteriormente a base para a Internet atual. Já no término dos anos 70, houve um movimento por parte de corporações, governos e usuários finais no sentido de conectar PCs e outras máquinas. E durante a década de 80, as Redes Locais (LANs, do inglês *Local Area Network*) emergiram como um meio eficiente e popular.

Figura 1 – Rede IoT



Fonte: Adaptado de Jayakumar et al. (2024)

Teicher (2018), em seu artigo, cita que o primeiro aparelho conectado à ARPANET foi uma máquina de venda de Coca Cola na Universidade Carnegie Mellon. No início da década de 1980, David Nichols, um estudante de pós-graduação do departamento de ciência da computação da universidade, junto a outros dois estudantes, Mike Kazar e Ivor Durham, e um engenheiro pesquisador da universidade, John Zsarnay, começaram a trabalhar no dispositivo que tinha o objetivo de determinar se a máquina de Coca Cola tinha refrigerantes e se esses estavam gelados, tudo isso à distância.

Uma linha de conexão ia até um *gateway* para o computador principal do departamento, que estava conectado à ARPANET, disponibilizando a comunicação e o compartilhamento dos dados e informações em grupos de PCs. Em tempo real, era possível saber se a máquina estava abastecida e se os refrigerantes estavam gelados, a partir do acesso ao dispositivo através de um computador conectado à rede local ou à ARPANET.

Este projeto inovador utilizando a máquina de vendas de Coca Cola, representou um marco significativo na história da Internet e da interconectividade de dispositivos, antecipando conceitos que se tornariam essenciais na era da IoT. Sendo frequentemente citado como um dos primeiros exemplos de dispositivos "inteligentes", que podem ser monitorados e controlados remotamente. Abriu caminho para o desenvolvimento de

tecnologias mais avançadas que hoje fazem parte do nosso cotidiano. Além disso, o projeto destacou a importância da ARPANET como um precursor da Internet moderna.

A capacidade de compartilhar informações e dados em tempo real através de uma rede de computadores foi um avanço revolucionário, mostrando o potencial da interconectividade para transformar diversas áreas, desde a pesquisa acadêmica até a vida cotidiana. O sistema de monitoramento remoto implementado na máquina de vendas da Coca Cola também evidenciou o valor da eficiência e conveniência que a tecnologia poderia trazer, permitindo aos usuários economizar tempo e esforço ao verificar o status da máquina sem precisar se deslocar fisicamente até ela.

Os esforços de David Nichols e seus colegas não apenas demonstraram a viabilidade técnica de conectar dispositivos à ARPANET, mas também inspiraram gerações futuras de cientistas e engenheiros a explorar novas possibilidades na integração de tecnologia e automação.

Em meados da década de 1990, a Internet expandiu essas capacidades a nível mundial, levando pesquisadores e tecnólogos a explorar novas formas de aprimorar a interação entre humanos e máquinas. E, de acordo com Greengard (2023), em 1997, Kevin Ashton, um dos fundadores do *Massachusetts Institute of Technology's Automatic Recognition Lab*, começou a explorar a identificação por rádio frequência (RFID, do inglês *Radio-Frequency Identification*), uma estrutura tecnológica que permitiria a conexão de dispositivos físicos através de microchips e sinais sem fio e foi em um discurso para a Procter & Gamble (P&G), em 1999, que Ashton cunhou o termo "Internet das Coisas". Ashton visualizou um sistema onde objetos do cotidiano poderiam ser equipados com sensores e conectividade para coletar e compartilhar dados via Internet, promovendo uma nova era de automação e eficiência.

De acordo com Oliveira (2021) e Godoi e Araújo (2019), o avanço das tecnologias de comunicação e computação foi crucial para o desenvolvimento da IoT. A miniaturização de sensores, melhoria da eficiência energética e o aumento da capacidade de processamento e armazenamento de dados permitiram a criação de dispositivos mais inteligentes e autônomos. Já a utilização de protocolos de comunicação como o TCP/IP (do inglês, *Transmission Control Protocol / Internet Protocol*), foram fundamentais para permitir a escalabilidade necessária para suportar bilhões de dispositivos conectados.

Reno (2023) e Mancini (2018) destacam os anos 2010, a IoT começou a ganhar notoriedade significativa. Com a combinação discursões sobre padronização internacional, redes moveis e conectividade sem fio facilitou o surgimento de dispositivos conectados em diversos setores, como saúde, agricultura, cidades inteligentes e automação residencial. Assim se tornando parte do nosso cotidiano, Sinha (2023) tem uma expectativa 18,8 bilhões de dispositivos conectados até 2025.

2.1.2 Arquitetura

A arquitetura básica da IoT é composta por 3 camadas que trabalham juntos para conectar, coletar, processar e analisar dados de dispositivos físicos. A estrutura básica da IoT pode ser dividida em: camada de percepção ou de sensores, camada de rede e camada de aplicação como demonstrado na Figura 2.

Figura 2 – Camadas da Internet das Coisas.

Fonte: (Gusmão; Brito, 2015).

Sethi e Sarangi (2017) definem as camadas como:

- Camada de Percepção: É responsável por coletar dados do ambiente físico. Esta camada inclui sensores e atuadores que detectam e medem parâmetros como temperatura, umidade, luz, movimento, entre outros. Os dispositivos nesta camada transformam essas informações em sinais digitais que podem ser transmitidos para outras partes do sistema.
- Camada de Rede: facilita a comunicação entre os dispositivos IoT e outros dispositivos conectados à rede. Garantindo que os dados coletados pelos sensores sejam transmitidos de forma confiável e eficiente para os sistemas de processamento. Ela inclui tecnologias de comunicação sem fio e com fio, como Wi-Fi, Bluetooth, Zigbee, LTE e 5G.

- Camada de Aplicação: Onde os dados coletados são processados, analisados e as informações são usadas para fornecer serviços aos usuários finais. Esta camada inclui interfaces de usuário, aplicativos para Smatphones e sistemas de gestão que permitem monitorar e controlar os dispositivos IoT.

A arquitetura da IoT pode apresentar mais camadas, mas elas podem ser vistas como uma forma expandida, pois é uma estrutura complexa e multifacetada que integra hardware, software e redes para criar um ecossistema de dispositivos interconectados. Cada camada desempenha um papel crucial, desde a coleta de dados até a entrega de serviços finais, tornando a IoT uma tecnologia poderosa e versátil para inúmeras aplicações.

2.1.3 Dispositivos com Recursos Limitados e IoT

Dispositivo com recursos limitado em IoT, refere-se à pequenos equipamentos, como sensores e atuadores, projetados para realizar tarefas específicas com capacidade de processamento, memória e fornecimento de energia restritos. Sendo otimizados para baixo consumo de energia, funcionando com baterias ou fontes de energia alternativas. Os dispositivos frequentemente possuem recursos limitados em comparação com dispositivos tradicionais, como computadores pessoais ou smartphones.

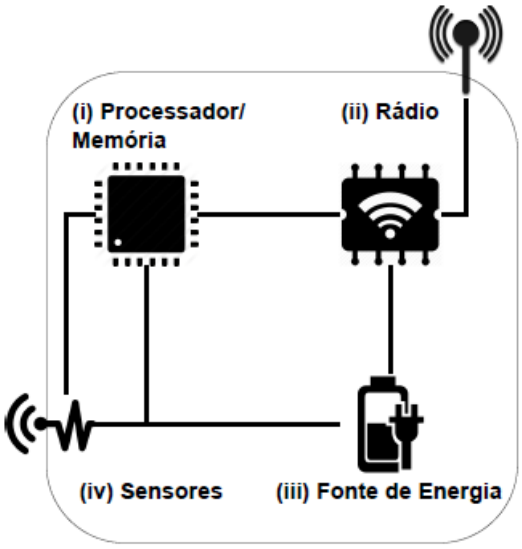
De acordo com Santos et al. (2016) a arquitetura dos dispositivos IoT podendo ser divididas em quatro unidades principais que trabalham de forma interdependente. Sendo elas:

- Processamento / memória: geralmente composta por um micro-controlado, é responsável por processar e armazenar dados capturados pelos sensores.
- Comunicação: uma interface de comunicação que permite a transmição e recepção de dados de outros dispositivos ou sistemas, podendo ser por redes cabeada ou sem fio.
- Fonte de energia: crucial para alimentar o dispositivo garante a operação do dispositivo por longos períodos sem necessidade de intervenção humana. Podendo ser proveniente de baterias, energia solar ou afins.
- Sensores / atuadores: permite o dispositivo interagir com o ambiente físico. Os sensores atuam na captura de dados do entorno, como temperatura, umidade ou

movimento, enquanto os atuadores realizam ações, como ligar ou desligar aparelhos ou ajustar mecanismos.

A Figura 3 ilustra esta arquitetura, destacando a interconexão entre as quatro unidades, permitindo uma visão clara de como os componentes colaboram para o funcionamento do dispositivo inteligente.

Figura 3 – Arquitetura dos Dispositivos IoT



Fonte: (Santos et al., 2016)

Devido ao fato de muitos dispositivos IoT terem o tamanho reduzido, por serem projetados para ocupar o mínimo de espaço possível. Essa miniaturização permite que sejam discretamente instalados em locais estratégicos. Além disso, o tamanho reduzido contribui para a portabilidade e flexibilidade destes, permitindo seu uso em aplicações móveis ou em locais de difícil acesso. Mas essas limitações físicas, também afeta a quantidade de recurso disponível no dispositivo. Assim segundo Mohd et al. (2015) e Thakor et al. (2020) podem incluir:

- A capacidade de processamento limitada.
- Memória (RAM, ROM) que pode limitar a quantidade de dados processados localmente ou armazenados temporariamente.
- Restrições de energia onde alimentados por baterias ou fontes de energia de baixa demanda.
- Conectividade intermitente e resposta em tempo real.

Apesar das limitações, esses dispositivos são essenciais em redes IoT, onde a quantidade de dados gerada é pequena, mas a conectividade e o baixo custo são fatores críticos para a implementação em larga escala.

2.1.4 Desafios e Futuro

Apesar dos avanços, a IoT enfrenta desafios consideráveis, como segurança, privacidade e interoperabilidade. A quantidade massiva de dados gerados por dispositivos IoT requer robustas medidas de segurança para evitar violações e garantir a privacidade dos usuários. A interoperabilidade entre diferentes dispositivos e plataformas também é crucial para o sucesso contínuo da IoT.

Conforme Greengard (2023), há uma crescente integração de sistemas IoT em todos os aspectos da vida humana. Essa expansão abrangente trouxe uma diversidade de tecnologias envolvidas na troca de dados entre dispositivos incorporados, o que, por sua vez, introduziu uma série de complexidades e desafios significativos. Entre esses desafios, destacam-se várias questões cruciais que precisam ser abordadas para garantir o sucesso e a sustentabilidade da IoT.

- A segurança é preocupação primordial. Com bilhões de dispositivos conectados, a quantidade de ataques por cibercriminosos aumentou exponencialmente, tornando essencial o desenvolvimento de robustas medidas de segurança para proteger dados sensíveis.
- A privacidade dos usuários também está em risco, já que a coleta e análise massiva de dados pessoais podem levar a abusos e violações de privacidade.
- A interoperabilidade e a padronização são desafios significativos. A falta de normas universais dificulta a integração e a comunicação eficiente entre dispositivos de diferentes fabricantes. A criação de padrões comuns é essencial para garantir que a IoT possa operar de forma harmoniosa e eficaz.
- Os aspectos legais e regulatórios também são fundamentais. A rápida evolução da IoT muitas vezes supera a capacidade das leis e regulamentos existentes, criando lacunas que podem ser exploradas. Regulamentações claras e abrangentes são necessárias para proteger os direitos dos consumidores e garantir a responsabilidade dos fabricantes e desenvolvedores de IoT, como exemplo a LGPD (Lei Geral de Proteção de Dados Pessoais) descrita pela Lei nº 13.709/2018e em Brasil (2018).

Essas questões complexas e interligadas destacam a necessidade de uma abordagem multidisciplinar para enfrentar os desafios da IoT e maximizar seu potencial positivo. Pois o futuro é promissor, Sinha (2023) prever que 18,8 bilhões de dispositivos estarão conectados até o final de 2024. Espera-se que tecnologias emergentes, como 5G e inteligência artificial, impulsionem ainda mais a evolução da IoT, permitindo novas aplicações e serviços. Aumentando a expectativas de 41,1 bilhões de dispositivos conectados até 2030.

2.2 Segurança da Informação

Um dos principais desafios da segurança da informação na IoT é a enorme variedade de dispositivos conectados. Cada dispositivo, seja ele uma lâmpada inteligente ou um monitor cardíaco, possui sua própria configuração de segurança e, consequentemente, suas próprias vulnerabilidades. Essa diversidade dificulta a criação de uma solução de segurança única que possa proteger todos os dispositivos de forma eficaz.

De acordo com Hintzbergen *et al.* (2018), Antes de definirmos uma estratégia de segurança, é fundamental compreender o que exatamente estamos protegendo e contra quais ameaças. A metodologia que utilizamos para obter esse conhecimento é conhecida como análise de risco. Dessa forma, a segurança da informação é garantida pela implementação de um conjunto apropriado de controles, que envolvem políticas, processos, procedimentos, estruturas organizacionais e funções tecnológicas, tanto de software quanto de hardware. Esses controles trabalham de forma integrada para diminuir riscos e garantir a proteção dos ativos mais críticos da organização.

Assim Hintzbergen *et al.* (2018) apresenta o Código de Práticas para segurança da informação da ISO 27002:2013, que menciona a importância de:

- Entender os requisitos de segurança da informação da organização e reconhecer a importância de definir políticas e metas claras para assegurar a proteção das informações.
- Implementar e operar controles que permitam gerenciar os riscos de segurança da informação da organização, alinhando-os aos riscos gerais de negócios para garantir uma abordagem integrada e eficiente na redução de ameaças.
- Monitorar e revisar o desempenho e a eficácia do Sistema de Gerenciamento de Segurança da Informação.
- Melhoria contínua baseada em medições objetivas.

Nesse contexto surge a tríade da CID (Confidencialidade, Integridade e Disponibilidade). De acordo com (Holdsworth e Kosinski, 2023), a tríade da CID "tem como objetivo orientar as organizações na escolha de tecnologias, políticas e práticas para proteger seus sistemas de informação".

Holdsworth e Kosinski (2023), Hintzbergen *et al.* (2018) e Gonçalves (2015) apresentam a tríade da CID como pilares da segurança da informação, pois cada um desempenha um papel essencial na proteção de dados e sistemas. A seguir uma breve explicação de cada conceito:

- Confidencialidade: Refere-se à proteção das informações contra o acesso não autorizado. Apenas pessoas ou sistemas autorizados devem ter acesso aos dados.
- Integridade: Assegura que os dados não sejam alterados, corrompidos de maneira não autorizada ou ao longo do tempo. Isso garante que as informações permaneçam precisas e completas durante todo o seu ciclo de vida.
- Disponibilidade: Garante que as informações e sistemas estejam acessíveis para os usuários autorizados sempre que necessário. Isso implica em manter a continuidade de operações e evitar interrupções ou falhas.

Esses três elementos trabalham em conjunto para garantir a segurança total da informação dentro de uma organização, ajudando a proteger os dados contra ameaças e garantir o funcionamento contínuo dos sistemas.

Nesse quesito a criptografia desempenha um papel crucial na segurança da informação na IoT. Ela garante que os dados transmitidos entre dispositivos sejam protegidos contra interceptações e acessos não autorizado dos dados, assim de acordo com Hintzbergen *et al.* (2018), a criptografia é utilizada para alcançar a confidencialidade. Podendo atuar protegendo os dados armazenados nos dispositivos IoT e durante a transmição entre dispositivos e servidores.

De acordo com Dhanda e et all. (2020), a confidencialidade e a integridade das informações podem ser alcançadas através da utilização de criptografia com o auxílio de funções *Hash*. A função *hash* atua na garantir a integridade dos dados. Pois transforma uma entrada (arquivo ou mensagem) em uma sequência fixa de caracteres. Essa sequência é única para cada conjunto de dados, e até mesmo uma pequena alteração na entrada resultará em um valor *hash* diferente.

A utilização de criptografia e funções *hash* são elementos essenciais para a segurança da informação. Embora existam desafios consideráveis, a adoção de práticas de segurança robusta pode reduzir significativamente os riscos, tornando a utilização de dispositivos Iot confiáveis.

Mas os métodos criptográficos tradicionais exigem grande alocação de recursos. Por outro lado, os dispositivos IoT são caracterizados por suas limitações. Isso descreve claramente a necessidade de utilizar algoritmos de criptografias leves que serão apresentados nos capítulos seguintes.

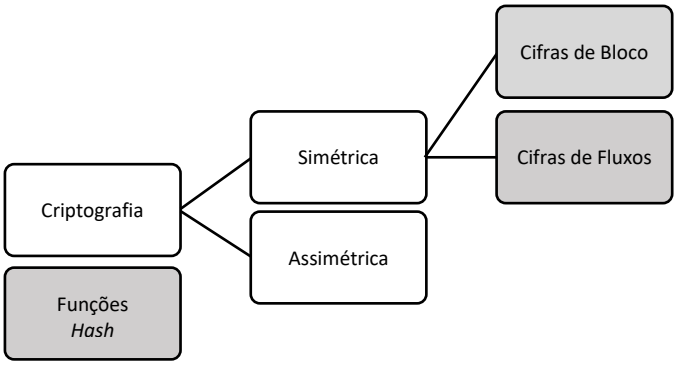
2.3 Criptografia e Função *Hash* Leve

A IoT vem se expandindo exponencialmente e a cada ano surgem novos gadgets com as mais variadas propostas, como apontado por Sinha (2023). No entanto, com esta expansão, é importante reconhecer os desafios de segurança que surgem quanto mais dispositivos se conectam à Internet.

Nesse quesito garantir a proteção dos dados durante a transmissão entre dispositivos, é um dos principais desafios devido à variedade de dispositivos que estão conectados, cada um com sua própria configuração de segurança. As implementações de medidas de segurança robustas podem ajudar a minimizar o risco, e nesse cenário as criptografias e funções *hash* de baixo custo estão surgindo como uma solução acessível e eficaz para garantir a confiabilidade e integridade durante a comunicação desses dispositivos.

A criptografia é um conjunto de recursos tecnológicos aplicados para proteger e evitar que dados armazenados ou transmitidos em sistemas sejam acessados por terceiros. Ela é essencial para garantir a segurança de informações. Os principais tipos de criptografia demonstrados na Figura 4.

Figura 4 – Hierarquia subdivisão da criptografia



Fonte: Própria autoria.

De acordo com Dutta et al. (2019) a criptografia de chave simétrica, também denominada como chave secreta ou criptografia de chave compartilhada, envolve o compartilhamento de uma chave única entre remetente e destinatário através de comunicação sigilosa, permitindo tanto a criptografia quanto a descriptografia dos dados com a mesma chave. A criptografia simétrica é geralmente mais rápida e eficiente do que a criptografia assimétrica, mas requer um processo seguro para compartilhar a chave entre as partes envolvidas. Ela subdividida em:

- Cifras de Bloco: É um tipo de criptografia em que um bloco de dados é transformado em outro bloco de mesmo tamanho por meio de operações de substituição ou permutação, com base em uma chave secreta.
- Cifras de Fluxo: São algoritmos de criptografia que operam em tempo real, criptografando e descriptografando dados à medida que são transmitidos.

A criptografia de chave assimétrica, também conhecida como criptografia de chave pública, conforme mencionado por Dutta et al. (2019), requer um par de chaves, pública e privada, onde a chave pública é compartilhada livremente e é usada para criptografar os dados pelo remetente e uma vez criptografados, os dados só podem ser descriptografados pela chave privada correspondente, que é mantida em sigilo pelo destinatário. Isso significa que qualquer pessoa pode criptografar os dados usando a chave pública, mas apenas o destinatário com acesso à chave privada correspondente pode descriptografá-los com sucesso. A segurança da criptografia assimétrica baseia-se na complexidade matemática de certos problemas, como a fatoração de números grandes em fatores primos ou o problema do logaritmo discreto.

Já as funções *hash* São algoritmos matemáticos que transformam dados de entrada de comprimento variável em uma sequência de bits de tamanho fixo, geralmente chamada de resumo ou *hash*. De acordo com Dhanda, Singh e Jindal (2020) as funções *hash* leves são outra maneira de fornecer segurança. Criando um resumo de tamanho fixo a partir de uma mensagem de tamanho arbitrário. Este resumo é usado é utilizado para garantir a integridade dos dados transmitidos.

Mas a maior parte dos dispositivos IoT possuem restrições significativas de energia, memória e poder de processamento, como citado no Tópico 2.1.3, o que torna a implementação de técnicas de criptografia tradicionais desafiadora ou impraticável assim como mencionado por Hatzivasilis et al. (2018). Portanto, a criptografia leve busca encontrar soluções que ofereçam um equilíbrio entre segurança e eficiência, adaptando os algoritmos criptográficos para atender às restrições desses dispositivos. As principais características da criptografia leve segundo Thakor et al.,(2020) incluem: Simplicidade, consumo de energia reduzido, tamanho compacto, velocidade rápida de processamento e segurança adequada.

De acordo com Mohd et al. (2015), alguns pesquisadores adotaram uma abordagem mais quantitativa para definir cifras leves. Por exemplo, para classificar uma cifra como leve, eles sugeriram as seguintes propriedades:

- Tamanho do Bloco: Uma cifra leve geralmente possui um tamanho de bloco pequeno, como 32, 48 ou 64 bits, em comparação com cifras convencionais que têm tamanhos de bloco maiores, como 128 ou mais bits.
- Tamanho da Chave: O tamanho da chave também tende a ser menor.
- Operações e Escalonamento de Chaves: Podem simplificar o escalonamento de chaves e empregar operações elementares com um maior número de rodadas.

Neste trabalho, exploraremos exclusivamente a criptografia simétrica, uma técnica que utiliza uma única chave para cifrar e decifrar informações. E algoritmos de funções *hash*. Dentre os de criptografia simétrica, nos concentraremos nesses 10. Citados na tabela 1

Tabela 1 – Criptografias e *Hash* abordadas

Tipo	Criptografia
BLOCOS	AES
	Camellia
	Simon
	Speck
	XXTEA
FLUXO	Chacha20
	Salsa20
HASH	Blake - S
	SHA256
	SHA3 - 256

Nas próximas seções, serão apresentados detalhes sobre os principais algoritmos.

2.3.1 AES

O Padrão de Criptografia Avançada (AES, do inglês *Advanced Encryption Standard*) é um padrão de criptografia simétrica amplamente utilizado para proteção de dados. Desenvolvido por Daemen e Rijmen (2002), o algoritmo criptográfico descrito pelo relatório AES Proposal: Rijndael (Daemen e Rijmen, 1999), foi escolhido pelo Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST, do inglês *National Institute of Standards and Tecnology*) em 2001 para se torna o AES e sucessor do DES (do inglês, *Data Encryption Standard*), devido à necessidade crescente de um algoritmo de criptografia mais robusto e seguro. Pois nos anos 1990, o DES, que havia sido o padrão de criptografia por mais de duas décadas, começou a mostrar vulnerabilidades devido ao aumento do poder computacional e ao surgimento de novas técnicas de criptoanálise. Em resposta a essas ameaças, o NIST lançou um concurso para encontrar um substituto. Entre várias propostas, o Rijndael, destacou-se por sua eficiência, segurança e flexibilidade.

De acordo com EVANS e BROWN (2001) o AES é baseado em uma rede de substituição-permutação, que combina operações de substituição e permutação para criar uma forte criptografia. Ele opera em blocos de 128 bits e utiliza chaves de 128, 192 ou 256 bits. A estrutura do AES é composta por uma série de etapas de transformação aplicadas repetidamente, conhecidas como rodadas. O número de rodadas varia de acordo com o tamanho da chave: 10 rodadas para chaves de 128 bits, 12 para chaves de 192 bits e 14 para chaves de 256 bits.

Cada rodada do AES consiste em quatro etapas principais, como demonstrado na Figura 5:

- SubBytes: Aplicação de uma substituição não-linear a cada byte do bloco, utilizando uma tabela de substituição (S-box).
- ShiftRows: Deslocamento cíclico das linhas da matriz de dados, que ajuda a difundir a influência de cada byte pelos blocos.
- MixColumns: Combinação linear das colunas da matriz, misturando ainda mais os dados.
- AddRoundKey: Adição da chave de rodada ao bloco de dados por meio de uma operação XOR.

Sendo que a primeira rodada exclui a etapa MixColumns, enquanto a última rodada exclui a etapa AddRoundKey. Essa estrutura contribui para a alta segurança do AES, tornando extremamente difícil para os criptoanalistas revertê-lo sem a chave correta.

Segundo Naseer et al. (2018) e Daemen e Rijmen (2002), uma S-Box (*Substitution Box*) é um componente utilizado em muitos algoritmos de criptografia simétrica. Ela realiza a operação de substituição não linear, transformando uma entrada de bits em uma saída de bits de tamanho fixo, de acordo com uma tabela predefinida.

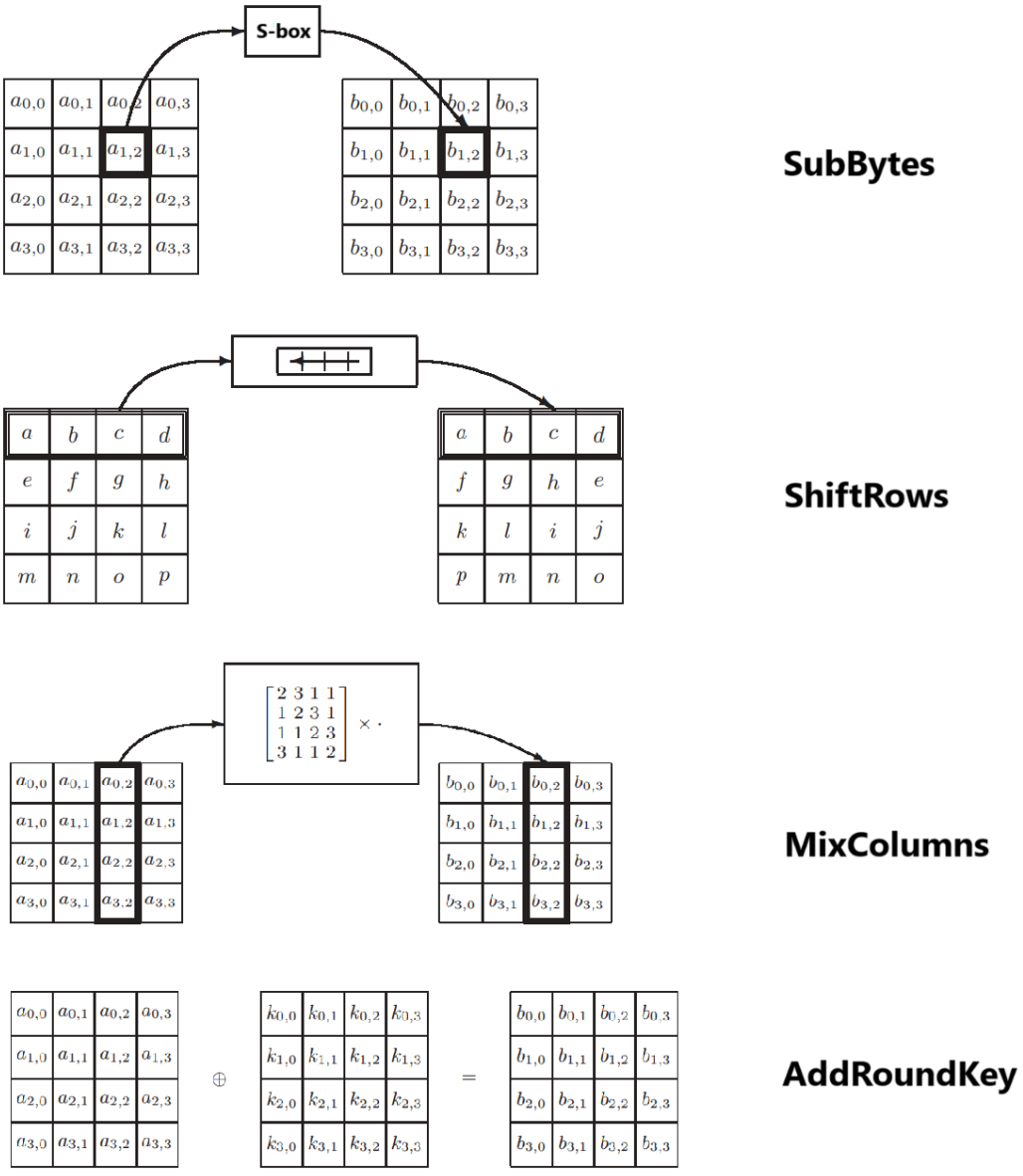
A função principal da S-Box é introduzir confusão no processo de criptografia, o que significa tornar a relação entre a chave de criptografia e o texto cifrado o mais complexa e não linear possível. Isso impede que padrões claros surjam nos dados criptografados, dificultando a quebra da criptografia.

Daemen e Rijmen (2002) também demonstram que AES é amplamente considerado seguro contra todos os tipos conhecidos de ataques. Sua segurança foi extensivamente analisada por criptógrafos de todo o mundo. A eficiência do AES é outra vantagem significativa; ele é projetado para ser eficaz tanto em hardware quanto em software, o que o torna ideal para uma ampla gama de aplicações, desde dispositivos móveis até grandes servidores.

Desde a sua adoção, o AES se tornou o padrão de criptografia simétrica mais utilizado no mundo. Ele é empregado em diversas aplicações, como a proteção de dados em trânsito e em repouso, criptografia de discos, e comunicações seguras em redes. Além

disso, a flexibilidade do AES permite que ele seja integrado em várias tecnologias emergentes, como a Internet das Coisas (IoT) e a computação em nuvem.

Figura 5 – Propagação do padrão de atividade (em cinza) em uma única rodada



Fonte: (DAEMEN e RIJMEN, 2002)

2.3.2 Camellia

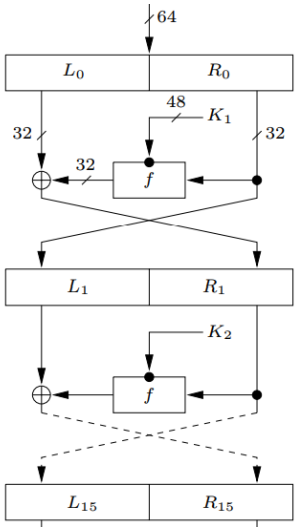
A criptografia Camellia é um algoritmo de criptografia simétrica desenvolvido pela Mitsubishi Electric e pela *Nippon Telegraph and Telephone Corporation* (NTT) em 2000. É conhecido por sua alta segurança e eficiência, sendo amplamente utilizado em

diversas aplicações de segurança da informação. Sendo projetado para ser uma alternativa ao AES, oferecendo um nível comparável de segurança e desempenho, e como citado por RASHIDI (2021) ele é reconhecido por organizações como a *International Organization for Standardization / International Electrotechnical Commission* (ISO/IEC) e o *New European Schemes for Signatures, Integrity, and Encryption* (NESSIE).

De acordo com Aoki et al. (2001), Camellia é uma criptografia de blocos com tamanho de 128 bits, suportando chaves de 128, 192 ou 256 bits, semelhante ao AES. O algoritmo foi projetado com um forte foco em segurança, eficiência e flexibilidade, podendo ser implementado de maneira eficiente tanto em software quanto em hardware. Sua estrutura é baseada em uma rede de Feistel modificada, que consiste em uma série de rodadas, onde cada rodada transforma parcialmente o bloco de dados. O número de rodadas varia conforme o tamanho da chave, sendo: 18 rodadas para chaves de 128 bits e 24 rodadas para chaves de 192 ou 256 bits.

De acordo com Menezes, Oorschot e Vanstone (1996), a rede Feistel é uma estrutura usada na construção de algoritmos de cifra de bloco. Ela divide o bloco de entrada em duas metades (L_0 e R_0) e, em cada rodada, uma metade é transformada por meio de uma função de criptografia (f) derivada da subchave (k_1) e, em seguida, combinada com a outra metade usando uma operação XOR. As duas metades são trocadas a cada rodada. Como demonstrado na Figura 6. Essa estrutura tem a vantagem de permitir que o processo de decifração seja realizado de maneira semelhante à cifração, invertendo a ordem das rodadas e reutilizando as mesmas operações.

Figura 6 – Rede Feistel



Fonte: (Menezes, Oorschot e Vanstone, 1996)

Segundo Aoki et al. (2001), cada rodada do Camellia inclui uma série de operações que combinam substituição, permutação e mistura de dados, como demonstrada na Figura 7. As principais etapas de uma rodada do Camellia são:

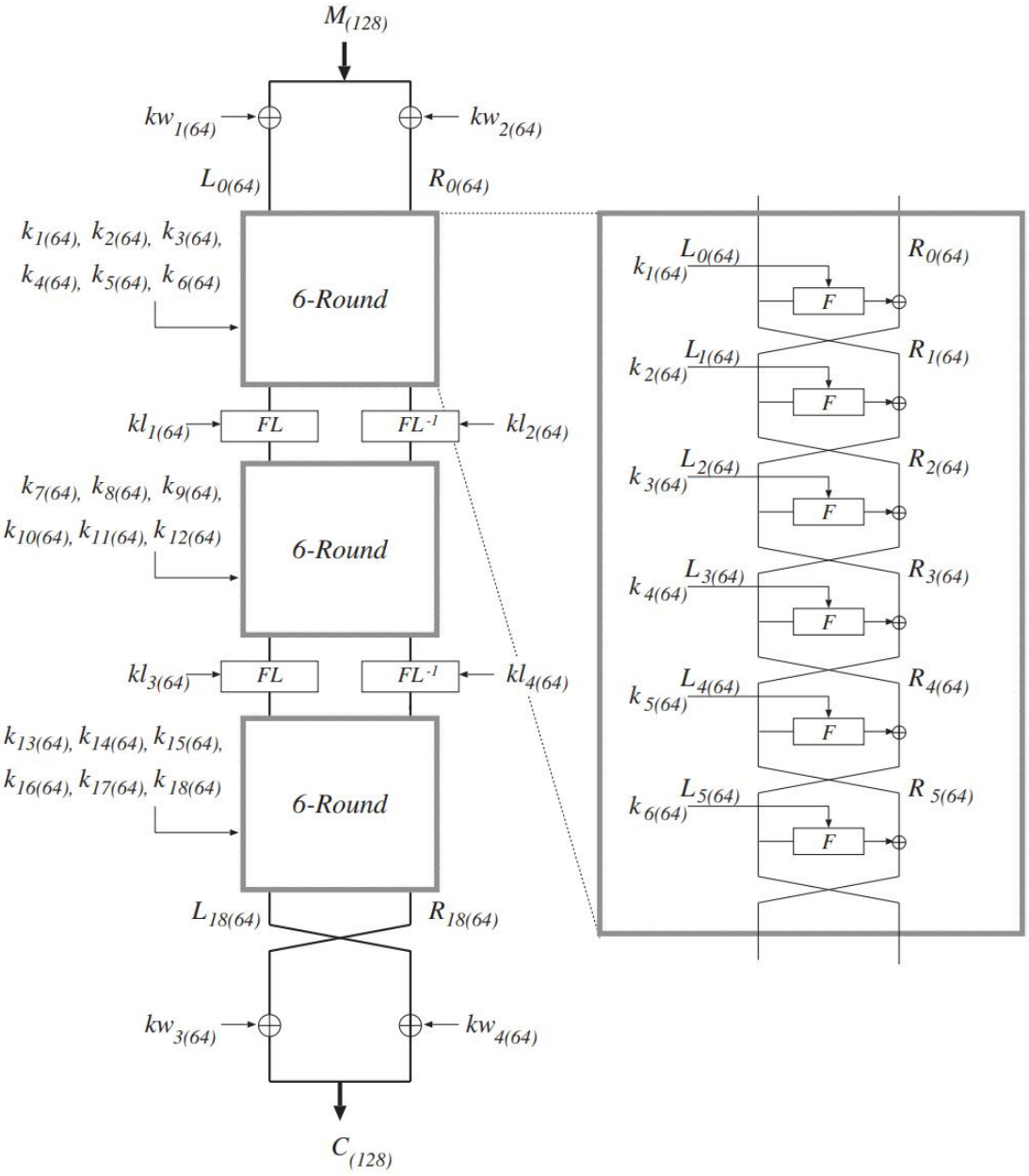
- Função F: Esta é uma função complexa que mistura os dados de entrada usando uma combinação de operações lógicas e matemáticas. Ela utiliza duas tabelas de substituição (S-boxes), uma função logica (FL) e uma permutação chaveada.
- S-boxes: As tabelas de substituição são usadas para substituir partes dos dados de forma não-linear, aumentando a difusão dos dados e dificultando a criptoanálise.
- Permutação de Feistel: A estrutura de Feistel divide o bloco de dados (M) em duas metades (L_0 e R_0) e processa uma metade com a função F, depois combina os resultados com a outra metade usando uma operação XOR.
- Função logica (FL e FL^{-1}): Essas funções são aplicadas para aumentar a difusão dos dados dentro do bloco, misturando os bits de forma linear a cada 6 rodadas.

O termo criptoanálise refere-se ao estudo e a prática de quebrar sistemas de criptografia para recuperar informações sem o uso da chave apropriada. Ela envolve a exploração de fraquezas no algoritmo ou na sua implementação, como ataque de força bruta, analise diferencial, entre outro. Com o objetivo de decifrar dados protegidos, Daemen e Rijmen (2002) faz um breve relato sobre cada método de criptoánalise.

Aoki et al. (2001) demonstra que a cifra Camellia foi projetada para resistir a vários tipos de ataques criptográficos conhecidos, como ataques de força bruta, criptoanálise diferencial e criptoanálise linear. Passando por extensivas análises de segurança por criptógrafos independentes e organizações de padronização.

Uma das principais vantagens do Camellia é sua eficiência tanto em software quanto em hardware. O design do algoritmo permite implementações rápidas e de baixo consumo de recursos, o que o torna adequado para uma ampla gama de dispositivos, desde sistemas embarcados e dispositivos móveis até servidores e aplicações de rede. A criptografia representa um marco significativo na segurança da informação, oferecendo uma combinação robusta de segurança, eficiência e flexibilidade. O sendo uma alternativa viável ao AES, sendo amplamente reconhecido e utilizado em várias aplicações e padrões de segurança em todo o mundo.

Figura 7 – Processo de encriptação da cifra Camellia para chaves de 128 bits



Fonte: (AOKI et al., 2001)

2.3.3 Simon

A cifra Simon é uma criptografia de bloco leve desenvolvida pela *Natinnal Security Agency* (NSA) dos Estados Unidos em 2013. Juntamente com a cifra Speck, Simon foi projetada para oferecer segurança robusta e eficiência em ambientes com recursos limitados, como dispositivos da IoT e sistemas embarcados.

De acordo com Beaulieu et al. (2013) a cifra Simon usa uma estrutura de rede de Feistel modificada. Ela foi projetada para ser simples e eficiente em hardware e software.

Oferecendo diversas variantes com diferentes tamanhos de bloco, chave e número de rodadas, apresentados na tabela 2. O design é relativamente simples, facilitando a análise de segurança e a implementação a diferentes requisitos de segurança e desempenho.

Tabela 2 – Parâmetros da Cifra Simon.

Bloco Tamanho (2n)	**Chave Tamanho (mn)**	**Tamanho *Word* (n)**	**Tamanho *Key word* (m)**	**Const seq**	**Rodadas (T)**
32	64	16	4	z_0	32
48	72	24	3	z_0	36
	96		4	z_1	36
64	96	32	3	z_2	42
	128		4	z_3	44
96	96	48	2	z_2	52
	144		3	z_3	54
128	128	64	2	z_2	68
	192		3	z_3	69
	256		4	z_4	72

Fonte: (Beaulieu et al., 2013)

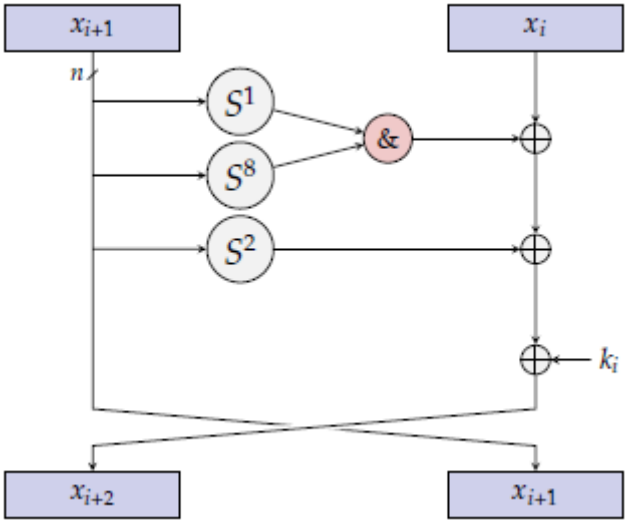
Onde de acordo com Beaulieu et al. (2013), n é o tamanho da subdivisão do bloco de dados, conhecida como *word* em bit, podendo assumir os valores de 16, 24, 32, 48 ou 64. Já m é a subdivisão da chave em *words* de tamanho 2, 3 ou 4. A constante sequencial (z) existe com o propósito de eliminar propriedades de deslizamento e simetrias de deslocamento circular. Biryukov e Wagner (1999) explica que isso acontece quando a estrutura da cifra permanece similar ou se repete ao longo das rodadas de cifração, e quando um padrão ou uma sequência de bits pode ser deslocada circularmente e ainda assim resultar em um valor equivalente. De fato, a criptográfica Simon 5 diferente sequencias de mesmo tamanho de bloco definindo: z_0, z_1, z_2, z_3, z_4. Cada uma dessas sequências é composta por 31 repetições.

Beaulieu et al. (2013) explica que a cifra Simon utiliza operações básicas que são fáceis de implementar em hardware e software, como:

- Deslocamento (S): Cada metade do bloco de dados é deslocada ciclicamente para a esquerda quando o expoente é positivo ou direita, quando negativo.
- AND bit a bit (&): Realiza uma operação AND bit a bit entre duas metades do bloco.
- Exclusive OR (XOR): Aplica a operação XOR para misturar os dados com a subchave da rodada.

Essas operações são iteradas por várias rodadas, conforme determinado pelo tamanho do bloco e da chave definidos na Tabela 2. A simplicidade dessas operações, é demonstrada na Figura 8, representando 1 *round* da criptografia. Devido à essa estrutura simples, Simon pode ser implementada em hardwares, utilizando poucos recursos de área e energia. Em software, a cifra também é eficiente, sendo adequada para microcontroladores e dispositivos com capacidades limitadas.

Figura 8 – Etapa de *Round* Feistel da Cifra Simon



Fonte: (Beaulieu et al., 2013)

A segurança da cifra também foi avaliada por Beaulieu et al. (2013) em relação a diversos tipos de ataques criptográficos, incluindo ataques diferenciais, lineares e de força bruta. A cifra foi projetada para resistir a esses ataques, utilizando uma estrutura que maximiza a difusão dos dados em cada rodada, combina operações não lineares e inclui variantes com chaves mais longas que oferecem maior resistência contra ataques de força bruta.

Assim sendo uma contribuição significativa da NSA para a criptografia leve, atendendo às necessidades de segurança de dispositivos IoT e sistemas embarcados. Sua combinação de simplicidade, eficiência e robustez a torna uma escolha ótima para ambientes com recursos limitados.

2.3.4 Speck

A Criptografia Speck é uma família algoritmos de criptografia de bloco leves desenvolvida pela NSA em 2013. Projetada para dispositivos com recursos limitados, como sistemas embarcados e sensores de IoT, a Speck se destaca por sua eficiência e flexibilidade, adaptando-se a diferentes tamanhos de bloco e chave.

A Speck oferece diversas variantes, cada uma com diferentes tamanhos de bloco e chave, para atender às necessidades de diferentes aplicações. As variantes estão apresentadas na tabela 3:

Tabela 3 – Parâmetros da cifra Speck

Bloco Tamanho (2n)	**Chave Tamanho (mn)**	**Tamanho *Word* (n)**	**Tamanho *Key word* (m)**	**Rot (α)**	**Rot (β)**	**Rodadas (T)**
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3			26
	128		4			27
96	96	48	2			28
	144		3			29
128	128	64	2			32
	192		3			33
	256		4			34

Fonte: (BEAULIEU et al., 2013)

Onde Rot α e Rot β, representam o número de rotações respetivamente para o sentido esquerdo e sentido direito, de acordo com o tamanho do bloco de dados e chave criptografica.

Beaulieu et al. (2013) explica que semelhante a cifra Simon, a cifra Speck utiliza operações básicas, como:

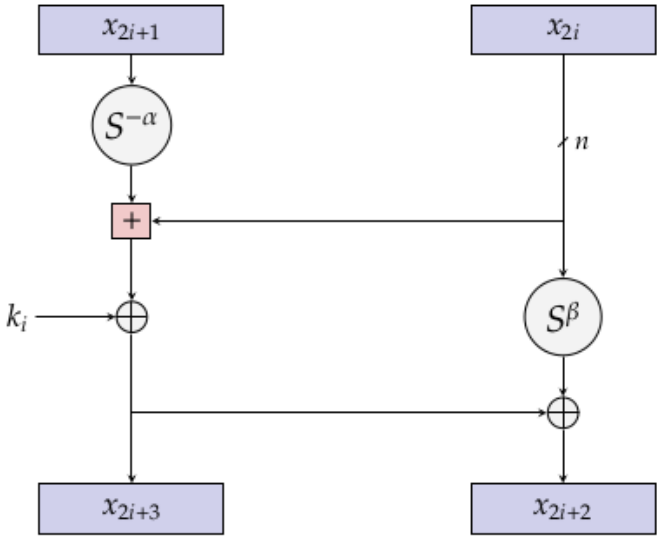
- XOR bit a bit,
- Adição módulo 2n, +, AND,
- Deslocamentos circulares para a esquerda e para a direita, S^j e S^{-j}, respectivamente, por j bits.

De acordo com Beaulieu et al. (2013), a cifra Speck utiliza uma estrutura de Feistel. Cada rodada consiste em operações de adição modular, rotação de bits e XOR, aplicadas às metades do bloco, como demonstrado nas Figuras 9 e 10. O número de

rodadas varia de acordo com o tamanho do bloco de dados e chave criptográfica, como demonstrados na tabela 3 garantindo assim um nível de segurança adequado a aplicação.

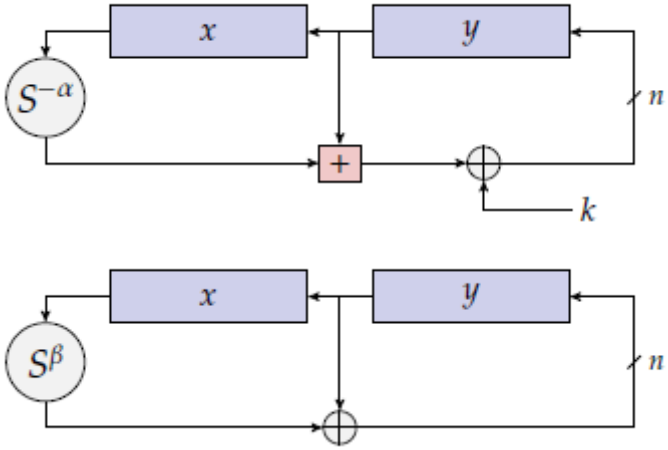
Já a chave criptográfica é expandida em uma sequência de subchaves, que são utilizadas em cada rodada da cifra, sendo projetado para ser resistente a ataques criptoanalíticos.

Figura 9 – Rodada da função Speck



Fonte: (Beaulieu et al., 2013)

Figura 10 – Rodada da função Speck decomposta nos passos utilizando Feistel.



Fonte: (Beaulieu et al., 2013)

Devido essa variedade de tamanhos de bloco e chave é possível adaptar a cifra a diferentes requisitos de segurança e desempenho. Assim projetada para ser eficiente em dispositivos com recursos limitados, como dispositivos IoT, apresenta um bom desempenho em software e hardware, tornando-a adequada para diversas plataformas.

A Speck não é um padrão amplamente adotado, o que pode limitar sua interoperabilidade com outros sistemas, mesmo sendo adequada para aplicações que exigem criptografia leve e eficiente, como a IoT, sistemas embarcados ou comunicação segura.

Mesmo sendo uma cifra de bloco promissora para aplicações que exigem criptografia leve e eficiente com sua flexibilidade e desempenho que a tornam uma opção atraente para dispositivos com recursos limitados. No entanto, é importante estar ciente das preocupações sobre sua segurança e da falta de padronização antes de implementá-la em aplicações críticas.

2.3.5 XXTEA

A cifra XXTEA (*Correction to* XTEA), foi desenvolvida como uma extensão e correção do *Tiny Encryption Algorithm* (TEA). A história de sua criação é uma evolução do esforço contínuo dos criptógrafos para criar algoritmos de criptografia simétrica que sejam ao mesmo tempo seguros e eficientes.

O TEA foi desenvolvido por Wheeler e Needham (1995) no *Computer Laboratory* da Universidade de Cambridge em 1994. O objetivo do TEA era criar um algoritmo de cifra de bloco que fosse simples de implementar, eficiente e adequado para dispositivos com recursos limitados. Ganhando popularidade devido à sua simplicidade e eficiência. No entanto, logo se descobriu que o TEA tinha algumas vulnerabilidades significativas, como relatado por Saarinen (1998), incluindo a suscetibilidade a ataques de criptoanálise diferencial.

Para abordar essas vulnerabilidades, Needham e Wheeler (1997) desenvolveram uma versão aprimorada do TEA chamada XTEA (*Extended* TEA). XTEA foi introduzido em 1997 e apresentava algumas modificações na estrutura do TEA original para aumentar a segurança. Essas modificações incluíam a alteração da função de mistura usada nas rodadas do algoritmo, o que ajudou a mitigar alguns dos ataques conhecidos contra o

TEA. Embora o XTEA fosse mais seguro que seu predecessor, ainda apresentava algumas fraquezas, especialmente em relação à segurança de chaves relacionadas.

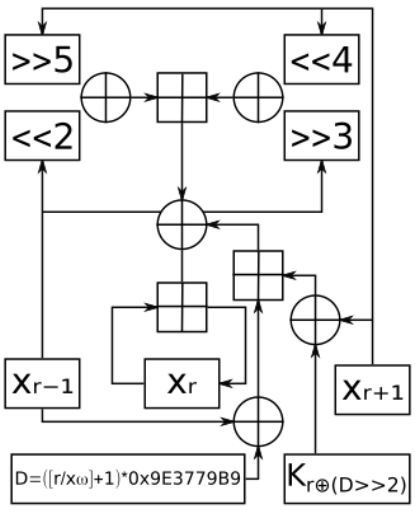
Como resultado, Wheeler e Needham (1998) continuaram a buscar melhorias. Assim nasceu a XXTEA em 1998, com o objetivo de corrigir vulnerabilidades presentes no algoritmo TEA e proporcionar uma cifragem mais confiavel e eficiente.

Sendo uma cifra de bloco que opera em dados equiparados ao TEA de tamanho variável, assim como dito por Wheeler e Needham (1995), tem *Words* de 32 bits, blocos de dois ou quatro *Words* e utilização de uma chave de 128 bits. E de acordo Wheeler e Needham (1998), a estrutura da XXTEA envolve várias rodadas de operações matemáticas simples, como adição, subtração, XOR e deslocamentos, para garantir a difusão e a confusão dos dados cifrados.

A seguir estão os passos gerais envolvidos na criptografia de dados com XXTEA e demonstrado na Figura 11:

- Inicialização: O bloco de dados é dividido em sub-blocos de 32 bits cada, e a chave de 128 bits é dividida em quatro sub-chaves de 32 bits.
- Rodadas: Cada sub-bloco é processado por uma série de operações que incluem: Adição, operação XOR e deslocamento.
- Finalização: Após um número suficiente de rodadas, o bloco de dados cifrado é produzido.

Figura 11 – Uma Rodada da XXTEA



(Wikipedia Contributors, 2024)

Como narrado por Wheeler e Needham (1998) a cifra XXTEA uma versão corrigida e mais segura do TEA, abordando várias fraquezas do algoritmo original. A estrutura de rodadas variáveis e as operações não lineares ajudam a proteger contra ataques de criptoanálise diferencial e de chave relacionada.

A cifra XXTEA é relativamente simples de implementar em software e hardware, graças ao uso de operações aritméticas básicas. E com a capacidade de operar em blocos de dados de tamanho variável e utilizar chaves de 128 bits, a cifra se destaca como uma escolha robusta para criptografia simétrica em diversos contextos. Como demonstrado por Yarrkov (2010), sua resistência aprimorada contra ataques conhecidos reforça sua utilidade como uma ferramenta criptográfica confiável. Isso o torna ideal para uma variedade de aplicações, incluindo a criptografia de dados em IoT, sistemas embarcados, comunicações seguras e proteção de arquivos.

2.3.6 Salsa 20

A cifra Salsa20 é um algoritmo de criptografia simétrica pertencente à família das cifras de fluxo, desenvolvido por Daniel J. Bernstein em 2005. Projetada para ser rápida, eficiente, destacando por sua simplicidade e desempenho em uma ampla gama de plataformas, desde dispositivos com recursos limitados até servidores de alta performance.

A criação de Salsa20 foi motivada pela necessidade de um algoritmo de cifra de fluxo que pudesse operar eficientemente em hardware e software, fornecendo alta velocidade de cifragem e resistência contra ataques criptográficos. A simplicidade do design de Salsa20 também facilita sua implementação e análise de segurança.

De acordo Bernstein (2007), Salsa20 é uma cifra de fluxo que opera em blocos de 64 bytes ou 512 bits. Utiliza uma chave de 256 bits e um nonce (número usado apenas uma vez) de 64 bits, proporcionando um forte nível de segurança. O funcionamento básico de Salsa20 envolve a aplicação de uma série de operações aritméticas e lógicas simples em uma matriz de 16 *Words* de 32 bits, derivadas da chave, do nonce e de um contador.

O núcleo da cifra é a função de 20 rodadas, demonstrado na Figura 12, que aplica uma série de operações que incluem adição modular, XOR e rotação de bits para misturar os dados. Estas operações são realizadas em duas fases alternadas de 10 rodadas cada:

primeiro nas colunas e depois nas diagonais da matriz. Após 20 rodadas, a matriz resultante é somada à matriz original e convertida em um fluxo de chave que é aplicado XOR com o texto original para produzir o texto cifrado.

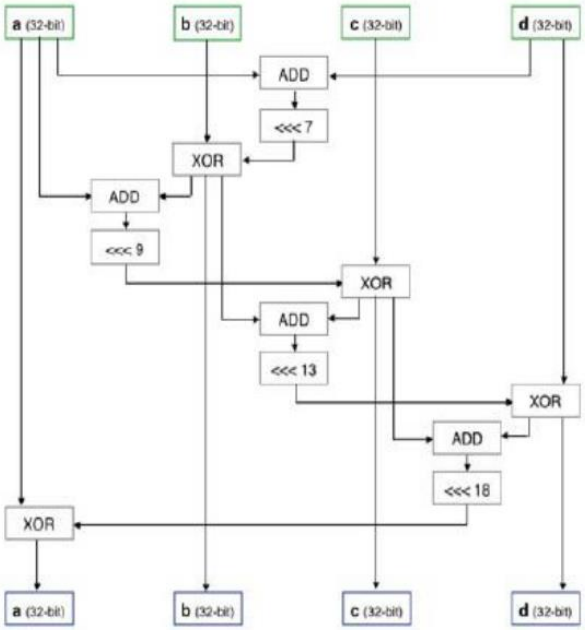
A matriz inicial de Salsa20 é composta por quatro partes principais:

- Constantes: Quatro *Word*s de 32 bits que são fixas para todos os usos da cifra.
- Chave: Oito *Word*s de 32 bits derivadas da chave de 256 bits.
- Nonce: Duas *Word*s de 32 bits derivadas do nonce de 64 bits.
- Contador: Duas *Words* de 32 bits que começam em zero e são incrementadas a cada bloco de 64 bytes cifrado.

Algumas características que contribuem para a segurança de Salsa20 incluem:

- Difusão Forte: As operações de rotação, adição e XOR garantem que pequenos desvios no texto claro ou na chave se espalhem rapidamente por todo o bloco.
- Resistência a Ataques Diferenciais e Lineares: A combinação das operações não lineares dificulta a predição dos bits do texto cifrado a partir do texto claro.
- Tamanho da Chave e Nonce: O uso de uma chave de 256 bits e um nonce de 64 bits proporciona um espaço de chave suficientemente grande para prevenir ataques de força bruta.

Figura 12 – *Quarter Round Function Salsa20*



Fonte: (Hameed, Hussein e Khalaf, 2020)

Assim aliado a estrutura de 20 rodadas da cifra proporciona uma excelente difusão de bits, dificultando criptoanálises bem-sucedidas. Crowley (2005) demonstrou que, embora seja possível encontrar distinções em cinco rodadas da cifra usando criptoanálise diferencial, a cifra completa de 20 rodadas permanece segura contra esse tipo de ataque.

Combinando segurança robusta, eficiência e simplicidade a cifra tem sido amplamente adotada em várias aplicações de criptografia devido à sua capacidade de fornecer proteção forte contra uma ampla gama de ataques, ao mesmo tempo em que oferece desempenho superior em uma variedade de plataformas.

2.3.7 Chacha 20

A cifra Chacha20 é um algoritmo de criptografia simétrica de fluxo. Ela foi desenvolvida por Daniel J. Bernstein em 2008 como uma variação aprimorada da cifra Salsa20. Sendo do mesmo modo projetada para ser rápida, segura e eficiente tanto em hardware quanto em software.

Bernstein (2008) cita que a cifra Chacha20 é considerada mais segura que sua predecessora Salsa20 devido a mudanças no design, incluindo a modificação das constantes de rotação, o que resulta em uma melhor dispersão dos bits. Estas mudanças não apenas melhoraram a segurança, mas também aumentaram a resistência a alguns tipos de ataques.

De acordo com Bernstein (2008), a cifra Chacha20 opera em blocos de 64bytes. Ela utiliza uma chave de 256 bits e um nonce de 96 bits, proporcionando um forte nível de segurança. A operação básica da cifra envolve a aplicação de uma série de operações aritméticas e lógicas simples em uma matriz de 16 *Word*s de 32 bits, derivadas da chave, do nonce e de um contador.

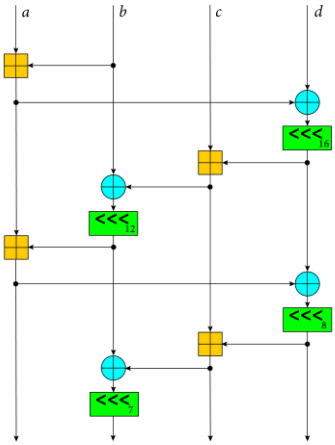
A matriz inicial de Chacha20 é composta por quatro partes principais:

- Constantes: Quatro *Word*s de 32 bits que são fixas para todos os usos da cifra.
- Chave: Oito *Word*s de 32 bits derivadas da chave de 256 bits.
- Nonce: Três *Word*s de 32 bits derivadas do nonce de 96 bits.
- Contador: Uma *Word* de 32 bits que começa em zero e é incrementada a cada bloco de 64 bytes cifrado.

O núcleo da cifra Chacha20 é a função de rodadas, que aplica 20 rodadas de operações que incluem adição modular, XOR e rotação de bits, como mostrado na Figura 13. Cada rodada consiste em uma série de quatro operações chamadas "*Quarter Rounds*", que misturam os bits da matriz de maneira não linear. Após 20 rodadas, a matriz resultante é somada à matriz original e então convertida em um fluxo de chave que é aplicado XOR com o texto original para produzir o texto cifrado.

Chacha20 foi projetada para ser segura contra uma variedade de ataques criptográficos. A estrutura de 20 rodadas proporciona uma excelente difusão de bits, dificultando a realização de criptoanálises bem-sucedidas. Mantendo as mesmas características que contribuem para a segurança da Salsa20 citada na seção 2.3.6, que são: difusão forte, resistência a ataques diferenciais e lineares e tamanho da chave e nonce.

Figura 13 – *Quarter Round Function Chacha20*



Fonte: (“chacha20 - Rust”, 2024)

ChaCha20 se destaca como uma cifra de fluxo moderna que combina segurança robusta, eficiência e simplicidade. Desenvolvida como uma melhoria do Salsa20, ChaCha20 tem sido amplamente adotada em várias aplicações de criptografia devido à sua capacidade de fornecer proteção forte contra uma ampla gama de ataques criptográficos, ao mesmo tempo em que oferece desempenho superior em uma variedade de plataformas.

2.3.8 Blake2s

BLAKE2s é uma função de *hash* criptográfico que faz parte da família BLAKE2. A função de *hash* BLAKE2 foi proposta em 2008 por Jean-Philippe Aumasson, Luca

Henzen, Willi Meier e Raphael C.-W. Phan, destaca-se pela sua segurança e eficiência. BLAKE2s, uma variante simplificada da BLAKE2, foi criada para oferecer uma alternativa leve e rápida, mantendo os princípios de segurança da versão completa.

Segundo Aumasson et al. (2013), a *hash* BLAKE2S foi desenvolvida motivado pela necessidade de uma função de *hash* eficiente para plataformas de 8 a 32 bit, produzindo resumos de 1 a 32 bytes. Incorporando a estrutura de segurança de sua versão completa, mas otimizada para a implementação em cenários onde a potência e a capacidade de processamento são restritas.

Utiliza uma construção baseada em uma rede de permutação e substituição inspirada na cifra de fluxo ChaCha. A estrutura é composta por 10 rodadas de operações que incluem adição modular, XOR e rotações de bits. A função de compressão da BLAKE2s é projetada para ser rápida e segura, garantindo a integridade dos dados processados.

A configuração inicial envolve a criação de uma matriz de estado a partir do texto de entrada e das constantes de inicialização. Tendo a função de compressão como núcleo da BLAKE2S. Ela toma como entrada a matriz de estado e a mensagem de entrada, aplicando uma série de operações para misturar os bits de forma não linear. Sendo então processada por uma série de rodadas de transformações para produzir o *hash* final.

Além disso, foram implementadas medidas de segurança adicionais para fortalecer a resistência da cifra contra criptoanálise. Essas medidas incluíram a introdução de constantes de inicialização, que adicionam um elemento de aleatoriedade ao processo de *hash*, e utilização de rede de permutação complexa, que aumenta a difusão dos dados, tornando mais difícil para um atacante descobrir padrões ou relações que possam comprometer a segurança da função *hash*.

Suas características de baixo consumo de recursos e alta velocidade tornam-na ideal para dispositivos com recursos limitados, enquanto sua robustez de segurança garante a integridade e autenticidade dos dados processados.

2.3.9 SHA – 256

Secure Hash Algorithm 256-bit (SHA-256) é uma função de *hash* desenvolvida pela NSA dos Estados Unidos e publicada pelo NIST como parte da família de algoritmos

SHA-2. Desde sua introdução em 2001, SHA-256 tornou-se amplamente adotada devido à sua robustez e segurança, desempenhando um papel crucial na utilização para comprovar a integridade de dados em diversas aplicações.

De acordo com Hawkes, Paddon e Rose (2004) a *hash* SHA-256 foi introduzida como parte da série de algoritmos SHA-2, sucedendo a série SHA-1. A necessidade de SHA-2 surgiu devido a preocupações de segurança com SHA-1. Fornecendo uma maior segurança em comparação com SHA-1, usando uma saída de 256 bits em vez de 160 bits. Sendo publicada pelo NIST suas especificações em 2001 no documento FIPS PUB 180-2 (*Federal Information Processing Standards Publication*).

Como uma função de *hash* que processa blocos de dados de forma iterativa, produzindo um valor de *hash* fixo de 256 bits. De acordo com May (2015) a função é composta por várias etapas que garantem a difusão e confusão dos bits de entrada.

- Vetores de Inicialização: O algoritmo inicializa com a sequência de 64 *Word*s constantes de 32 bits, conhecidas como constantes predefinidas.
- Preparação da Mensagem: A mensagem de entrada é dividida em blocos de 512 bits. Se a mensagem não é múltipla de 512 bits, ela é preenchida para atingir o tamanho necessário. Este preenchimento inclui um bit '1', seguido por zeros, e termina com a representação em 64 bits do tamanho original da mensagem.
- Expansão da Mensagem: O bloco de 512 bits é expandido em 64 *Word*s de 32 bits usando uma série de operações lógicas e aritméticas.
- Rodadas de *Hash*: Realiza 64 rodadas de operações, cada uma envolvendo a combinação dos valores das *Word*s expandidas e as constantes de rodada predefinidas. As principais operações usadas são:
- Adição Modular: Operações de adição modular (mod 2^32) garantem a propagação de alterações nos bits.
- Operação XOR: Combina bits de diferentes *Word*s de maneira não linear.
- Rotações e Deslocamentos: Operações de rotação e deslocamento de bits garantem a difusão dos bits por todo o estado interno.

Cada rodada transforma os vetores de estado através dessas operações, garantindo que pequenas alterações na entrada resultem em mudanças significativas na saída final. Após processar todos os blocos de 512 bits da mensagem de entrada, os vetores de estado

são combinados para produzir o valor de *hash* final de 256 bits. Este valor é uma representação condensada e única da entrada original.

A SHA-256 é eficiente em termos de implementação, é suportada por muitos processadores modernos com instruções específicas para acelerar as operações podendo ser executada em uma ampla gama de plataformas, desde dispositivos com recursos limitados até servidores de alta performance. Sua resistência comprovada a uma ampla gama de ataques criptográficos e sua eficiência de implementação garantem sua relevância

2.3.10 SHA3 – 256

SHA3-256 é uma função de *hash* desenvolvida como parte da família de algoritmos SHA-3. Esta função foi padronizada pelo NIST em 2015, desenvolvido por Guido Bertoni, Joan Daemen, Michaël Peeters e Gilles Van Assche, a SHA3-256 é uma das variantes do KECCAK, oferecendo um tamanho de saída de 256 bits, semelhante ao SHA-256 da família SHA-2.

De acordo com Bertoni et al. (2013), a SHA3-256 é fundamentada em uma estrutura de esponja, uma abordagem diferente dos modelos iterativos usados nas funções de *hash* SHA-1 e SHA-2. A estrutura de esponja é altamente flexível e oferece robustez contra vários tipos de ataques criptográficos. Sendo consistida em duas fases principais:

- Absorção: A mensagem de entrada é absorvida em blocos pelo estado interno da esponja.
- Extrusão: O *hash* final é extraído do estado interno através de uma série de operações.

O estado interno da esponja é uma matriz de 5x5 *Word*s de 64 bits, totalizando 1600 bits. Durante a fase de absorção, a mensagem é dividida em blocos e misturada com o estado interno através de operações XOR. Após a fase de absorção, o estado interno é processado por várias rodadas de uma função de permutação antes da fase de extrusão, onde o *hash* final é extraído.

Sendo a função de permutação o núcleo do SHA3-256 como descrito por Dworkin (2015), ela consiste em cinco operações principais aplicadas ao estado interno:

- Theta (θ): Mistura os bits dentro de cada coluna da matriz.

- Rho (ρ): Rotaciona os bits dentro de cada *Word*.
- Pi (π): Permuta as posições das *Word*s na matriz.
- Chi (χ): Aplica uma transformação não linear a cada *Word* da matriz.
- Iota (ι): Introduz constantes de rodada para garantir que cada rodada seja única.

Estas operações são aplicadas repetidamente durante um número fixo de rodadas, garantindo uma mistura intensa dos bits e dificultando a predição do estado interno.

A flexibilidade da estrutura de esponja permite ajustes de desempenho para otimizar a implementação em hardware e software. SHA3-256 pode ser implementado de maneira eficiente em microcontroladores, dispositivos móveis e sistemas embarcados.

Como uma função de *hash* moderna que combina segurança robusta, eficiência e flexibilidade. Sua estrutura de esponja inovadora e a função de permutação complexa garantem uma resistência robusta a ampla gama de ataques criptográficos, tornando-a uma ferramenta essencial na segurança da informação moderna.

3. METODOLOGIA

Neste trabalho, demonstraremos o impacto no consumo ao utilizar as criptografias mencionadas na Seção 2.3, avaliando sua eficiência em termos de processamento e uso de recursos energéticos, a fim de garantir um desempenho adequado dentro das restrições de hardware. Foi utilizado dois microcontroladores, sendo um ESP32 Devkit V1 e um Arduino nano, suas especificações podem ser observadas nas tabelas 4 e 5. Para análise, levaremos em conta os tópicos discutidos por Hatzivasilis et al. (2018) e realizaremos uma avaliação empírica do consumo de cada cifra.

- Consumo de memória: O consumo de RAM afeta a capacidade de execução da aplicação, enquanto a ROM influencia a capacidade de armazenamento de variáveis e inicialização do sistema.
- Latência: Refere-se ao número de ciclos de clock de processamento que o dispositivo utiliza para criptografar ou descriptografar um bloco de dados.
- *Throughput*: Refere-se à *Throughput* em bits por segundo (b/s) alcançada pela criptografia ou descriptografia do algoritmo em uma frequência específica. Mesmo o ESP32 e o Arduino trabalhando em suas frequências padrão citada na tabela 4 e 5. A *Throughput* será aproximada para a frequência de 4MHz pois

muitos trabalhos utilizam essa frequência para algoritmos implementados em softwares. Podemos obter a *Throughput* (Th) a partir da equação 1, onde, Bs refere-se ao tamanho do bloco em bits, f é a frequência de clock e a latência é o número de ciclos de clock necessários para cifrar ou descifrar um bloco.

$$Th_{b/s} = \frac{Bs_{bits} * f_{clock}}{Latency_{cicles/block}} \quad (1)$$

- Consumo energético: Refere-se ao consumo de energia por bit para cifrar ou descifrar um bloco de dados, sendo um aspecto crucial em dispositivos alimentados por bateria. Esse consumo pode ser determinado usando a equação 2, com o resultado expresso em microjoules (µJ). A potência (P) é calculada a partir da tensão de alimentação e da corrente consumida pelo microcontrolador estando em operação, conforme demonstrado nas tabelas 4 e 5.

$$Energia_{\mu J} = \frac{Latency_{cicles/block} * P_{\mu W}}{Bs_{bits}} \quad (2)$$

- A eficiência de software em cifras leves é um aspecto crítico em ambientes com recursos limitados, como dispositivos IoT, sensores embarcados e dispositivos móveis. Esses dispositivos geralmente operam com restrições severas de energia, memória e capacidade de processamento, tornando essencial o uso de algoritmos criptográficos que sejam tanto seguros quanto eficientes.

  A equação de eficiência pode ser representada como:

$$Eficiencia = \frac{Th_{b/s}}{Cs_{KB}} \quad (3)$$

  onde:

  Cs: Tamanho da Implementação, se referindo à quantidade de recursos de memória (ROM) que a cifra ocupa.

  Quanto maior o valor da eficiência, melhor é a implementação da cifra para um dado ambiente. A eficiência é maximizada quando o desempenho é alto e o tamanho da implementação é pequeno, permitindo que a cifra funcione de maneira rápida e economize recursos valiosos, como energia e memória, em dispositivos com capacidades limitadas.

- Análise empírica: Para realizar uma análise detalhada do consumo energético das cifras, seguimos o processo descrito no fluxograma da Figura 14. Este processo envolve a execução de um loop com 5.000 repetições no ESP32 e 100 repetições no Arduino Nano. O número de repetições foi estimado com base na maior latência obtida de cada dispositivo. Com esses dados em mãos, verificamos quantas repetições poderiam ser realizadas em 1 segundo e, a partir dessa informação, escolhemos um número arredondado próximo ao valor calculado. Dessa forma, garantindo uma comparação eficaz e consistente na análise. Esse procedimento rigoroso permitiu estimar a diferença no consumo energético entre as cifras, fornecendo uma base robusta para a avaliação comparativa de sua eficiência energética.

  Realizamos cinco medições para cada cifra, com uma taxa de amostragem de aproximadamente 70 medições por segundo durante um período de 5 minutos, a fim de obter uma média de consumo energetico. Este procedimento foi repetido para todas as cifras analisadas. O layout de medição utilizado está ilustrado na Figura 15 e 16, demonstrando o cuidado meticuloso empregado para garantir a precisão e a confiabilidade dos dados obtidos.

As cifras Salsa20 e Simon foram implementadas a partir dos algoritmos genéricos disponibilizados por seus respectivos autores: Bernstein (2007) e Beaulieu et al. (2013). Para as demais cifras, foram utilizadas bibliotecas já existentes desenvolvidas por Weatherley (2017), Bose (2021) e Falchetto (2019). Essas bibliotecas foram adaptadas para assegurar a compatibilidade e o desempenho ideal nos dois dispositivos utilizados nos experimentos.

Para os testes de criptografia, a palavra "Hello" foi escolhida como entrada e declarada como um vetor do tipo bytes. Essa abordagem garantiu consistência e padronização nos testes de todas as cifras, permitindo uma avaliação comparativa precisa e confiável dos algoritmos implementados.

Tabela 4 – Descrição do modulo ESP32

Descrição: ESP32	
Processador	32-bits
Tensão de entrada	3.3V
Corrente alimentação	30mA ~ 68mA
CPU Clock	240MHz
SRAM	520KB
ROM	448KB

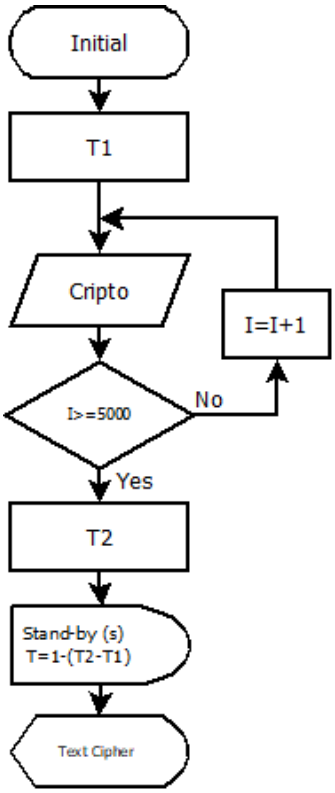
Fonte: Própria Autoria

Tabela 5 – Descrição do modulo Arduino Nano

Descrição: Arduino Nano	
Processador	8-bits
Tensão de entrada	5V
Corrente alimentação	1,9mA ~ 14mA
CPU Clock	16MHz
SRAM	2KB
ROM	33KB

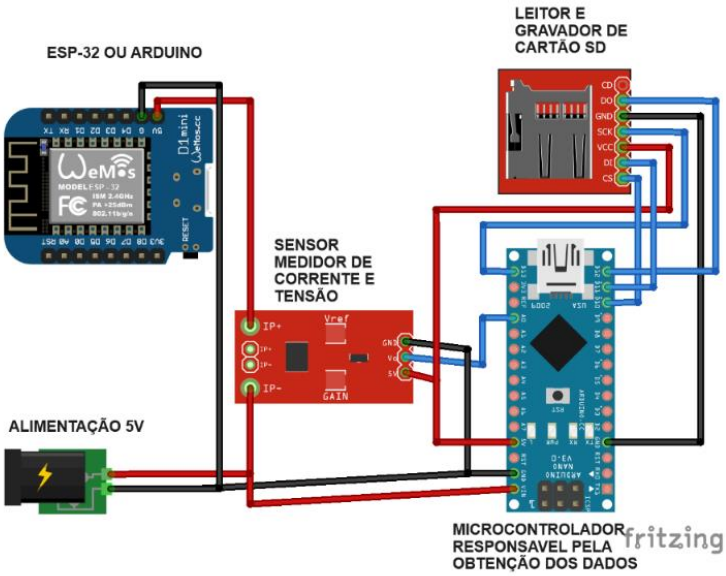
Fonte: Própria Autoria

Figura 14 - Fluxograma do processo executado para medição do consumo.



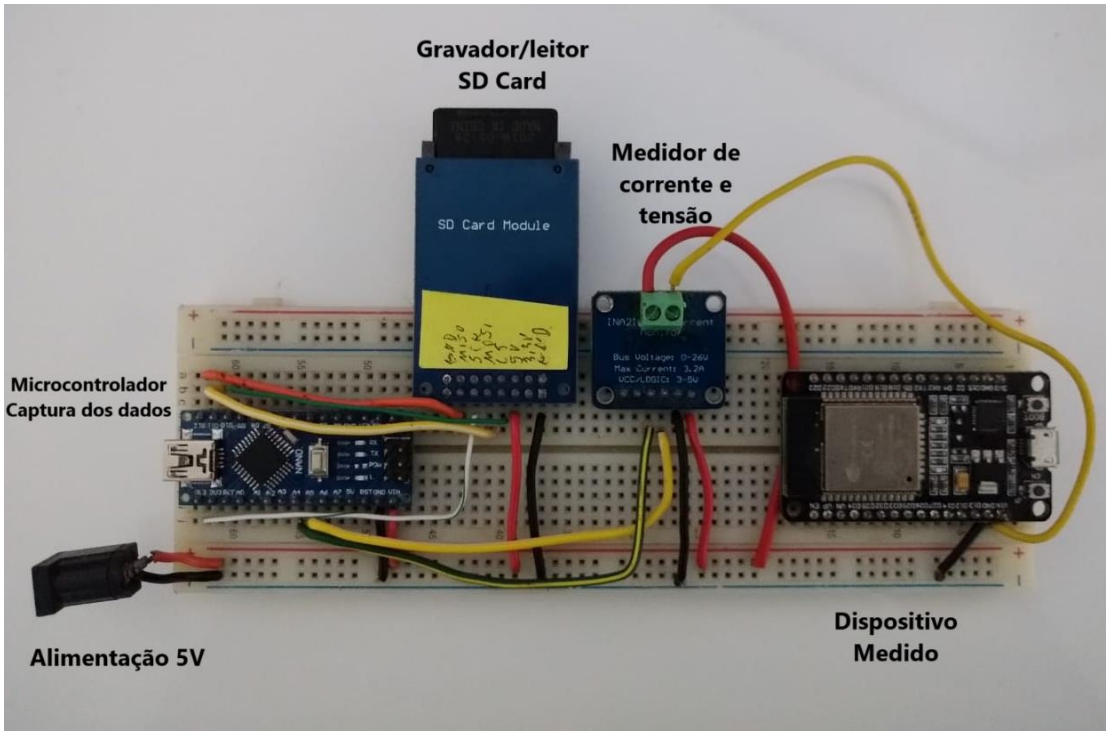
Fonte: Própria Autoria

Figura 15 – *Layout* de medição



Fonte: Própria Autoria

Figura 16 – *Layout* Montado



Fonte: Própria Autoria

A Tabela 6 apresenta as configurações utilizadas nos testes, incluindo o tamanho dos blocos de dados e o comprimento das chaves criptográficas. Os valores foram cuidadosamente escolhidos de modo a se alinharem ao padrão de 128 bits, ou, quando não possível, mais próximo desse valor.

Tabela 6 – Configuração das Cifras e *Hashs*

Cifra/*Hash*	Tamanho do Bloco (bits)	Tamanho da Chave (bits)	Rodadas
AES	128	128	10
Camellia	128	128	18
Simon	128	128	68
Speck	128	128	32
XXTEA	64	128	64
Salsa20	40	256	20
Chacha20	40	256	20

Blake2s	256	-	10
SHA-265	256	-	64
SHA3-265	256	-	24

Fonte: Própria Autoria

4. ANÁLISE E RESULTADOS

Este trabalho não tem como objetivo testar a resistência a ataques de criptoanálise.

Com o estudo do referencial teórico pretende-se, inicialmente, estimar o consumo de recursos para cada cifra e comparar com os disponíveis nos kits. Após serão verificadas a possibilidade de aplicação dos algoritmos de criptografia e descriptografia. Em seguida será analisada latência de cifra e decifra e *Throughput* de cada criptografia, em comparação com o envio de dados não criptografados, buscando ser capaz de criptografar e descriptografar dados rapidamente, sem causar atrasos perceptíveis no tempo de resposta do sistema.

4.1 Consumo de Memória

Como os microcontroladores dependem de memória ROM para armazenar instruções de inicialização e firmware permanentes, fornecendo estabilidade ao sistema, e da memória RAM que é utilizada para armazenar temporariamente dados e variáveis durante a execução do programa.

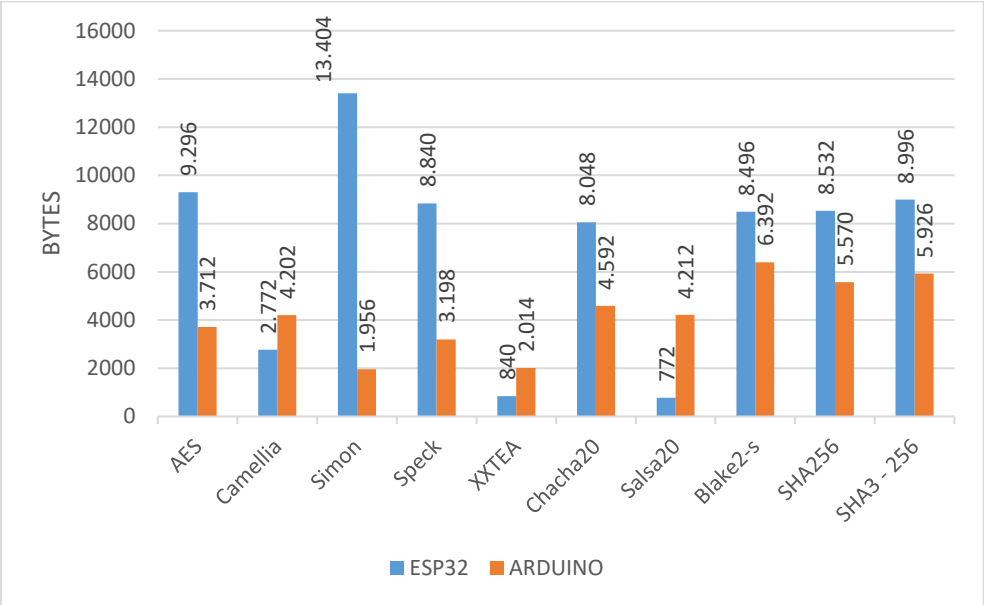
O tamanho e a velocidade da memória ROM e RAM afetam diretamente sua capacidade de execução de aplicações e processamento de dados. Consequentemente gerenciar eficientemente o uso das memórias é essencial para garantir o desempenho e estabilidade adequado ao sistema. Sabendo disso obtemos o consumo de memória RAM e ROM utilizando a ferramenta de carregamento do código fonte, que durante a compilação é possível estima o consumo total dessas memórias, e assim podendo avaliar o consumo de memória dos dispositivos.

As Tabelas 4 e 5 mostram a quantidade de memória disponível nos dois dispositivos, levando em consideração que parte dela já é ocupada pelo firmware base. Nos testes em bancada com um *sketch* vazio, o *firmware* ocupa, em média, 252.585 KB de memória ROM e 17 KB de RAM no ESP32, deixando 48% da memória ROM e 95% da memória RAM livres. No Arduino Nano, os valores são de 444 bytes de memória

ROM e 9 bytes de RAM, representando 98,65% e 99,55% de memória livre, respectivamente.

As Figuras 17 e 18 ilustram a ocupação da memória, já subtraídos os valores do firmware base. A Figura 17 mostra os valores de memória ROM utilizada, enquanto a Figura 18 apresenta a ocupação da memória RAM. Estas representações visuais facilitam a compreensão do impacto do firmware e das cifras criptográficas na disponibilidade de memória dos dispositivos.

Figura 17 – Utilização de memória ROM



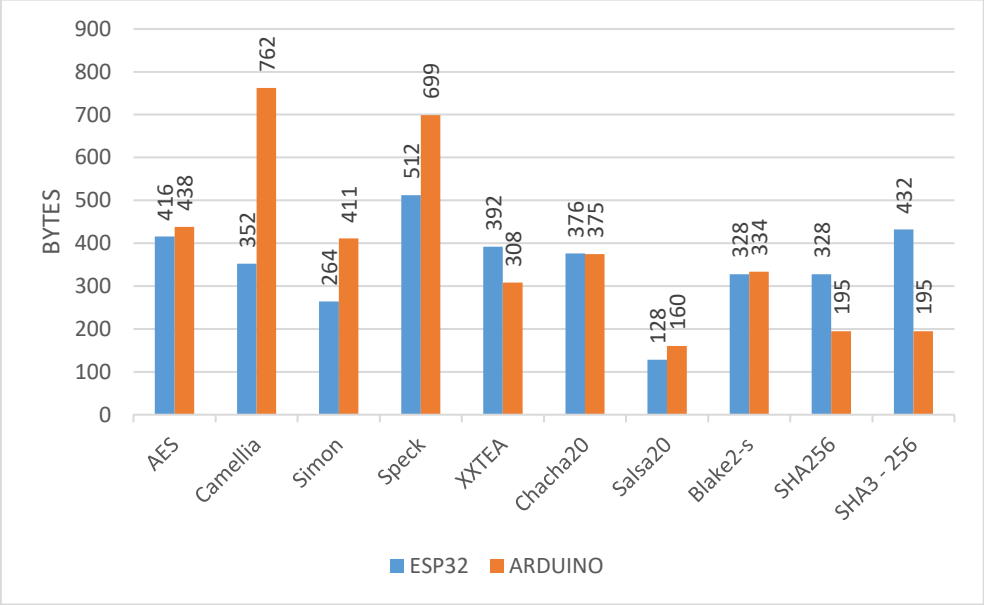
Fonte: Própria Autoria

A análise da Figura 17 revela que a taxa de ocupação de memória varia significativamente tanto entre as diferentes cifras quanto entre os dispositivos utilizados. Esta variação destaca-se como a otimização de algoritmos criptográficos pode ser específica para cada arquitetura de microcontrolador. Especificamente, observa-se que a maioria das cifras abordadas são menos otimizadas para o ESP32, em comparação com o Arduino Nano.

Por exemplo, a cifra Simon ilustra bem essa diferença. No Arduino Nano, Simon é a cifra que ocupa menos espaço de memória, destacando-se pela sua eficiência em termos de uso de recursos nesse dispositivo. Isso pode ser atribuído ao *design* leve e eficiente da cifra, que é bem adequado para a arquitetura mais simples e restrita do Arduino Nano. No entanto, quando a mesma cifra é implementada no ESP32, observa-se

o fenômeno oposto: Simon ocupa mais espaço de memória em comparação com outras cifras. Essa disparidade pode ser explicada por várias razões técnicas.

Figura 18 – Utilização de memória RAM



Fonte: Própria Autoria

O ESP32 possui uma arquitetura mais complexa e rica em recursos, o que pode exigir uma implementação diferente das cifras para aproveitar plenamente suas capacidades. As instruções adicionais e a gestão de recursos mais avançada podem resultar em um maior consumo de memória para algumas cifras que não foram especificamente otimizadas para essa plataforma. Além disso, o compilador utilizado para o ESP32 pode introduzir *overhead* adicional ao tentar otimizar a execução das cifras, enquanto o compilador do Arduino Nano, projetado para simplicidade, pode resultar em um código mais enxuto. Essas diferenças sublinham a importância de considerar a arquitetura específica e o ambiente de desenvolvimento ao implementar algoritmos criptográficos.

Outra cifra que exemplifica bem essa variação é a Salsa20. Embora também mostre eficiência no Arduino Nano, sua implementação no ESP32, embora ainda eficiente, pode não ser tão otimizada quanto outras cifras que foram desenvolvidas com a arquitetura do ESP32 em mente. Isso reflete a necessidade de ajustes e otimizações específicas para cada plataforma para garantir que o desempenho e o consumo de memória sejam os melhores possíveis.

Apesar dessas observações, ao analisarmos a porcentagem de memória ocupada, conforme demonstrado na Tabela 7, podemos ver diferenças notáveis no impacto das cifras nos dois dispositivos. No Arduino Nano, a cifra Simon ocupa a menor quantidade de espaço, utilizando apenas 5,93% da memória ROM disponível. Em contraste, a cifra Blake-s ocupa a maior quantidade de espaço, consumindo 19,37% da memória ROM. No ESP32, a distribuição de memória ocupada pelas cifras é bastante diferente. A cifra que ocupa menos espaço é a Salsa20, utilizando apenas 0,17% da memória ROM. Em contrapartida, a cifra Simon, que era com menor taxa de ocupação de memória ROM no Arduino, ocupa a maior quantidade de espaço no ESP32, utilizando 2,99% da memória ROM.

Quando somamos o espaço ocupado pelo firmware base – que, na Tabela 7, está representado por um sketch vazio – a ocupação total de memória ROM no Arduino Nano é de 20,72%. No ESP32, essa ocupação total é significativamente maior, chegando a 59,37%. Apesar dessa diferença, ambos os dispositivos ainda possuem uma quantidade considerável de memória disponível para a implementação de outras rotinas, como leitura de sensores e conexões sem fio.

No Arduino Nano, com uma ocupação total de memória ROM de 20,72%, restam aproximadamente 79,28% de memória livre. Esse espaço pode ser utilizado para integrar funções adicionais, garantindo que o dispositivo continue a operar de maneira eficiente sem comprometer a estabilidade ou a velocidade de processamento.

No ESP32, mesmo com uma ocupação de 59,37%, o dispositivo ainda dispõe de 40,63% de memória ROM livre. Esta quantidade de memória é mais do que suficiente para suportar funções adicionais complexas, incluindo a implementação de protocolos de comunicação sem fio, processamento de dados de sensores em tempo real e outras rotinas essenciais para aplicações avançadas de IoT.

A análise da Figura 18 revela que a taxa de ocupação de memória RAM não varia significativamente tanto em comparação com a memória ROM.

O consumo de memória RAM em sistemas embarcados é um fator crítico, especialmente em dispositivos com recursos limitados, como o Arduino e o ESP-32. No caso do Arduino, a cifra que mais consome memória RAM é a Camellia, utilizando 762 bytes. Esse alto consumo reflete a complexidade da cifra, que oferece um elevado nível

de segurança criptográfica, mas a custo de maior uso de recursos. Em contraste, a cifra que apresenta o menor uso de memória no Arduino é a Salsa20, ocupando apenas 160 bytes. No ESP-32, por sua vez, a cifra que mais utiliza RAM é a Speck, com 512 bytes. Apesar de consumir menos que a Camellia no Arduino, essa cifra ainda ocupa uma parcela significativa da memória disponível no ESP-32. Em contrapartida, a Salsa20, também neste dispositivo, continua a ser a mais eficiente em termos de consumo de RAM, utilizando apenas 128 bytes.

Essas observações ressaltam a importância de uma análise detalhada e específica do consumo de memória ao desenvolver soluções criptográficas para diferentes plataformas. Cada microcontrolador possui suas próprias características e limitações, e entender como cada cifra impacta esses dispositivos é crucial para otimizar o desempenho e a eficiência do sistema. A escolha cuidadosa das cifras e a consideração do consumo total de memória permitem maximizar o uso eficiente dos recursos disponíveis, garantindo que o sistema possa executar todas as suas funções planejadas de maneira eficaz.

Em resumo, a análise das taxas de ocupação de memória nas Figuras 17 e 18 revela a importância da otimização específica para a arquitetura de cada microcontrolador. Cifras como Simon e Salsa20 mostram como o mesmo algoritmo pode ter desempenhos muito diferentes em dispositivos distintos. Essa compreensão é essencial para o desenvolvimento de soluções criptográficas eficientes e seguras em microcontroladores como o ESP32 e o Arduino Nano, assegurando que as aplicações possam operar de maneira eficaz sem comprometer os recursos disponíveis.

Tabela 7 – Taxa de ocupação das memórias

	ESP-32		**ARDUINO**	
CIFRAS	ROM	RAM	ROM	RAM
AES	2,08%	0,08%	11,25%	21,90%
CAMELLIA	0,62%	0,07%	12,73%	38,10%
SIMON	2,99%	0,05%	5,93%	20,55%
SPECK	1,97%	0,10%	9,69%	34,95%
XXTEA	0,19%	0,08%	6,10%	15,40%
CHACHA20	1,80%	0,07%	13,92%	18,75%
SALSA20	0,17%	0,02%	12,76%	8,00%
BLAKE2-S	1,90%	0,06%	19,37%	16,70%
SHA256	1,90%	0,06%	16,88%	9,75%
SHA3 - 256	2,01%	0,08%	17,96%	9,75%
VAZIO	56,38%	3,27%	1,35%	0,45%

Fonte: Própria Autoria

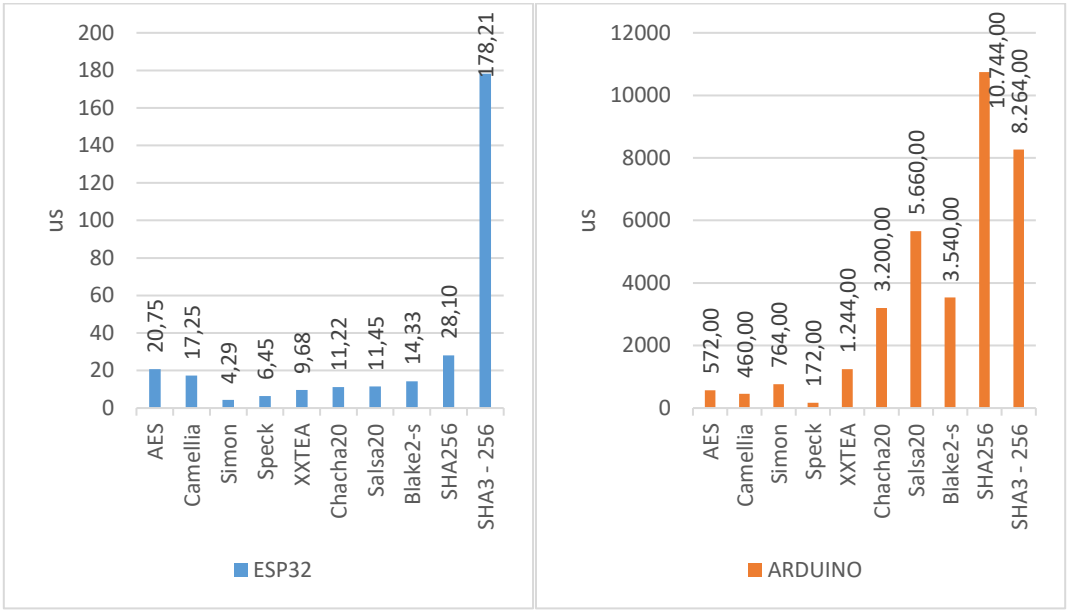
4.2 Latência de criptografia

Como a criptografia leve deve ser executada de forma eficiente em dispositivos com recursos limitados. Isso significa que o método de criptografia deve executas a criptografia ou descriptografia no menor tempo possível, principalmente quando alimentado por baterias ou fonte de energia sazonal. Sendo o principal objetivo deste trabalho verificar o impacto no consumo de energia pela utilização de criptografia, consequentemente demonstrar as cifras com menor consumo de energia e melhor relação custo/segurança.

Para medir com precisão a latência, especialmente considerando que alguns algoritmos executam a cifra em poucos microssegundos e como o ESP32 não possui registradores que operem em nanosegundos, adotamos uma abordagem alternativa para a medição. Utilizamos a função ESP.getCycleCount() antes e depois da cifra ou decifra. Essa função, quando acionada, registra em uma variável o número de pulsos de clock contados desde o instante em que o dispositivo foi inicializado. Essa estratégia nos permite capturar com detalhe a quantidade necessária de pulsos de clock para realizar a cifra ou a decifra da *Word*.

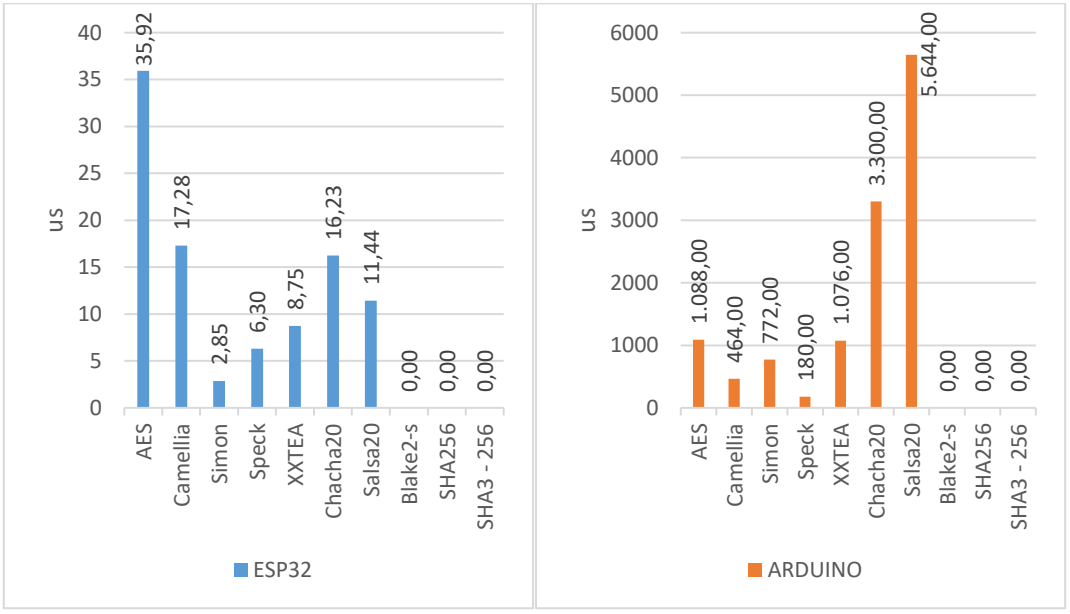
No Arduino Nano, para medir o tempo de cifragem e decifragem, utilizamos a função micros(), que registra a quantidade de milissegundos decorridos desde a inicialização do dispositivo. O processador do Arduino opera a uma frequência de 16 MHz quando alimentado por 5 Volts, o que significa que cada microssegundo corresponde a 16 pulsos de máquina. Isso permite estimar com precisão a quantidade de pulsos e o tempo em microssegundos necessários para executar as tarefas de cifragem e decifragem. Assim, ao calcular a diferença entre o momento final e inicial, obtemos a latência em ciclos por bloco criptografado, oferecendo uma visão do desempenho do algoritmo em termos de tempo de processamento. As Figuras 19 e 20 apresentam as latências microssegundos enquanto as Figuras 21 e 22 apresentam as latências em pulsos de máquina necessário para criptografar ou descriptografar utilizando um texto qualquer.

Figura 19 – Tempo de criptografia em microssegundos



Fonte: Própria Autoria

Figura 20 – Tempo de descriptografia em microssegundos



Fonte: Própria Autoria

Na apresentação dos resultados de latência, os tempos são primeiramente apresentados em microssegundos, com a exclusão da decifra das funções *hash*, devido à sua característica de não permitirem a recuperação da *Word* original. A maior parte das cifras apresentam tempos de cifragem e decifragem bastante próximos, refletindo uma simetria de desempenho. No entanto, duas exceções notáveis são as cifras AES e Simon,

que mostram diferenças significativas nos tempos de cifragem e decifragem, excedendo 50% em ambos os dispositivos.

Para a cifra AES, essa assimetria é particularmente pronunciada ambos os dispositivos. A decifragem pode demorar até 90% mais do que a cifragem para processar o mesmo bloco de dados. A cifra Simon também apresenta uma diferença significativa, mas esta discrepância é dependente da plataforma. No ESP-32, a cifragem com Simon pode levar até 60% mais tempo do que a decifragem. Este comportamento anômalo no ESP-32 pode ser atribuído a particularidades na arquitetura do processador ou na maneira como as instruções de cifragem são executadas no hardware específico.

Outro aspecto relevante destacado nas Figuras 19 e 20 é o tempo de resposta de cada dispositivo durante o processo de cifragem e decifragem. Esta métrica é crucial para avaliar a eficiência das cifras em diferentes plataformas, proporcionando *insights* valiosos sobre a adequação de cada algoritmo para aplicações específicas.

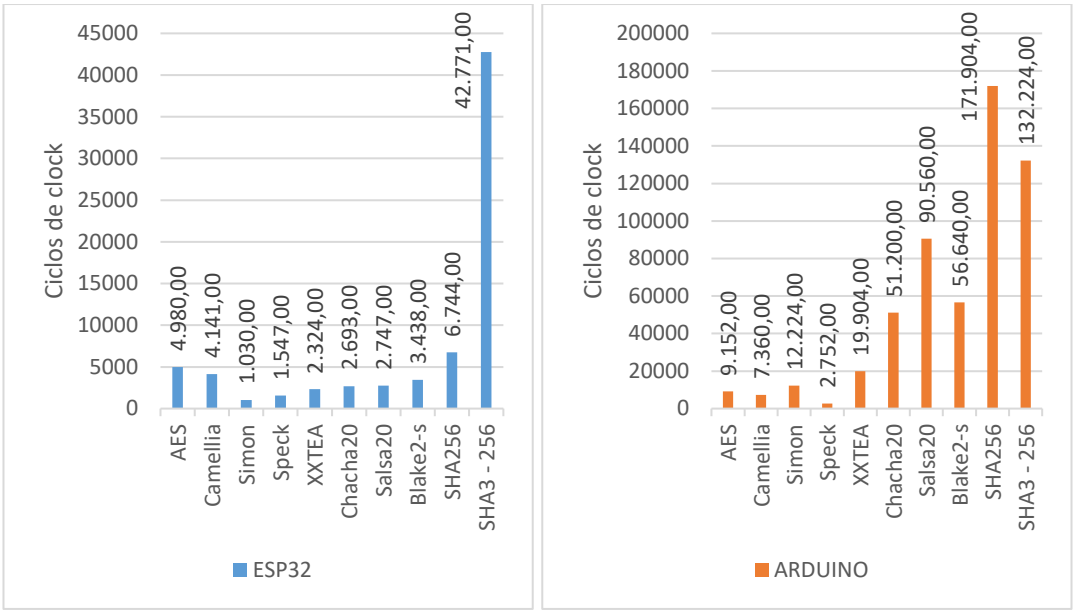
No Arduino Nano, a cifra mais rápida é a Speck, que leva, em média, 172 µs para criptografar um bloco de dados. Esta eficiência torna a Speck uma escolha ideal para aplicações que exigem rapidez e baixo consumo de recursos. Em contrapartida, a função de *hash* SHA-256 é a mais lenta, demorando 10,74 ms para processar um bloco de dados. A lentidão relativa do SHA-256 no Arduino Nano reflete a complexidade do algoritmo e a demanda de recursos computacionais.

Por outro lado, no ESP-32, os tempos de resposta caem abruptamente, demonstrando a superioridade da plataforma em termos de capacidade de processamento. A cifra Simon é a mais rápida no ESP-32, levando apenas 2,85us para descifrar um bloco de dados. Esta velocidade é significativamente maior do que a observada no Arduino Nano, evidenciando a eficiência do ESP-32 em executar operações criptográficas complexas de maneira rápida e eficaz. Entretanto, a função de *hash* SHA3-256 se destaca como a mais lenta no ESP-32, levando 178 µs para processar um bloco de dados. Embora este tempo seja consideravelmente menor do que o tempo de processamento do SHA-256 no Arduino Nano, ainda representa uma carga substancial em comparação com outras cifras no ESP-32.

Essas variações podem ser atribuídas à diferença na frequência de operação entre os dois dispositivos. Enquanto o Arduino Nano opera a uma frequência de clock de 16

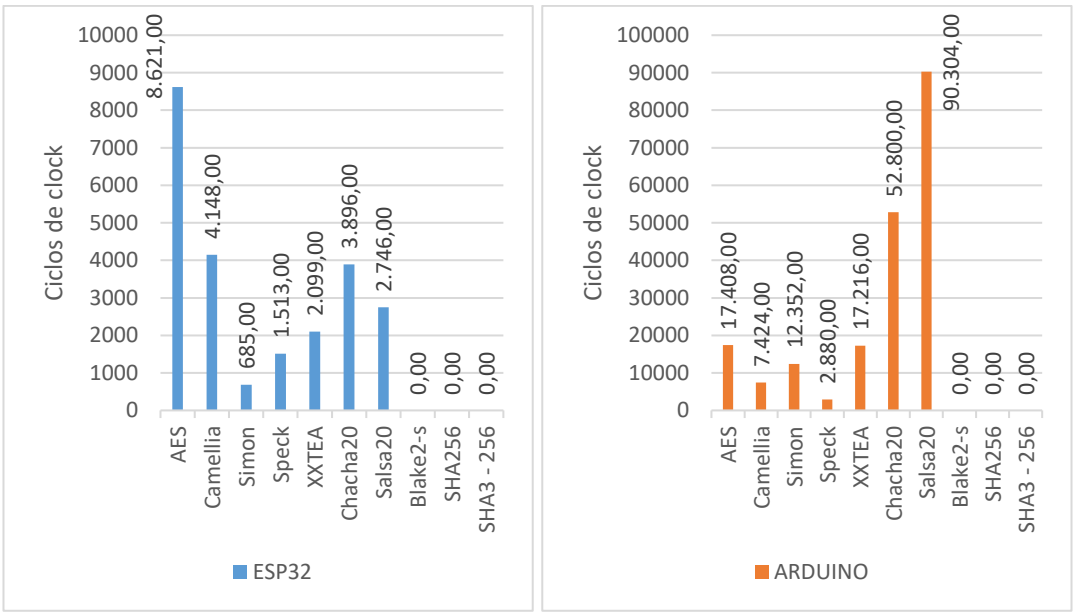
MHz, o ESP-32 funciona a uma frequência significativamente maior de 240 MHz, cerca de 15 vezes mais rápida. Essa diferença de desempenho sublinha a importância de considerar tanto a plataforma de hardware quanto as características específicas de cada algoritmo criptográfico ao projetar sistemas de segurança.

Figura 21 – Latência de Criptografia por bloco



Fonte: Própria Autoria

Figura 22 – Latência de Descriptografia por Bloco



Fonte: Própria Autoria

Nas Figuras 19 e 20, foi possível comparar a diferença de desempenho devido às diferentes frequências de clock dos dispositivos. Já as Figuras 21 e 22 ilustram o

desempenho das arquiteturas, comparando a quantidade de ciclos de máquina necessários para que os dispositivos realizem a criptografia ou a descriptografia. Embora os resultados sejam próximos para as cifras de bloco, o ESP-32 se destaca pela sua arquitetura de 32 bits e tecnologia mais recente em comparação à arquitetura de 8 bits do Arduino Nano.

A superioridade do ESP-32 é evidente devido à sua capacidade de processar mais dados por ciclo de clock e realizar operações mais complexas de forma eficiente. A arquitetura de 8 bits do Arduino Nano, por outro lado, requer o uso de mais registradores e, consequentemente, um maior número de ciclos de máquina para realizar a mesma tarefa. Isso se torna especialmente perceptível nas cifras de fluxo e nas funções de *hash*, que são mais complexas do que as cifras de bloco abordadas neste estudo.

As cifras de fluxo e as funções de *hash*, devido à sua complexidade inerente, exigem o manejo de variáveis contendo até 1600 bits e a execução de operações não lineares, como deslocamentos de bits e operações XOR. Essas operações são naturalmente mais demoradas e exigem mais ciclos de máquina, especialmente em uma arquitetura de 8 bits como a do Arduino Nano. Por outro lado, o ESP-32, com sua arquitetura de 32 bits, pode lidar com essas operações de maneira muito mais eficiente, utilizando menos ciclos de máquina para processar a mesma quantidade de dados.

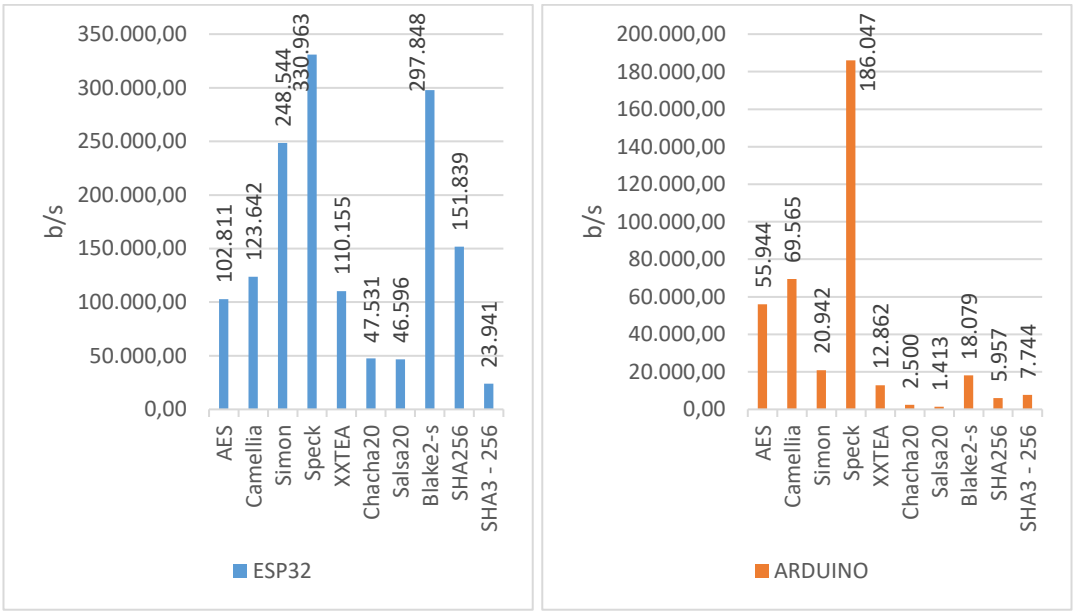
4.3 *Throughput*

A *Throughput* mede a quantidade de bits que a criptografia pode processar por segundo, sendo uma métrica crucial para avaliar o desempenho do algoritmo em microcontroladores. Quanto maior a *Throughput* de bits, mais rapidamente os dados podem ser transmitidos ou processados, resultando em comunicações ou operações mais ágeis e eficientes. Esta métrica é fundamental para determinar a adequação de um algoritmo criptográfico para aplicações que exigem alto desempenho e rapidez.

Utilizando a Equação 1, calculamos a *Throughput* para cada algoritmo, e os resultados são apresentados nas Figuras 23 e 24. Estas figuras ilustram claramente a eficiência de cada algoritmo em termos de processamento de dados, permitindo uma comparação direta entre os diferentes algoritmos implementados nos microcontroladores testados.

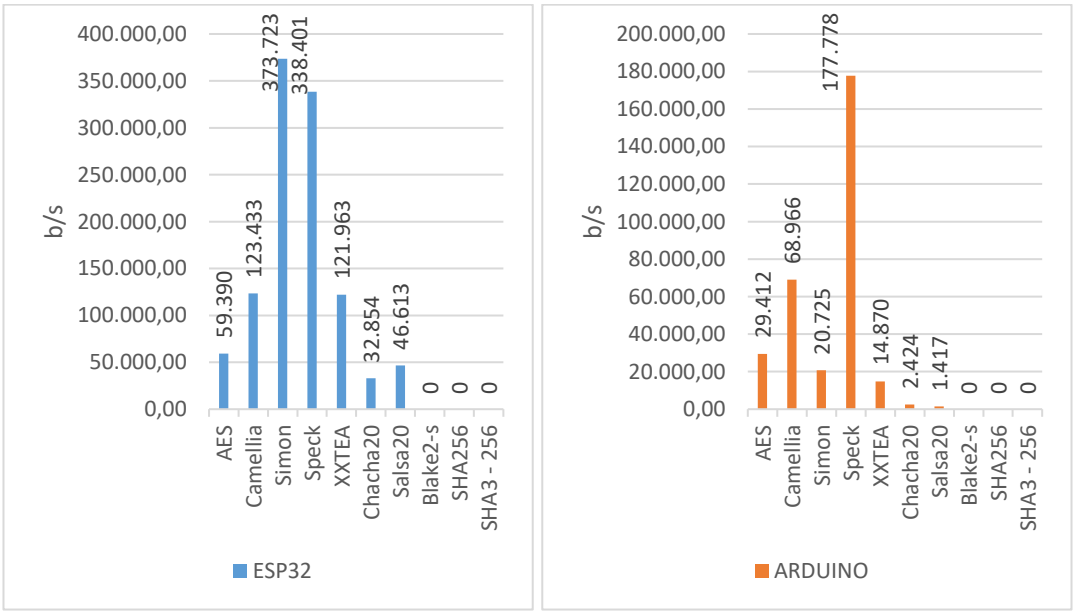
A análise dos valores obtidos revela a capacidade de cada algoritmo de lidar com o volume de dados em tempo real, fornecendo *insights* valiosos sobre quais algoritmos são mais adequados para aplicações específicas que exigem alta *Throughput*.

Figura 23 – *Throughput* de Criptografia



Fonte: Própria Autoria

Figura 24 – *Throughput* de Descriptografia



Fonte: Própria Autoria

Podemos observar nas Figuras 23 e 24 que a *Throughput* é inversamente proporcional à latência: quanto menor a latência, maior será a *Throughput*. Outro detalhe evidente é o impacto do tamanho do bloco na *Throughput*. Apesar das latências das cifras

Speck e XXTEA serem próximas, a diferença no tamanho dos blocos com os quais trabalham, 128 bits para Speck e 64 bits para XXTEA, faz com que a *Throughput* da Speck seja o dobro da XXTEA se ambos tivessem a mesma latência. Um exemplo adicional é a comparação entre AES e XXTEA no ESP-32. A AES trabalha com blocos de 128 bits e, durante a fase de criptografia, utiliza um pouco mais do que o dobro de ciclos de máquina para cifrar um bloco em comparação com a XXTEA. Esta diferença no consumo de ciclos de máquina resulta em taxas de transferência muito próximas: aproximadamente 102 Kb/s para a AES e um pouco mais de 110 Kb/s para a XXTEA.

Na análise, consideramos as cifras de fluxo como cifras de bloco com tamanho igual ao do dado cifrado, neste caso, a *Word* "Hello", que contém 40 bits. Dessa forma, podemos analisá-las e compará-las com as cifras de bloco e as funções de *hash*. Mesmo tratando os dados em blocos menores, as cifras de fluxo apresentam taxas de transferência inferiores em comparação com as cifras de bloco mencionadas. Isso ocorre principalmente devido à maior complexidade dos algoritmos de cifras de fluxo. Enquanto as cifras de bloco processam dados em grandes blocos de forma mais eficiente, as cifras de fluxo, apesar de serem adequadas para certas aplicações, requerem mais operações para cada bit de dados, resultando em uma menor *Throughput*. Esta diferença é destacada pela análise detalhada dos tempos de processamento e ciclos de máquina necessários para cada tipo de cifra mostrados nas Figuras de 19 a 22. As cifras de bloco, como a AES, demonstram uma maior eficiência no manuseio de grandes volumes de dados, enquanto as cifras de fluxo, embora robustas e seguras, são penalizadas por sua complexidade algorítmica.

As cifras que se destacam no Arduino Nano em termos de maior e menor *Throughput* são a Speck e a Salsa20, respectivamente. A cifra Speck apresenta uma *Throughput* de até 186 Kb/s durante a criptografia e 177 Kb/s durante a descriptografia, destacando-se como a mais eficiente nesse dispositivo. Em contraste, a cifra Salsa20 tem uma *Throughput* significativamente menor, com pouco mais de 1,4 Kb/s tanto para criptografia quanto para descriptografia, demonstrando a maior complexidade do algoritmo e sua menor eficiência no Arduino Nano.

No ESP-32, a cifra com a maior *Throughput* durante a criptografia é novamente a Speck, com uma taxa de 330 Kb/s. Durante a descriptografia, a cifra que se destaca é a Simon, alcançando uma *Throughput* de 373 Kb/s. Esses resultados refletem a capacidade

do ESP-32 de processar dados de maneira extremamente eficiente, graças à sua arquitetura de 32 bits e maior frequência de clock. Por outro lado, as cifras com as menores taxas de transferência no ESP-32 são a SHA3-256 e a Chacha20. Durante a fase de criptografia, a SHA3-256 apresenta uma *Throughput* de 23,941 Kb/s, enquanto a Chacha20 tem uma taxa de 32,825 Kb/s durante a descriptografia. Esses números indicam que, apesar da robustez e segurança desses algoritmos, eles são penalizados por sua complexidade, resultando em menores taxas de transferência.

Essas observações destacam como tanto a latência quanto o tamanho do bloco influenciam significativamente a *Throughput* de dados. Algoritmos com blocos maiores podem processar mais dados por operação, resultando em uma *Throughput* mais alta, desde que a latência não aumente proporcionalmente. Além disso análise detalhada das taxas de transferência em ambos os dispositivos evidencia as capacidades do hardware. No caso do Arduino Nano, onde os recursos são mais limitados, cifras como a Speck são preferíveis devido à sua eficiência superior. No ESP-32, embora cifras mais complexas como SHA3-256 e Chacha20 sejam viáveis, elas exigem uma consideração cuidadosa do impacto em termos de desempenho e *Throughput*. Pois esta relação é crucial para a escolha de algoritmos criptográficos em aplicações que requerem alta eficiência de processamento de dados, garantindo que a comunicação e as operações sejam realizadas de forma ágil e eficiente.

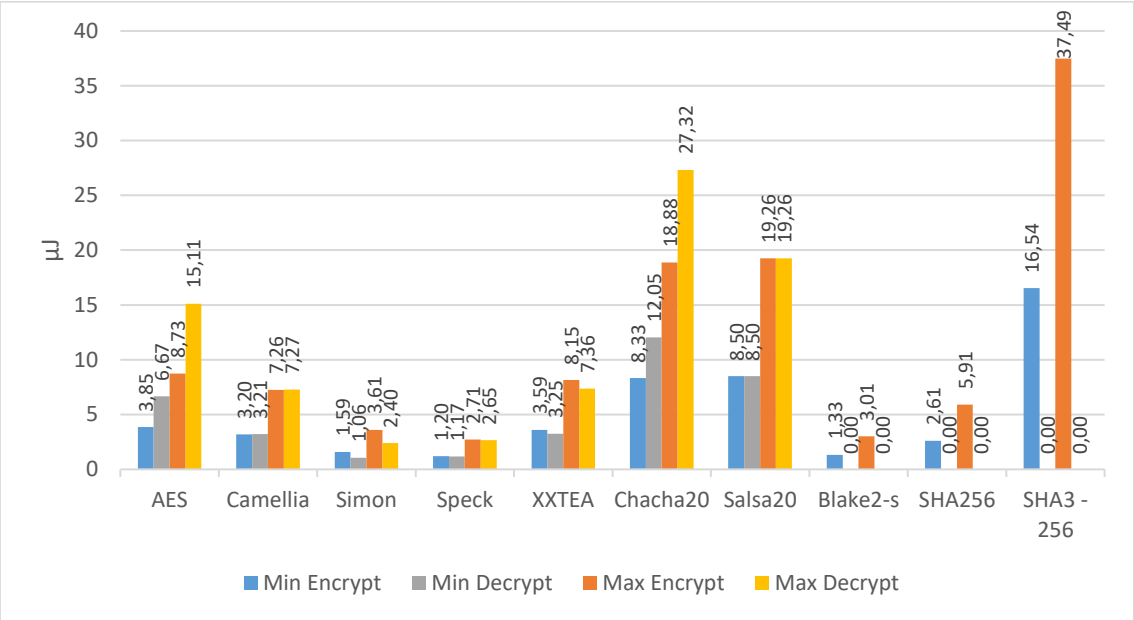
4.4 Custo Energético por bit

O consumo de energia por bit é um fator crucial na avaliação de cifradores leves, especialmente em dispositivos com recursos limitados, como os dispositivos IoT e sistemas embarcados. A eficiência energética impacta diretamente a vida útil da bateria e a viabilidade de operações contínuas. Sendo o consumo de energia por bit avaliado medindo a quantidade de energia necessária em micro Joules para processar cada bit de dados através da cifra. Podemos estimar a energia consumida por bit utilizando a Equação 2 mencionada anteriormente. Este cálculo fornece uma medida útil para comparar a eficiência energética dos algoritmos de criptografia, pois quanto menor a energia por bit, mais eficiente é o sistema em termos de consumo de energia por unidade de informação processada.

Cifras mais simples tendem a consumir menos energia por bit, enquanto cifras mais complexas podem oferecer maior segurança, mas com um custo energético mais

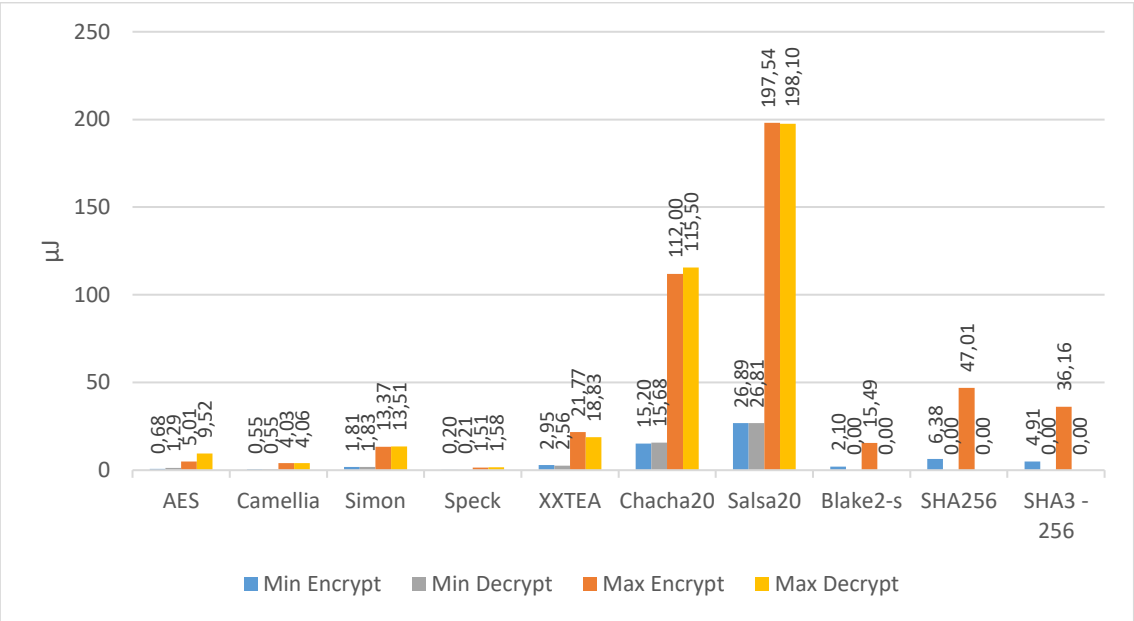
elevado. Portanto, é essencial encontrar um equilíbrio entre simplicidade e complexidade para otimizar tanto a segurança quanto a eficiência energética.

Figura 25 – Custo de Energia por bit no ESP32



Fonte: Própria Autoria

Figura 26 – Custo de Energia por bit no Arduino Nano



Fonte: Própria Autoria

Como não podemos medir diretamente a corrente e a tensão demandadas pelos processadores dos microcontroladores, utilizamos a Equação 2 para estimar o custo

energético. Essa equação baseia-se na potência mínima e máxima demandada pelos dispositivos, conforme especificado em seus datasheets e apresentado nas Tabelas 4 e 5.

Os valores obtidos são apresentados nas Figuras 25 e 26. No Arduino Nano, as cifras que apresentam maior consumo de energia são as cifras de fluxo e as funções de *hash* SHA-256 e SHA3-256. Onde a cifra Salsa20 consumindo até 198 µJ sendo 132 vezes mais energia do que a cifra Speck que consume 0,2 a 1,5 µJ, que se mostra a mais eficiente entre as implementadas no Arduino, tanto em modos de criptografia quanto de descriptografia.

No ESP-32, as cifras com maior consumo de energia são novamente as de fluxo e a função de *hash* SHA3-256. A SHA3-256, em particular, consome 16,54 a 37,49 µJ sendo aproximadamente o dobro da energia das cifras de fluxo. A cifra Speck, por outro lado, continua a se destacar por sua eficiência, consumindo entre 1,2 e 2,71 µJ.

Além disso, esses achados ressaltam a relevância de considerar tanto o tipo de cifra quanto a arquitetura do hardware ao projetar sistemas de criptografia. Por exemplo, enquanto a Speck se mostrou consistentemente eficiente em ambas as plataformas, cifras como SHA3-256 e as cifras de fluxo, apesar de sua robustez, apresentaram um consumo de energia significativamente maior, o que pode não ser ideal para todas as aplicações.

4.5 Eficiência do Algoritmo Criptográfico

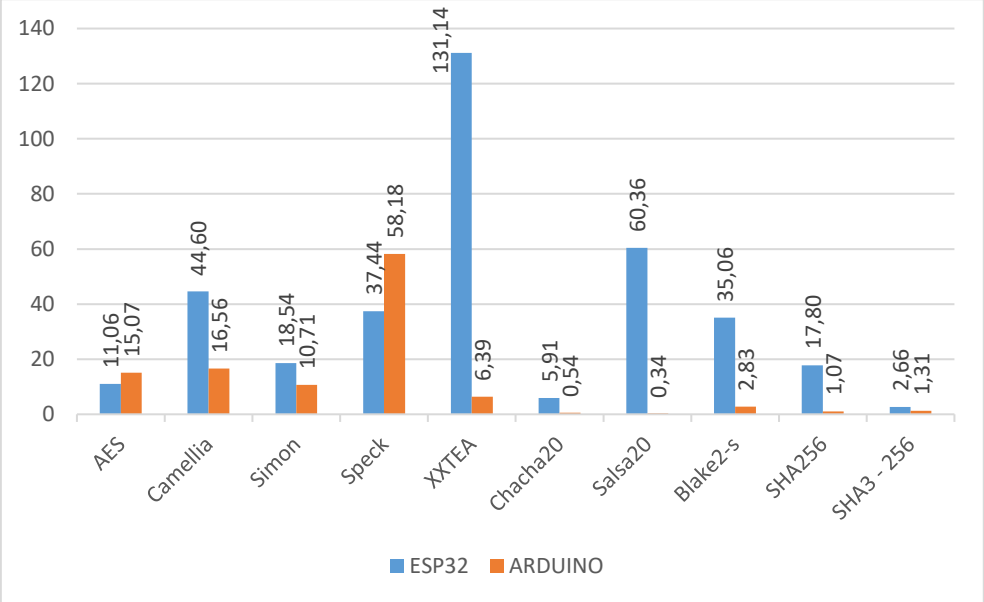
A eficiência de software em algoritmos criptográficos refere-se à eficácia com que esses algoritmos são implementados e executados em um microcontrolador. A Equação 3 é introduzida com o propósito de demonstrar a eficiência de software através da relação entre a taxa de transmissão e a quantidade de memória ROM alocada pelo algoritmo em KB. A rapidez com que um algoritmo pode cifrar ou decifrar dados é uma medida crítica de sua eficiência. Algoritmos que podem ser executados rapidamente são preferidos, especialmente em sistemas onde a latência é um fator importante. Além disso, algoritmos que utilizam menos recursos do sistema (CPU, memória) são considerados mais eficientes. Isso é especialmente relevante em dispositivos com recursos limitados, como microcontroladores e dispositivos IoT.

Algoritmos que podem processar dados rapidamente e utilizar eficientemente a memória disponível são essenciais para a operação eficaz de dispositivos IoT. Essa eficiência permite que os dispositivos executem tarefas criptográficas sem comprometer

outras funcionalidades críticas, mantendo a segurança e a performance. Portanto, ao desenvolver soluções criptográficas para microcontroladores, é fundamental considerar tanto a velocidade de execução quanto o uso de recursos para garantir uma implementação eficiente e eficaz.

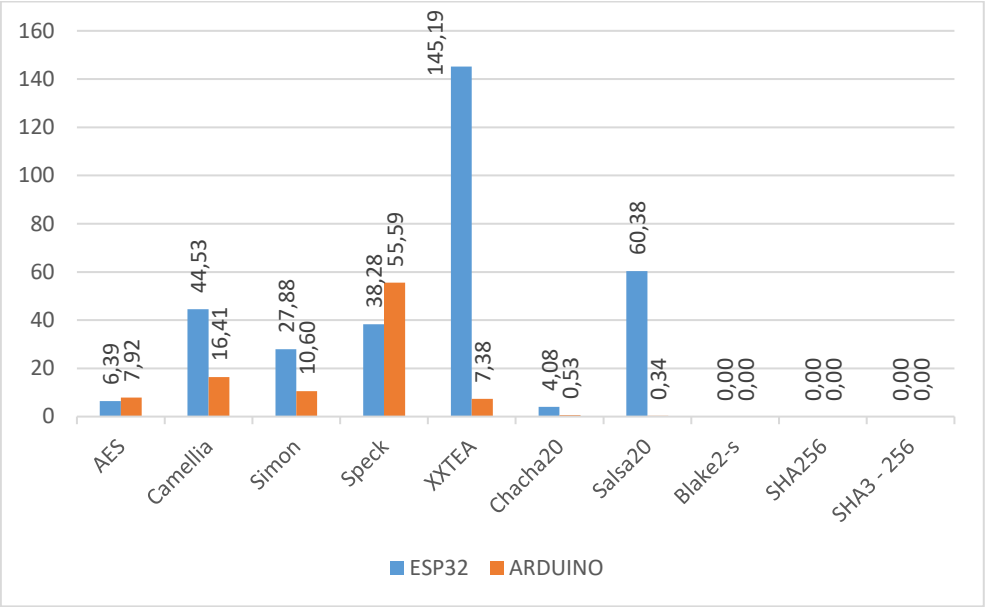
A eficiência de software em algoritmos criptográficos é uma medida multifacetada que considera a velocidade, o uso de recursos e a otimização do algoritmo. A combinação de alta taxa de transmissão e baixo consumo de memória resulta em um desempenho superior, garantindo que as operações criptográficas não se tornem um gargalo no sistema.

Figura 27 – Eficiência na Criptografia



Fonte: Própria Autoria

Figura 28 – Eficiência na Descriptografia



Fonte: Própria Autoria

Nas Figuras 27 e 28 apresentam a eficiência dos algoritmos criptográficos. Para os valores apresentados quanto maior melhor.

Um ponto interessante apresentado foi que as cifras AES e Speck se mostraram mais eficientes quando implementadas no Arduino, em vez do ESP-32. Isso ocorre principalmente devido à otimização na compilação, onde o algoritmo compilado para o Arduino resulta em um código com menos dados em comparação com o compilado para o ESP-32.

No Arduino, o algoritmo mais eficiente é o Speck, que combina uma alta *Throughput* com uma baixa alocação de memória ROM, superada apenas pelas cifras Simon e XXTEA. Speck se destaca por ser o mais eficiente tanto na criptografia quanto na descriptografia. Essa eficiência se deve ao design leve da cifra. Em contraste, as cifras de fluxo e as funções *hash*, como SHA-256 e SHA3-256, ocupam mais espaço na memória ROM e apresentam uma *Throughput* inferior às cifras de bloco. Esses algoritmos são mais complexos e exigem mais operações por bit de dados, resultando em maior consumo de recursos.

No ESP-32, a cifra que mais se destaca é a XXTEA, graças à sua baixíssima alocação de memória e alta *Throughput*, superando significativamente os outros algoritmos. Sua eficiência é mais de duas vezes superior ao segundo colocado, que é a

cifra Salsa20. A arquitetura de 32 bits do ESP-32 e sua maior capacidade de processamento permitem que a XXTEA opere de forma extremamente eficiente, aproveitando ao máximo os recursos disponíveis. Diferentemente do Arduino, no ESP-32, os algoritmos menos eficientes incluem não apenas as cifras de fluxo e as funções *hash*, mas também algumas cifras de bloco. Entre os cinco algoritmos com pior desempenho estão as cifras de bloco Simon e AES, a cifra de fluxo Chacha20 e as funções *hash* SHA-256 e SHA3-256.

Essa análise destaca a importância de considerar a plataforma de hardware ao implementar algoritmos criptográficos. Pois enquanto certas cifras podem ser extremamente eficientes em um dispositivo, elas podem não apresentar o mesmo desempenho em outro devido a diferenças na otimização e arquitetura do hardware.

4.6 Consumo Medido

Em dispositivos como sensores, atuadores e outros componentes de IoT que se utilizam de baterias ou fontes sazonais, a energia é um recurso limitado. E o consumo de energia por algoritmos criptográficos podem impactar diretamente na duração da bateria e a viabilidade de operações contínuas. Portanto, a eficiência energética de um algoritmo criptográfico é tão importante quanto sua segurança. Assim algoritmos que consomem menos energia permitem que os dispositivos funcionem por períodos mais longos, reduzindo a necessidade de manutenção e substituição de baterias.

Neste tópico, analisaremos o consumo em watts-hora dos dois dispositivos, seguindo o fluxograma apresentado na Figura 14 e utilizando o método de medição ilustrado na Figura 15. Avaliar o consumo de energia em criptografia envolve medir a energia consumida para realizar operações de cifragem e decifragem. Esta avaliação foi feita utilizando um sensor de medição de corrente e tensão para monitorar o consumo de energia dos dispositivos durante a execução dos algoritmos criptográficos. Para calcular o consumo de energia, podemos usar a seguinte fórmula básica:

$$Consumo = Potência * Tempo \qquad (4)$$

Onde:

Potência: É a quantidade de energia consumida por unidade de tempo (em watts, W).

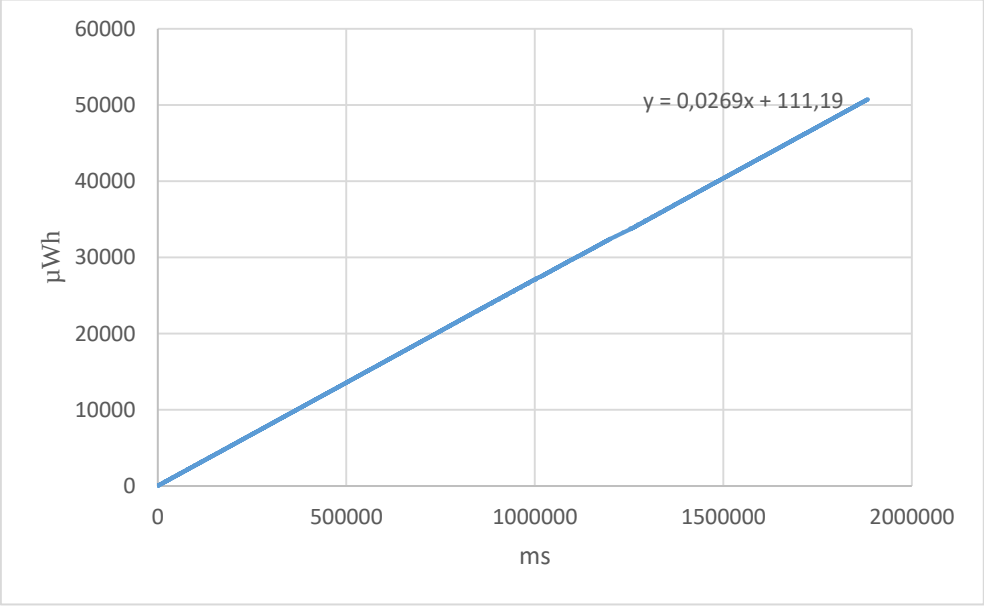
Tempo: É a duração da execução do algoritmo (em horas, h).

A potência pode ser obtida multiplicando a corrente I (em amperes, A) pela tensão V (em volts, V):

$$Potencia = V * I \quad (5)$$

Dessa forma, comparamos o consumo das cifras utilizando a mesma carga de trabalho. Conforme citado na Seção 3, os dados foram coletados através de medições realizadas por um período de cinco minutos, com uma taxa média de amostragem de 70 amostras por segundo. Além disso, realizamos uma medição de trinta minutos utilizando a cifra AES no ESP-32, conforme demonstrado na Figura 29. Essa medição teve o objetivo de demonstrar a linearidade do consumo utilizando o método apresentado no fluxograma da Figura 14, evidenciando que, independentemente do tempo de amostragem, a implementação fornece dados confiáveis.

Figura 29 – Consumo Medido na Cifra AES Implementado no ESP-32



Fonte: Própria Autoria

Nas Figuras 30 a 37, apresentamos uma amostra de 5 segundos das medições de consumo nos dois dispositivos, destacando as cifras Simon e SHA3-256 implementadas no ESP-32, que possuem a menor e maior latência, respectivamente. As Figuras 30 e 32 demonstram o aumento no consumo de energia, onde podemos observar claramente os momentos de operação, identificados pela inclinação acentuada da curva em direção ao

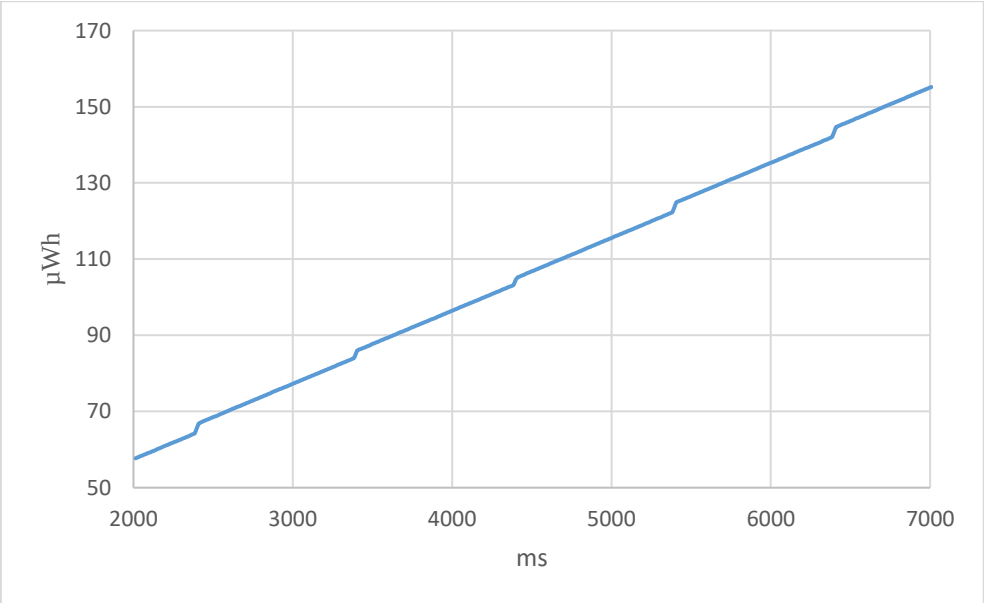
eixo vertical, e os momentos de *standby*, onde a curva tende a ficar próxima ao eixo horizontal.

Já as Figuras 31 e 33 apresentam a potência instantânea consumida. No caso da cifra Simon, o gráfico exibe um padrão de pulsos, refletindo sua operação intermitente. Em contraste, na *hash* SHA3-256, o gráfico se apresenta como uma mesa, indicando um consumo de energia mais constante e contínuo.

Nas Figuras 34 a 37, que apresentam as amostras do Arduino Nano, foi utilizada a cifra Speck, que obteve a menor latência, e a *hash* SHA256, com a segunda maior latência. Pois a *hash* SHA3-256, com a maior latência, saturou o teste mostrando apenas o modo ativo.

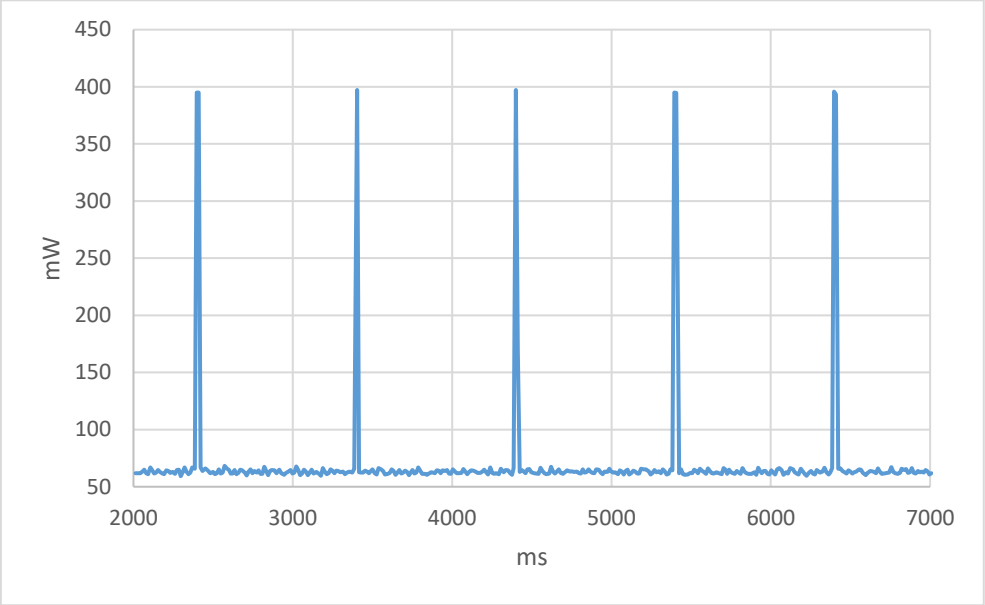
Os gráficos apresentados para o Arduino Nano comportam-se de maneira semelhante aos obtidos a partir do ESP-32. No entanto, a variação mais consistente nos gráficos do Arduino Nano se deve ao seu menor consumo de energia em comparação ao ESP-32.

Figura 30 – Amostra de 5 Segundos do Consumo da Cifra Simon no ESP-32



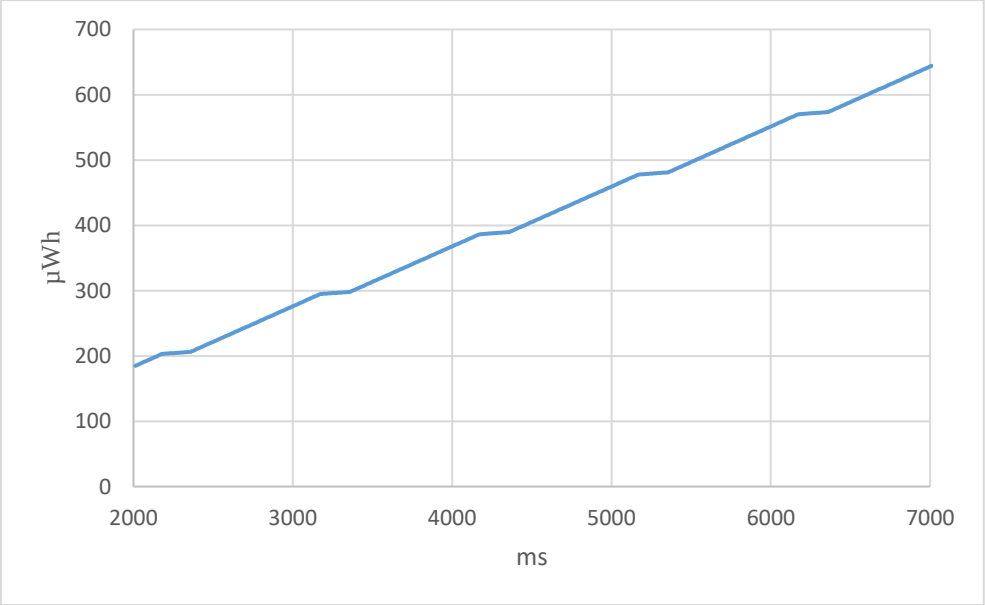
Fonte: Própria Autoria

Figura 31 – Amostra de 5 Segundos da Potência Demandada pela Cifra Simon no ESP-32



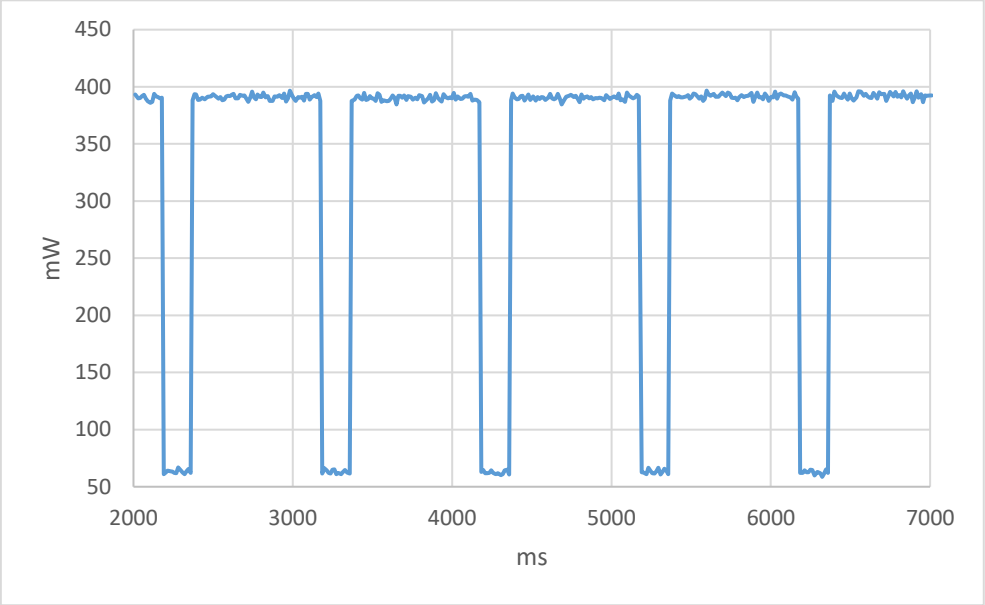
Fonte: Própria Autoria

Figura 32 – Amostra de 5 Segundos do Consumo da *Hash* SHA3-256 no ESP-32



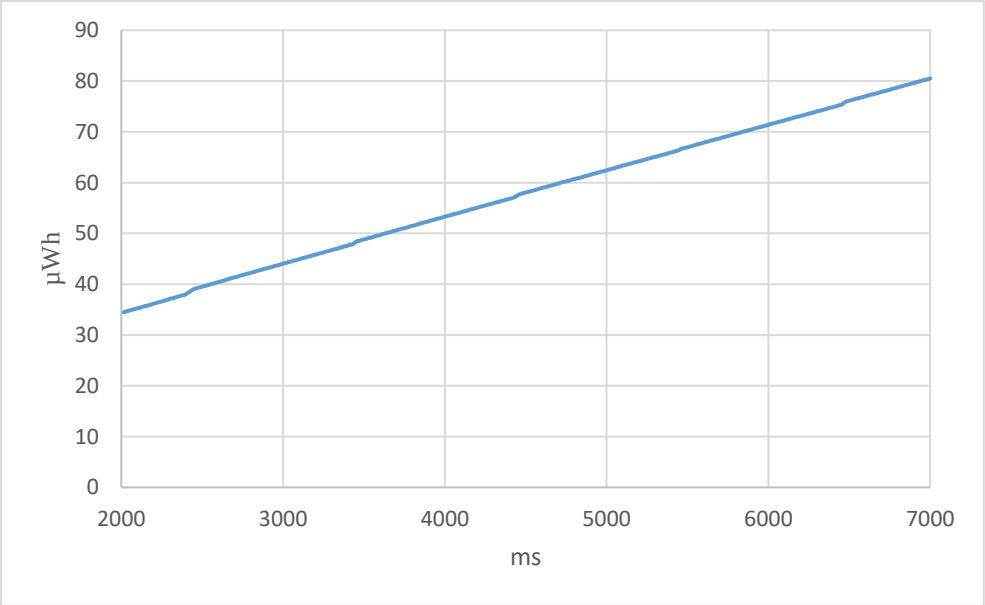
Fonte: Própria Autoria

Figura 33 – Amostra de 5 Segundos da Potência Demandada pela *Hash* SHA3-256 no ESP-32



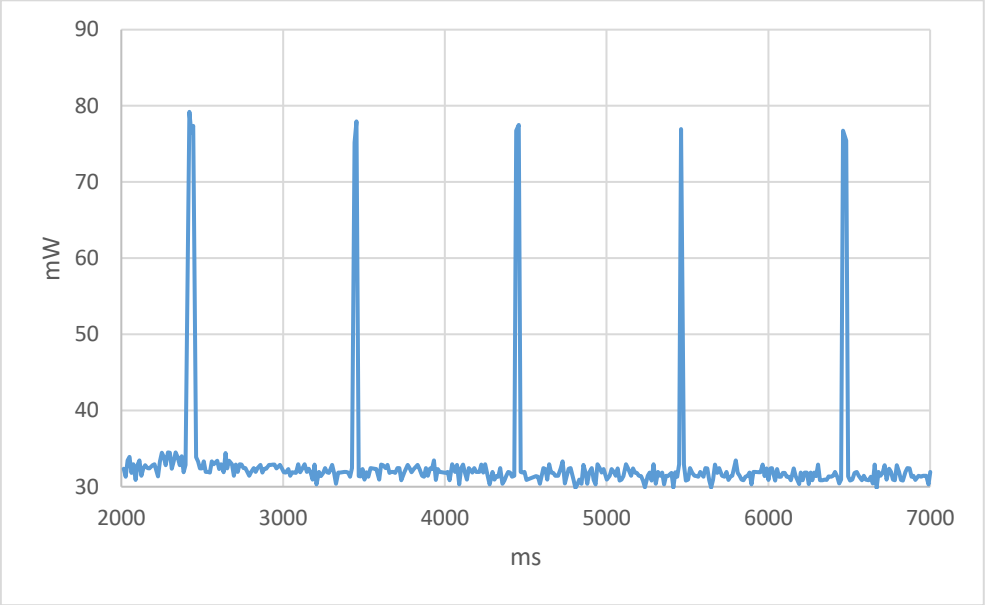
Fonte: Própria Autoria

Figura 34 – Amostra de 5 Segundos da Potência Demandada pela Cifra Speck no Arduino Nano



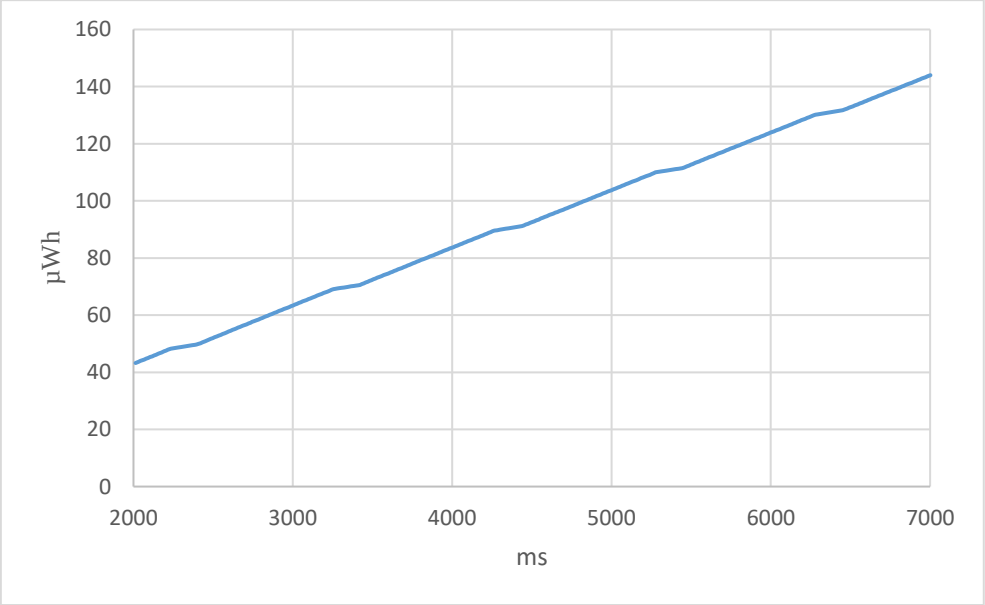
Fonte: Própria Autoria

Figura 35 – Amostra de 5 Segundos da Potência Demandada pela Cifra Speck no Arduino Nano



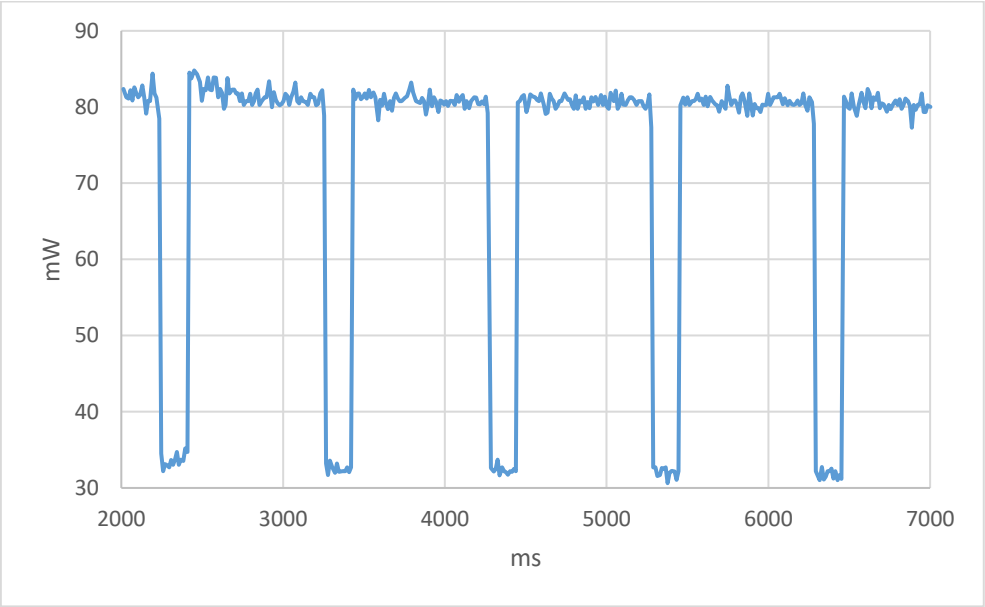
Fonte: Própria Autoria

Figura 36 – Amostra de 5 Segundos do Consumo da *Hash* SHA256 no Arduino Nano



Fonte: Própria Autoria

Figura 37 – Amostra de 5 Segundos do Consumo da *Hash* SHA256 no Arduino Nano



Fonte: Própria Autoria

Por fim, é apresentado nas Figuras 38 a 39 o consumo de energia de todas as cifras e funções *hash*. A Figura 38 mostram os resultados obtidos no ESP-32, enquanto a Figura 39 exibem os resultados do Arduino Nano.

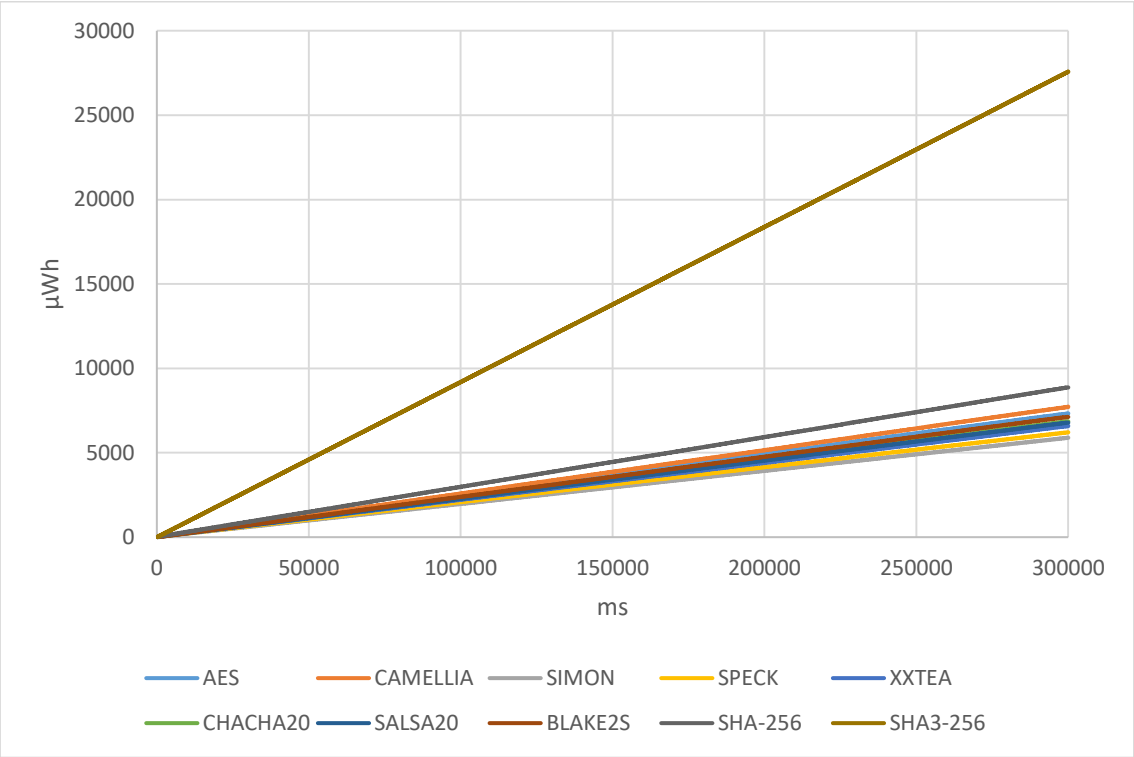
Pode-se observar, quando agrupado, o consumo mostra uma linearidade que pode ser aproximada por uma equação linear do tipo:

$$y = ax + b \qquad (6)$$

Onde a potência demandada está diretamente relacionada com o coeficiente angular da reta. Em outras palavras, quanto mais inclinada a reta, maior será o consumo de energia. Assim se olharmos para a inclinação das retas para o caso do ESP-32 veremos q a menos inclinação é da cifra Simon e a maior é da *hash* SHA3-256. Mas com exceção da SHA3, o coeficiente angula ‘a’ das cifras variam de 0,0190 a 0.0300 demonstrando que o consumo médio está bem próximo.

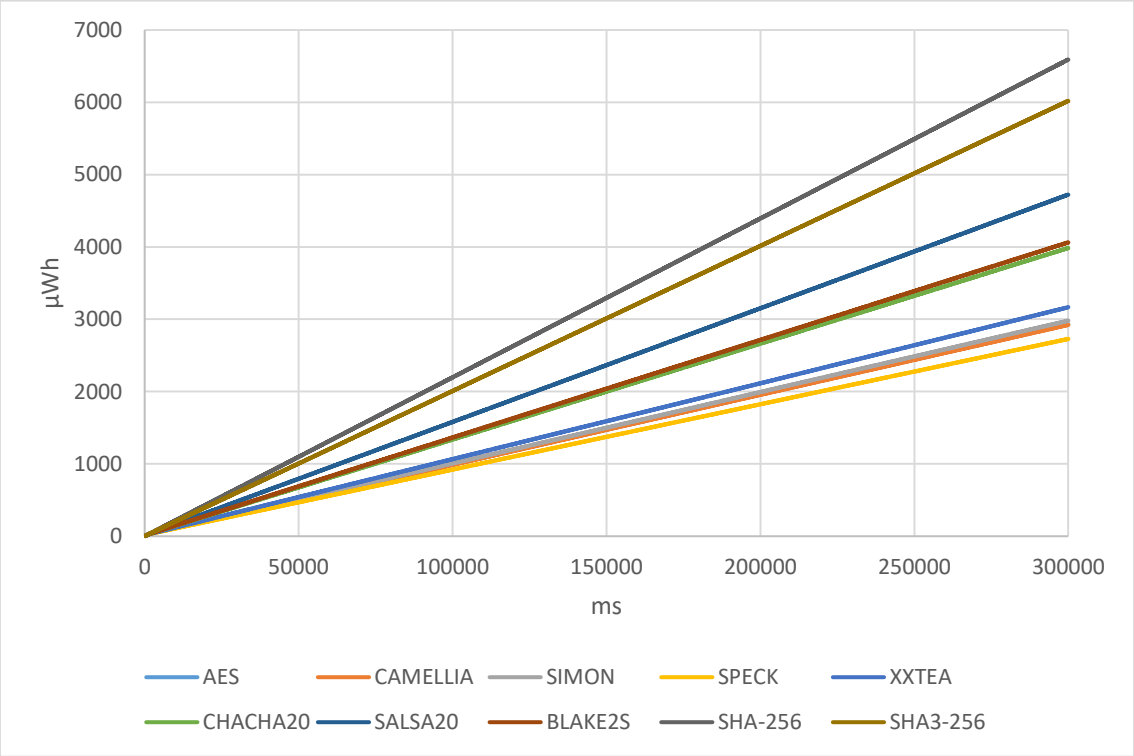
No Arduino Nano, essa característica é observada apenas nas cifras de bloco. A cifra Speck apresenta o menor consumo de energia, com um coeficiente angular de 0,009, enquanto a XXTEA tem o maior consumo entre as cifras de bloco, com um coeficiente angular de 0,0105. A cifra com o maior consumo de energia geral é a SHA256, que possui um coeficiente angular de 0,022.

Figura 38 – Consumo das Cifras e *Hashs* no ESP-32



Fonte: Própria Autoria

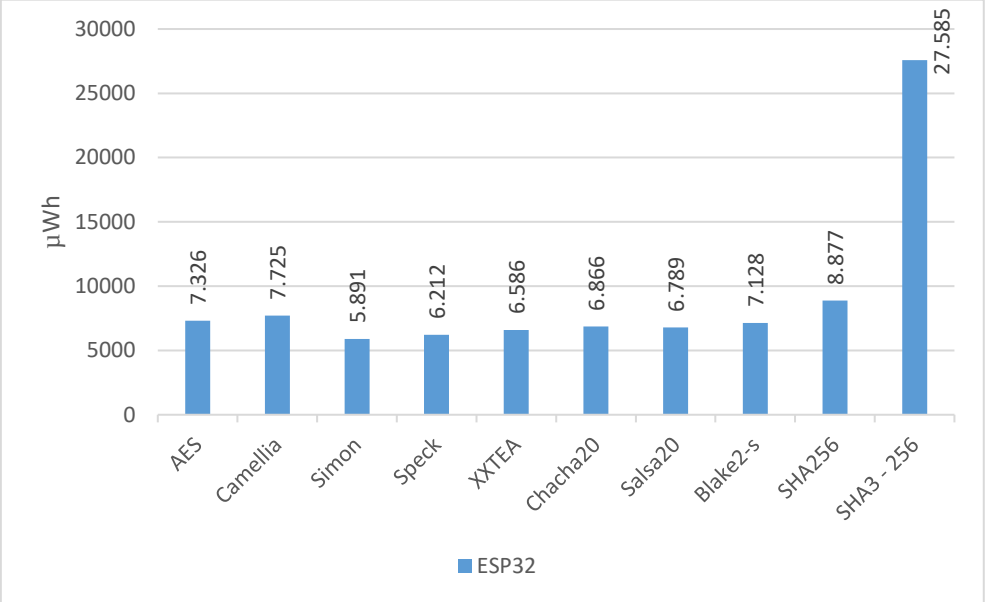
Figura 39 – Consumo das Cifras e *Hashs* no Arduino



Fonte: Própria Autoria

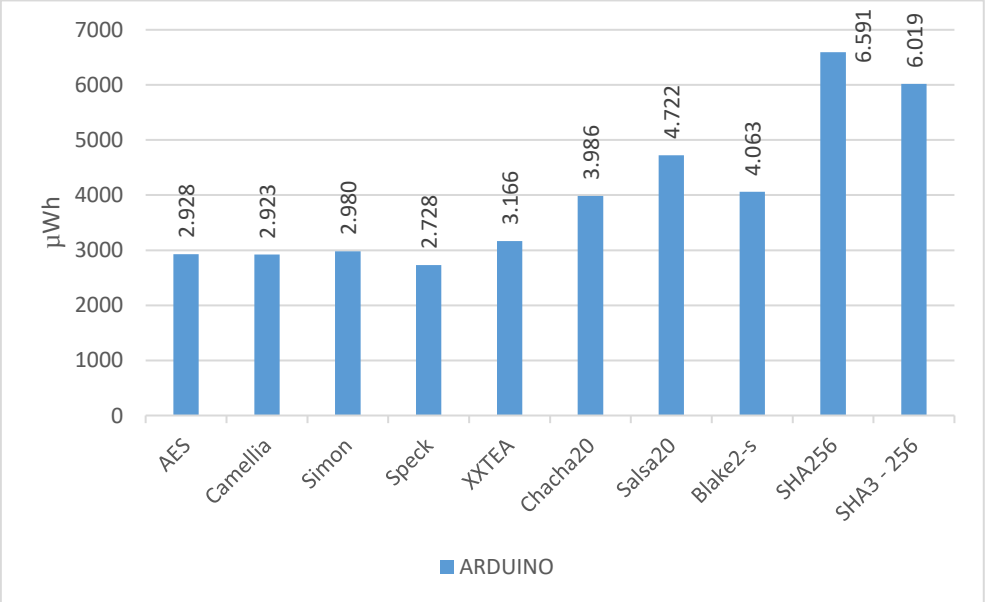
Ao final das medições obtivemos os seguintes valores de consumo que estão apresentados nas Figuras 40 e 41.

Figura 40 – Consumo Total no Decorrer de 5 Minutos no ESP-32



Fonte: Própria Autoria

Figura 41 – Consumo Total no Decorrer de 5 Minutos no Arduino



Fonte: Própria Autoria

Os valores apresentados revelam que, mesmo as cifras mais complexas, consomem relativamente pouca energia por ciclo de criptografia. Para efeito de comparação, consideremos a cifra com maior consumo geral, a SHA3-256, implementada

no ESP-32. Se utilizarmos essa cifra ao ritmo constante de 5000 operações de criptografia por segundo, ela levaria aproximadamente 7 horas para esgotar uma bateria de 9V com 250 mAh.

Esta análise destaca a eficiência energética das cifras mesmo em cenários de alta demanda. A capacidade de realizar um grande número de operações de criptografia sem comprometer significativamente a energia da bateria é crucial para a viabilidade de dispositivos IoT e sistemas embarcados. Por exemplo, para aplicações que requerem processamento contínuo e seguro de dados, a escolha de algoritmos com baixo consumo energético pode prolongar a vida útil do dispositivo e reduzir a necessidade de manutenção frequente.

Além disso, a proximidade nos valores de consumo entre as diferentes cifras indica que a escolha de uma cifra mais complexa, como a SHA3-256, não resulta em um impacto energético tão significativo quanto se poderia esperar. Isso sugere que, em muitas aplicações, é possível optar por cifras mais robustas sem comprometer drasticamente a eficiência energética do sistema.

Em resumo, os dados demonstram que a implementação de cifras criptográficas em dispositivos como o ESP-32 e o Arduino Nano pode ser realizada de forma eficiente em termos de energia. A escolha da cifra deve levar em conta não apenas a segurança, mas também a eficiência energética, para garantir a sustentabilidade e a funcionalidade prolongada dos dispositivos em operações contínuas.

5. CONSIDERAÇÕES FINAIS

Escolher adequadamente a cifra de criptografia é crucial para minimizar o consumo extra de energia em dispositivos IoT devido ao processamento da criptografia e escolher uma criptografia com baixo consumo de energia pode prolongar a vida útil da bateria e reduzir os custos operacionais em dispositivos alimentados por energia limitada, como dispositivos portáteis e sensores IoT. Mas embora a eficiência energética seja importante, a segurança dos dados também deve ser uma consideração primordial ao escolher uma cifra. Uma cifra que consome menos energia não deve comprometer a segurança dos dados. É essencial equilibrar eficiência energética com robustez e segurança.

Ao longo deste trabalho, ficou evidente que a escolha da cifra de criptografia desempenha um papel crucial no consumo de recursos em sistemas embarcados. É

necessário ponderar o custo-benefício de cada algoritmo. Em cenários onde é preciso enviar ou receber grandes volumes de dados, o uso de uma cifra com alta latência pode sobrecarregar o microcontrolador, impedindo-o de processar todos os dados de forma eficiente. Em sistemas que requerem respostas rápidas, a demora na criptografia pode causar perda de sincronismo, comprometendo a performance do sistema.

Além disso, em dispositivos IoT com baterias, o consumo adicional de energia devido ao processamento criptográfico pode reduzir a autonomia. Portanto, escolher uma cifra adequada que balanceie eficiência energética, latência e segurança é essencial.

As cifras Simon e Speck se destacam nas primeiras posições devido à sua criação voltada especificamente para dispositivos com recursos limitados. Essas cifras conseguem alinhar baixa latência com baixo consumo energético e ainda tem um ótimo nível de segurança, tornando-as ideais para sistemas embarcados e dispositivos IoT.

Em contraste, as cifras de fluxo e as funções *hash*, devido à sua maior complexidade, encontram-se entre as últimas em termos de eficiência energética e latência. No entanto, isso não impede seu uso em dispositivos com recursos limitados; elas podem ser empregadas em cenários onde uma alta *Throughput* não é crucial.

A comparação entre a cifra Camellia e a cifra XXTEA ilustra que a "modernidade" de uma cifra não é necessariamente indicativa de sua eficácia em ambientes embarcados. Embora a cifra Camellia seja mais recente que a XXTEA, sua otimização pode não ser ideal para dispositivos com recursos extremamente limitados.

Quando o foco é a segurança, a cifra AES é amplamente reconhecida por sua robustez e implementação difundida. Surpreendentemente, mesmo mantendo altos padrões de segurança, a AES apresenta eficiência energética comparável à das cifras Simon e Speck. Essa combinação de segurança robusta e eficiência torna a AES uma escolha sólida para muitas aplicações, equilibrando a necessidade de proteção de dados com o gerenciamento eficiente dos recursos do sistema.

Em resumo, a seleção da cifra de criptografia deve ser cuidadosa e levar em consideração vários fatores, incluindo eficiência energética, latência, adequação às limitações do hardware e necessidades de segurança. As cifras Simon e Speck, com sua baixa latência e consumo energético reduzido, são ideais para dispositivos limitados. Por outro lado, cifras mais complexas como as de fluxo e funções *hash* ainda têm seu lugar em aplicações específicas, desde que suas características sejam adequadamente alinhadas às necessidades do sistema. Portanto, é fundamental realizar uma análise cuidadosa das características e requisitos do sistema antes de selecionar a cifra de criptografia mais apropriada. Isso garante não apenas a segurança dos dados, mas também a eficiência e o desempenho adequado do sistema em operação.

BIBLIOGRAFIA

AIN, A. U.; KOUR, E. H. Lightweight cryptographic techniques in internet of things: A review. **IOSR Journal of Computer Engineering**, v. 23, n. 2, p. 1–14, 2021.

AOKI, K. *et al.* **Camellia: A 128-Bit block cipher suitable for multiple platforms — Design andAnalysis** (D. R. Stinson & S. Tavares, Eds.)Selected Areas in Cryptography. **Anais**... *In*: 7TH ANNUAL INTERNATIONAL WORKSHOP. Ontario, Canada: Springer Berlin Heidelberg, 2001Disponível em: <https://link.springer.com/chapter/10.1007/3-540-44983-3_4>. Acesso em: 12 nov. 2023

AQEEL, M. *et al.* A Review of Security and Privacy Concerns in the Internet of Things (IoT). **Journal of Sensors**, v. 2022, n. 1, p. 1–20, 29 set. 2022.

ATLAM, H. F.; WALTERS, R.; WILLS, G. Internet of Things: state-of-the-art, challenges, applications, and open issues. **International Journal of Intelligent Computing Research (IJICR)**, v. 9, n. 2042-4655, p. 928–938, 2018.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787–2805, out. 2010.

AUMASSON, J.-P. *et al.* **BLAKE2: simpler, smaller, fast as MD5www.blake2.net**. [s.l: s.n.]. Disponível em: <https://www.blake2.net/blake2.pdf>. Acesso em: 8 maio. 2024.

BEAULIEU, R. *et al.* **The SIMON and SPECK families of lightweight block ciphers**. Disponível em: <https://eprint.iacr.org/2013/404>.

BERNSTEIN, D. **The Salsa20 family of stream ciphers Cr.yp.to**. Chicago: The University of Illinois , 25 dez. 2007. Disponível em: <https://cr.yp.to/snuffle/salsafamily-20071225.pdf>. Acesso em: 5 maio. 2024.

___. **ChaCha, a Variant of Salsa20 Cr.yp.to**. Chicago: The University of Illinois , 20 jan. 2008. Disponível em: <https://cr.yp.to/chacha/chacha-20080128.pdf>. Acesso em: 5 maio. 2024.

BERTONI, G. *et al.* Keccak. **Advances in Cryptology – EUROCRYPT 2013**, v. 7881, p. 313–314, 2013.

BIRYUKOV, A.; WAGNER, D. Slide Attacks. **Fast Software Encryption**, v. 6th, n. 1, p. 245–259, 1 jan. 1999.

BKHEET, S. A.; AGBINYA, J. I. A Review of Identity Methods of Internet of Things (IOT). **Advances in Internet of Things**, v. 11, n. 04, p. 153–174, 2021.

BOSE, A. **edgarreis/xxtea-iot-crypt**. Disponível em: <https://github.com/edgarreis/xxtea-iot-crypt?tab=readme-ov-file>. Acesso em: 21 jun. 2024.

BRASIL. **LEI Nº 13.709, DE 14 DE AGOSTO DE 2018**. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em: 13 out. 2024.

chacha20 - Rust. Disponível em: <https://docs.rs/chacha20/0.9.1/chacha20/>. Acesso em: 23 out. 2024.

CHAN, H. Internet of Things Business Models Keywords Internet of Things (IoT), Business Model, Strategy and Tactic in IoT, Case Studies in IoT. **Journal of Service Science and Management**, v. 8, n. 1940-9907, p. 552–568, 2015.

CROWLEY, P. **Truncated differential cryptanalysis of five rounds of Salsa20**. [s.l: s.n.]. Disponível em: <https://www.ecrypt.eu.org/stream/papersdir/073.pdf>. Acesso em: 5 maio. 2024.

DAEMEN, J.; RIJMEN, V. **AES Proposal: Rijndael**. Gaithersburg: National Institute of Standards and Technology, 9 mar. 1999. Disponível em: <https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/documents/aes-development/Rijndael-ammended.pdf>. Acesso em: 25 maio. 2023.

___. **The Design of Rijndael**. Berlin: Springer Berlin Heidelberg, 2002.

DHANDA, S. S.; SINGH, B.; JINDAL, P. Lightweight Cryptography: A Solution to Secure IoT. **Wireless Personal Communications**, v. 112, n. 1947–1980, 25 jan. 2020.

DUTTA, I. K.; GHOSH, B.; BAYOUMI, M. **Lightweight cryptography for internet of insecure things: A survey**2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019. **Anais**...Institute of Electrical and Electronics Engineers Inc., mar. 2019

DWORKIN, M. *et al.* Advanced Encryption Standard (AES). **Federal Inf. Process. Stds. (NIST FIPS), Gaithersburg**, v. 197, 26 nov. 2001.

DWORKIN, M. J. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. **National Institute of Standards and Technology**, n. 20899-8900, jul. 2015.

ELIJAH, O. *et al.* An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. **IEEE Internet of Things Journal**, v. 5, n. 5, p. 3758–3773, out. 2018.

FALCHETTO, M. **mirkoflchtt/CamelliaLib**. Disponível em: <https://github.com/mirkoflchtt/CamelliaLib>.

FOTOVVAT, A. *et al.* Comparative performance analysis of lightweight cryptography algorithms for IoT sensor nodes. **IEEE Internet of Things Journal**, v. 8, n. 10, p. 8279–8290, maio 2021.

GIGLI, M.; KOO, S. Internet of Things: Services and Applications Categorization. **Advances in Internet of Things**, v. 01, n. 02, p. 27–31, 2011.

GODOI, M. G.; ARAÚJO, L. S. A INTERNET DAS COISAS: Evolução, Impactos e Benefícios. **Revista Interface Tecnológica**, v. 16, n. 1, p. 19–30, 30 jun. 2019.

GONÇALVES, E. L. **Políticas de Segurança da Informação**. 1. ed. Rio de Janeiro: Escola Superior de Redes, 2015.

GREENGARD, S. **Internet of Things | electronic network**. Disponível em: <https://www.britannica.com/science/Internet-of-Things>. Acesso em: 20 abr. 2024.

GUSMAO, G.; BRITO, P. **MODELO DE INTERNET DAS COISAS PARA O PARQUE ESTADUAL DO CANTÃO**. [s.l: s.n.].

HAMEED, R.; HUSSEIN, A.; KHALAF, B. A light-weight ESalsa20 ciphering based on 1D logistic and chebyshev chaotic maps. **Solid State Technology**, v. 63, n. 1, set. 2020.

HANSEN, T.; EASTLAKE 3RD, D. E. **US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)**. Disponível em: <https://datatracker.ietf.org/doc/html/rfc6234>. Acesso em: 5 maio. 2024.

HATZIVASILIS, G. *et al.* A review of lightweight block ciphers. **Journal of Cryptographic Engineering**, v. 8, n. 2, p. 141–184, jun. 2018.

HAWKES, P.; PADDON, M.; ROSE, G. **On Corrective Patterns for the SHA-2 Family**. [s.l: s.n.]. Disponível em: <https://eprint.iacr.org/2004/207.pdf>. Acesso em: 5 maio. 2024.

HINTZBERGEN, J. *et al.* **Fundamentos de Segurança da Informação: com base na ISO 27001 e na ISO 27002**. 3. ed. [s.l.] Brasport, 2018. v. único

HOLDSWORTH, J.; KOSINSKI , M. **O que é a segurança da informação? | IBM**. Disponível em: <https://www.ibm.com/br-pt/topics/information-security>. Acesso em: 14 out. 2024.

JAYAKUMAR, V. *et al.* Multicriteria group decision making for prioritizing IoT risk factors with linear diophantine fuzzy sets and MARCOS method. **Granular Computing**, v. 9, n. 3, 2 maio 2024.

JOSEFSSON, S.; JOACHIM STROMBERGSON; NIKOS MAVROGIANNOPOULOS. **The salsa20 stream cipher for transport layer security (TLS) and datagram transport layer security (DTLS)**. [s.l.] Internet Engineering Task Force, mar. 2013. Disponível em: <https://datatracker.ietf.org/doc/draft-josefsson-salsa20-tls/01/>.

KAMAL, R. **Internet of things : architecture and design principles**. Chennai, India: Mc Graw Hill India, 2017.

KIKUCHI, A. Y.; BETTERI, L. A.; ARAÚJO, L. S. IOT – INTERNET DAS COISAS. **Revista Interface Tecnológica**, v. 18, n. 2, p. 118–130, 14 maio 2022.

KUMAR, S.; TIWARI, P.; ZYMBLER, M. Internet of Things is a revolutionary approach for future technology enhancement: a review. **Journal of Big Data**, v. 6, n. 1, 9 dez. 2019.

MACHADO, J. S. *et al.* UM ESTUDO DOS ALGORITMOS DE CRIPTOGRAFIA LEVE PARA DISPOSITIVOS IOT. **Revista Expressão Científica (REC)**, v. 6, n. 2, p. 71–86, 27 out. 2021.

MADAKAM, S.; RAMASWAMY, R.; TRIPATHI, S. Internet of things (IoT): A literature review. **Journal of Computer and Communications**, v. 03, n. 05, p. 164–173, 2015.

MANCINI, M. **A história da Internet das Coisas ou Internet of Things (IoT)**. Disponível em: <https://www.linkedin.com/pulse/hist>. Acesso em: 13 out. 2024.

MAY, W. FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS) CATEGORY: COMPUTER SECURITY SUBCATEGORY: CRYPTOGRAPHY. **National Institute of Standards and Technology**, v. 4, n. 20899-8900, 2015.

MENEZES, A. J.; OORSCHOT, P. C.; VANSTONE, S. A. **Handbook of applied cryptography**. Boca Raton: Crc Press, 1996.

MERCHANT, N. **IoT Technologies Explained: History, Examples, Risks & Future**. Disponível em: <https://www.visionofhumanity.org/what-is-the-internet-of-things/>. Acesso em: 20 mar. 2024.

MOHD, B. J.; HAYAJNEH, T.; VASILAKOS, A. V. A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues. **Journal of Network and Computer Applications**, v. 58, p. 73–93, dez. 2015.

MOUHA, R. A. Internet of things (IoT). **Journal of Data Analysis and Information Processing**, v. 09, n. 02, p. 77–101, 2021.

NASEER, M. *et al.* **S-box Security Analysis of NIST Lightweight Cryptography Candidates: A Critical Empirical Study**. Disponível em: <https://ar5iv.labs.arxiv.org/html/2404.06094>. Acesso em: 17 out. 2024.

NEEDHAM, R.; WHEELER, D. **Tea extensions**. Inglaterra: Cambridge University, 1997. Disponível em: <https://www.club.cc.cmu.edu/~ajo/docs/xtea.pdf>. Acesso em: 30 mar. 2023.

OKELLO, W. J. *et al.* **A survey of the current state of lightweight cryptography for the Internet of things**2017 International Conference on Computer, Information and Telecommunication Systems (CITS). **Anais**...IEEE, jul. 2017

OLIVEIRA, S. **Internet das Coisas com ESP8266, Arduino e Raspberry Pi 2a edição**. 2. ed. [s.l.] Novatec Editora, 2021. v. 1

PERERA, C. *et al.* A Survey on Internet of Things From Industrial Market Perspective. **IEEE Access**, v. 2, n. 2169-3536, p. 1660–1679, 8 jan. 2015.

PIAZZA, M. L. **O Futuro da IoT: O Que Esperar na Revolução Conectada**. Disponível em: <https://www.weg.net/digital/blog/o-futuro-da-iot-o-que-esperar-na-revolucao-conectada/>. Acesso em: 15 out. 2024.

RASHIDI, B. Flexible and high-throughput structures of Camellia block cipher for security of the Internet of Things. **IET Computers & Digital Techniques**, v. 15, n. 3, p. 171–184, maio 2021.

RENO, R. **O caminho promissor da IoT no Brasil**. Disponível em: <https://abinc.org.br/o-caminho-promissor-da-iot-no-brasil/>. Acesso em: 13 out. 2024.

RODRIGUES, J. J. P. C. *et al.* Enabling Technologies for the Internet of Health Things. **IEEE Access**, v. 6, n. 2169-3536, p. 13129–13141, 2018.

SAARINEN, M. Cryptanalysis of block tea. 20 nov. 1998.

SANTOS, B. *et al.* **Internet das Coisas: da Teoria à Prática**. [s.l: s.n.]. Disponível em: <https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf>. Acesso em: 15 out. 2024.

SETHI, P.; SARANGI, S. R. Internet of Things: Architectures, Protocols, and Applications. **Journal of Electrical and Computer Engineering**, v. 2017, n. 9324035, p. 1–25, 26 jan. 2017.

SINHA, S. **State of IoT 2021: Number of Connected IoT Devices Growing 9% to 12.3 Billion globally, Cellular IoT Now Surpassing 2 Billion**. Disponível em: <https://iot-analytics.com/number-connected-iot-devices/>.

SLEEM, L.; COUTURIER, R. Speck-R: An ultra light-weight cryptographic scheme for Internet of Things. **Multimedia Tools and Applications**, v. 80, n. 11, p. 17067–17102, maio 2021.

TAWALBEH, L. *et al.* IoT Privacy and Security: Challenges and Solutions. **Applied Sciences**, v. 10, n. 12, p. 4102, 15 jun. 2020.

TEICHER, J. **The little-known story of the first IoT device**. Disponível em: <https://www.ibm.com/blog/little-known-story-first-iot-device/>. Acesso em: 20 mar. 2024.

THAKOR, V. A.; RAZZAQUE, M. A.; MUHAMMAD. **Lightweight cryptography for IoT: A state-of-the-art**. Disponível em: <http://arxiv.org/abs/2006.13813>.

VAILSHERY, L. S. **Number of IoT connected devices worldwide 2019-2030, by vertical**. Disponível em: <https://www.statista.com/statistics/1194682/iot-connected-devices-vertically/>. Acesso em: 3 jun. 2024.

WEATHERLEY, R. **Arduino Cryptography Library**. Disponível em: <https://rweather.github.io/arduinolibs/crypto.html>.

WHEELER, D. J.; NEEDHAM, R. M. TEA, a tiny encryption algorithm. **Fast Software Encryption**, v. 1, n. 0302-9743, p. 363–366, dez. 1995.

WHEELER, D. J.; NEEDHAM, R. M. Correction to Xtea. out. 1998.

WIKIPEDIA CONTRIBUTORS. **XXTEA**. Disponível em: <https://en.wikipedia.org/wiki/XXTEA>. Acesso em: 20 out. 2024.

YARRKOV, E. **Cryptanalysis of XXTEA**. Disponível em: <https://eprint.iacr.org/2010/254>. Acesso em: 20 abr. 2024.

ZANELLA, A. *et al.* Internet of Things for Smart Cities. **IEEE Internet of Things Journal**, v. 1, n. 1, p. 22–32, fev. 2018.