Uma Avaliação Comparativa de Algoritmos de Criptografia Simétrica para Dispositivos Restritos em Recursos

Mayksuel Ramalho[1], Gabriel S. de Oliveira[1], Nicholas Neves[1], Rafael Porto[2], Victor Afonso M. Sobral[1], Marcos Rezende[3], Dianne S. V. Medeiros[1]

[1]Departamento de Engenharia de Telecomunicações (TET)
[2]Departamento de Engenharia Agrícola e Meio Ambiente (TER)
Universidade Federal Fluminense (UFF)
[3] Prefeitura Municipal de Niterói

{mayksuelramalho,gabrielsampaio,diannescherly}@id.uff.br

Abstract. *Data security in the Internet of Things is crucial for protecting both devices and the data they transmit over the network. Nevertheless, security is often overlooked in this context, leaving systems vulnerable to cyber attacks that can compromise information confidentiality and integrity. This work focuses on a use case of a remote environmental variable monitoring system for disaster prevention, in which information must be confidential and intact. The AES (Advanced Encryption Standard) and Speck cryptographic algorithm families are evaluated in both traditional and memory-optimized implementations, targeting data confidentiality. The algorithms are assessed through practical experiments on a resource-constrained hardware platform. Transmission throughput, estimated energy consumption, execution time, and memory usage are evaluated. Results show that the Speck family executes more quickly, has lower estimated energy consumption, and occupies less memory space.*

Resumo. *A segurança de dados na Internet das Coisas é fundamental para proteger tanto os dispositivos quanto os dados que trafegam na rede. No entanto, a segurança é normalmente negligenciada nesse contexto, tornando os sistemas vulneráveis a ataques cibernéticos, o que pode comprometer a confidencialidade e integridade das informações. Este trabalho foca em um caso de uso de um sistema de monitoramento remoto de variáveis ambientais para prevenção de desastres naturais, cujas informações devem ser confidenciais e íntegras. As famílias de algoritmos AES (Advanced Encryption Standard) e Speck criptográficos são avaliadas em implementações tradicionais e otimizadas em uso de memória, visando a confidencialidade dos dados. Os algoritmos são avaliados por meio de experimentos práticos em uma plataforma de* hardware *restrita em recursos computacionais e de energia. Verifica-se a vazão da transmissão, a estimativa de consumo de energia, o tempo de execução e a ocupação de memória. Os resultados mostram que a família Speck executa mais rapidamente e apresenta o menor consumo de energia estimado ao mesmo tempo em que ocupa menos espaço em memória.*

1. Introdução

A Internet das Coisas (*Internet of Things* - IoT) representa uma revolução na forma como as pessoas interagem com o mundo [Mouha e Ait, 2021]. A IoT oferece uma infraestrutura para coleta remota de dados em tempo real e automação de processos

[Hossein Motlagh et al., 2020]. Contudo, o uso crescente de dispositivos IoT em sistemas de monitoramento remoto expõe os dados desses sistemas a ameaças de segurança, como acesso não autorizado e interceptação de dados. A falta de segurança pode levar a uma abertura da rede para agentes maliciosos, que podem capturar os dados monitorados e modificá-los para prejudicar o sistema. Muitos desses sistemas se concentram na coleta e transmissão de dados ambientais, mas é necessário também se preocupar com as vulnerabilidades provocadas pelo uso de protocolos e políticas de segurança fracos [Tawalbeh et al., 2020], que prejudicam a proteção dos dados, tornando-os vulneráveis a ataques e comprometendo a confiabilidade.

Em cenários de desastres naturais, por exemplo, a falta de sistemas robustos de alerta precoce para eventos climáticos extremos é uma preocupação global. O monitoramento inadequado pode resultar em danos significativos à propriedade, perda de vidas humanas e impactos socioeconômicos devastadores. Portanto, é imperativo desenvolver soluções eficazes e seguras para mitigar os riscos associados a tais desastres [Alvalá e Barbieri, 2017]. A precisão e integridade dos dados provenientes desses sistemas são cruciais para embasar um processo de tomada de decisão eficiente visando a segurança das pessoas em áreas de risco. A falta de medidas robustas de segurança coloca em risco a eficácia dos sistemas de monitoramento e a segurança e o bem-estar das pessoas afetadas. Dessa forma, a vulnerabilidade dos sistemas de monitoramento remoto deve ser mitigada desde o início do projeto do sistema.

O desenvolvimento de sistemas IoT seguros é desafiador devido, principalmente, às restrições de recursos de energia e computacionais dos dispositivos. Essas restrições dificultam a adoção de protocolos seguros que envolvam mecanismos de criptografia robustos [Hoffmann e Griebler, 2023]. Apesar dessa dificuldade, é primordial integrar aos sistemas IoT técnicas criptográficas, de forma a garantir a confidencialidade e integridade dos dados transmitidos em sistemas de monitoramento remoto. Contudo, as técnicas criptográficas adicionam uma sobrecarga que pode prejudicar o funcionamento do sistema, seja por requerer uma capacidade computacional além da disponível, seja por aumentar o consumo de energia, reduzindo o tempo de vida médio do sistema, ou por aumentar a latência total da comunicação. Nesse cenário, é essencial buscar algoritmos criptográficos leves para incorporar nos sistemas IoT. Este trabalho foca em um caso de uso de monitoramento remoto para prevenção de desastres naturais, no qual dispositivos IoT são distribuídos estrategicamente para coletar e transmitir dados ambientais em tempo real e de forma segura. Sendo fundamental a confidencialidade e integridade dos dados coletados e armazenados para composição de uma série histórica que permita a criação de políticas públicas eficazes, este trabalho avalia o impacto do uso de algoritmos criptográficos no desempenho da comunicação no sistema de monitoramento.

Diversos trabalhos na literatura realizam uma avaliação abrangente comparando diversas técnicas criptográficas utilizando múltiplas plataformas [Pereira et al., 2017, Fotovvat et al., 2021] ou simulação [Beg et al., 2019]. Outros trabalhos avaliam o desempenho dessas técnicas simulando as plataformas de *hardware* Arduino Mega e Raspberry Pi 3 [Panahi et al., 2021], enquanto outros focam em avaliar algoritmos criptográficos leves implementados nas plataformas de *hardware* Arduino UNO e Raspberry Pi 3 [El-hajj et al., 2023]. Este trabalho adiciona uma camada de segurança à transmissão de dados em um sistema de monitoramento remoto para prevenção de desastres naturais,

incorporando técnicas de criptografia tradicionais e leves para transmissão dos dados a fim de garantir confidencialidade. O diferencial deste trabalho está na avaliação comparativa entre um algoritmo de criptografia leve, Speck, e um tradicional, AES (*Advanced Encrytion Standard*), considerando as implementações originais e otimizadas de ambos os algoritmos, disponíveis em biblioteca pública. A comparação é feita por meio de experimentos práticos utilizando a plataforma Arduino UNO. As versões otimizadas dos algoritmos AES e Speck têm o objetivo de consumir menos memória no dispositivo e são denominadas "Tiny" e "Small". Verifica-se a influência do uso de cada algoritmo e do tamanho da chave de criptografia no impacto do desempenho do sistema em termos de tempo para criptografia, vazão, consumo de energia, ocupação da memória SRAM (*Static Random-Access Memory*) e da memória *flash*. Os resultados mostram que não há variação na vazão da transmissão devido ao uso da criptografia simétrica. O consumo de energia estimado para o algoritmo Speck com chave de 128 bits é menor do que o dos demais e a ocupação em memória é baixa, apesar de ser maior do que a das versões otimizadas.

O restante deste trabalho está organizado como segue. A Seção 2 discute os trabalhos relacionados. A Seção 3 introduz os algoritmos de criptografia utilizados. A Seção 4 apresenta a metodologia de avaliação. A Seção 5 discute os resultados obtidos. Por fim, a Seção 6 conclui este trabalho e apresenta direções de pesquisa futura.

2. Trabalhos Relacionados

A segurança de sistemas IoT é um tema importante, dada a expansão rápida das aplicações relacionadas ao paradigma IoT. No entanto, a aplicação de técnicas criptográficas nesses sistemas é desafiadora devido às restrições computacionais e de energia dos dispositivos que fazem parte do sistema. Diversos trabalhos avaliam o desempenho de algoritmos criptográficos com a finalidade de criar um *benchmark* para que os melhores algoritmos possam ser escolhidos para implementação nos diversos sistemas. Guinelli et al. realizam uma análise comparativa entre os algoritmos de criptografia Rijndael e Serpent, considerando como métricas de avaliação: consumo de armazenamento, ciclos de relógio, consumo de energia, consumo de memória e tempo de execução [Guinelli et al., 2018]. A ideia é determinar em que cenário e para qual requisito cada algoritmo é mais adequado. Os algoritmos são implementados em uma plataforma Arduino Mega 2560 e os resultados mostram que o algoritmo Rijndael apresenta vantagens significativas em termos de eficiência computacional, consumindo menos ciclos de relógio e tempo de processamento em comparação com o Serpent, especialmente em cenários de recursos limitados. No entanto, o Serpent consome menos memória SRAM e exibe um consumo de energia inferior durante a execução dos algoritmos, sugerindo que é mais adequado para aplicações com restrições de memória e energia. Não há comparação com o algoritmo Speck.

Zanon et al. abordam a vulnerabilidade dos dispositivos vestíveis de monitoramento cardíaco, especialmente aqueles integrados à Internet das Coisas Médicas. Os autores focam em um cenário de transmissão segura de dados por dispositivos vestíveis durante exames cardíacos [Zanon et al., 2022]. Também ressaltam a necessidade de garantir a segurança dos dados transmitidos por dispositivos vestíveis durante exames cardíacos, devido à sensibilidade e ao valor desses dados. Os autores implementam uma camada de segurança baseada em criptografia leve e avaliam o desempenho de diferentes algoritmos de criptografia nesse contexto. Os algoritmos avaliados são: AES-256 CBC (*Cipher Block Chaining*) implementado em *hardware* no microcontrolador ESP32 e dois algorit-

mos de criptografia leve baseados em *software*, Speck e CLEFIA (Cipher for Lightweight Encryption Fast with Integer Approximation). Os autores levam em consideração o desempenho e a facilidade de implementação para cada algoritmo. O desempenho é avaliado em termos de vazão, latência e consumo de energia. Os resultados mostram que o AES-256 CBC tem bom desempenho em segurança e eficiência de energia, mas com tempo de execução relativamente maior. Speck e CLEFIA, como algoritmos de criptografia leve, apresentam menor consumo de energia e tempo de execução. Os autores não avaliam as implementações otimizadas dos algoritmos AES e Speck.

Albarello et al. avaliam o desempenho de três diferentes algoritmos que possuem finalidades distintas: AES, SHA e x25519, em dispositivos IoT com recursos limitados [Albarello et al., 2020]. Além disso, os autores propõem um protocolo que utiliza o algoritmo x25519 para trocar chaves de forma segura entre dispositivos IoT em uma rede. Esse protocolo permite a comunicação segura entre os dispositivos sem a necessidade de inserir chaves previamente nos dispositivos, tornando-o eficiente em termos de tempo de execução e segurança. Os resultados mostram que o Arduino Uno R3 executou satisfatoriamente os algoritmos AES e SHA, mesmo tendo um baixo processamento. O protocolo proposto para troca de chaves baseado no algoritmo x25519 demonstrou também ser viável e eficaz, sendo adequado para comunicação segura entre dispositivos IoT e um *gateway*. Os autores não avaliam algoritmos leves para criptografia, como o Speck.

Vaz et al. propõem uma otimização para o algoritmo AES de forma a adequá-lo para execução em dispositivos com recursos computacionais restritos [Vaz et al., 2023]. Os autores modificam duas funções do algoritmo para utilizar uma matriz de substituição menor do que a do algoritmo original e operações menos custosas para gerar a matriz de estados. As otimizações são concentradas principalmente nas operações de SubBytes que substitui cada byte do bloco de texto claro por outro byte usando uma tabela de substituição pré-definida, e MixColumns combinando cada coluna do bloco de texto cifrado usando uma matriz de coeficientes fixos para misturar os bytes. A ideia é melhorar o tempo de execução e o consumo de memória. Os autores comparam a proposta com o algoritmo original utilizando um microcontrolador ARM Cortex-M0+ baseado no chip SAM D21 da Microchip Technology e verificam que a proposta reduz o tempo de execução médio em 86,71%, o consumo de memória de programa em 31,82% e o de memória dinâmica em 89,04%. Os autores não comparam a proposta com outros algoritmos otimizados para uso em dispositivos restritos em recursos.

Fotovvat et al. avaliam de forma abrangente o desempenho de 32 algoritmos criptográficos leves nas plataformas IoT Raspberry Pi 3, Raspberry Pi Zero W e iMX233 [Fotovvat et al., 2021]. Apesar da grande variedade de algoritmos testados, os algoritmos Speck e AES não são avaliados. Os algoritmos são avaliados em termos de uso de memória, consumo de energia e tempo de execução. Os resultados mostram que o tempo de execução e o consumo de energia variam muito entre os algoritmos, mas o consumo de memória e de processamento se mantêm semelhantes. Pereira et al. avaliam implementações de referência de diversos algoritmos de criptografia simétrica, função resumo criptográfico, códigos de autenticação de mensagens e criptografia autenticada com dados associados. Os algoritmos são avaliados em duas plataformas IoT, TelosB e Intel Edison, e três sistemas operacionais IoT populares, Yocto, ContikiOS e TinyOS [Pereira et al., 2017]. A avaliação comparativa é feita em termos de tempo de

execução e consumo de energia. Dentre os algoritmos de criptografia simétrica avaliados, o AES possui menor tempo de execução e consumo de energia na plataforma TelosB executando o sistema operacional TinyOS ou ContikiOS. Já para a plataforma Intel Edison executando o sistema operacional Yocto, a criptografia AES possui menor tempo de execução e consumo de memória apenas se as mensagens forem menores do que 40 Bytes.

El-hajj et al. avaliam o desempenho de algoritmos de criptografia simétrica nas plataformas Arduino UNO e Raspberry Pi 3 em termos de velocidade, custo e eficiência energética. Dentre os algoritmos avaliados estão o AES e o Speck [El-hajj et al., 2023]. Contudo, a implementação AES avaliada não é otimizada para redução de consumo de memória. Os resultados mostram que os algoritmos da família Speck estão entre os de melhor desempenho em relação ao consumo de energia, velocidade e consumo de memória. Panahi et al. avaliam o desempenho de 10 algoritmos criptográficos leves, dentre os quais está o AES [Panahi et al., 2021]. A avaliação é feita utilizando as plataformas Arduino MEGA 2560 e Raspberry Pi 3. A ideia é comparar o uso de memória, a eficiência energética, a vazão e o tempo de execução dos algoritmos. Os resultados mostram que o AES apresenta bom desempenho, mas não é o algoritmo de menor consumo de energia e menor tempo de execução. Ademais, na plataforma Raspberry Pi 3, é o algoritmo com maior uso de memória RAM e na plataforma Arduino Mega é o que possui maior consumo de memória ROM (*Read-Only Memory*).

Diferentemente dos trabalhos citados, este artigo compara os algoritmos AES e Speck em suas versões tradicionais e otimizadas, variando o tamanho da chave de criptografia. A avaliação é feita por meio de experimentos práticos em uma plataforma Arduino UNO R3, que é limitada em termos de processamento e armazenamento.

3. Algoritmos de Criptografia Simétrica AES e Speck

Os algoritmos criptográficos avaliados neste trabalho compõem duas famílias de algoritmos de criptografia simétrica, AES e Speck. A criptografia simétrica utiliza uma única chave que serve tanto para criptografar quanto para descriptografar os dados, sendo menos complexa do que a criptografia assimétrica. AES é conhecido por sua robustez e segurança, com diferentes tamanhos de chave, que permitem ajustar o nível de segurança à disponibilidade de recursos computacionais. AES opera separando blocos do texto em claro que são submetidos a rodadas alternadas de substituições e permutações. O tamanho dos blocos pode variar entre 128, 192 e 256 bits, que representam o tamanho das chaves. Em cada rodada de criptografar, são executadas as funções de substituição de bytes (SubBytes), deslocamento de linhas (ShiftRows), mistura de colunas (MixColumns) e adição de chave da rodada (AddRoundKey). Para descriptografar, são realizadas as operações inversas, exceto para a operação AddRoundKey, sendo a primeira operação ShiftRows, em vez de SubBytes. As últimas rodadas tanto na criptografia quanto na descriptografia não envolvem a operação MixColumns. As operações executadas envolvem a manipulação de uma matriz de bytes que é modificada a cada rodada. O número de rodadas varia dependendo do tamanho da chave. Por exemplo, para chaves de 128 bits, são realizadas 10 rodadas, enquanto para chaves de 256 bits, são 14 rodadas. As operações SubBytes, ShiftRows e MixColumns transformam a matriz de acordo com funções não lineares e lineares. A etapa AddRoundKey realiza uma operação XOR entre a matriz e a chave da rodada [Abdullah, 2017]. Existem variantes do AES otimizadas para dispositivos com recursos limitados. Algumas dessas variantes possuem implementações

de referência em bibliotecas disponibilizadas publicamente. Neste trabalho, utilizam-se duas variantes que otimizam o consumo de memória, denominadas "Tiny" e "Small", que buscam melhorar o desempenho e reduzir o uso de recursos do dispositivo. A otimização "Tiny" visa minimizar o tamanho do código ou a quantidade de memória necessária, simplificando o algoritmo ou utilizando técnicas de compactação. Por outro lado, a otimização "Small" procura equilibrar eficiência e uso de recursos, mantendo um compromisso aceitável entre desempenho e quantidade de memória necessária.

A família de algoritmos Speck foi projetada especificamente para dispositivos com recursos limitados, provendo segurança ao mesmo tempo e eficiência no consumo de recursos. É um algoritmo de criptografia de bloco leve que opera em blocos de dados de tamanho fixo, geralmente 64 ou 128 bits. Consiste em uma série de iterações de substituição e permutação, em que os dados são transformados repetidamente usando a chave secreta. O algoritmo utiliza uma estrutura de Feistel modificada, que divide o bloco de entrada em duas metades e aplica operações de substituição e permutação em cada metade alternadamente, combinando-as no final de cada iteração. O Speck é altamente otimizado para implementação em *hardware* e oferece uma boa relação entre segurança e eficiência [de Paiva et al., 2023]. Mesmo a família Speck tendo como foco dispositivos limitados em recursos, também existem as variantes "Small" e "Tiny" disponíveis em bibliotecas públicas, com os mesmos objetivos citados para as variantes AES [de Paiva et al., 2023].

A criptografia AES já foi submetida a um extenso escrutínio de criptoanalistas. Speck é mais recente e ainda precisa ser extensivamente testada. Considerando a chave de 256 bits, a margem de segurança do AES é nula, visto que as 14 rodadas necessárias foram quebradas em 2011 [Bogdanov et al., 2011]. Já na família Speck, o ataque mais efetivo revelou uma margem de segurança de 26% [Song et al., 2016].

4. Metodologia

Este trabalho está inserido no contexto de um projeto financiado pela Prefeitura Municipal de Niterói, cujo objetivo é desenvolver um sistema de monitoramento remoto para prevenção de desastres naturais capaz de transmitir dados de monitoramento ambiental de forma segura. Utilizam-se dispositivos limitados em recursos para construir sensores remotos que capturam os dados ambientais, criptografa esses dados e os transmite para um servidor de armazenamento e processamento de dados por meio de um *gateway* IoT.

Os sensores remotos utilizam como *hardware* a plataforma Arduino UNO R3, integrada a um módulo RTC modelo DS 3231, pluviômetro modelo DAVIS 6464M, e um módulo LoRa (*Long Range*) E32TTL100D para comunicação de baixa potência e longo alcance. A confidencialidade dos dados transmitidos utiliza um algoritmo de criptografia simétrica. O *gateway* IoT é composto por um módulo LoRa E32TTL100D e um módulo RTC modelo DS 3231 integrados a um Arduino UNO R3 que se conecta a um Raspberry Pi 4 Model B via porta serial. O Arduino UNO R3 que compõe o *gateway* IoT é responsável por receber os dados, descriptografá-los e enviá-los para o Raspberry Pi 4, que, por sua vez, é responsável pela comunicação com o servidor através do protocolo MQTTS (*Message Queuing Telemetry Transport Secured*). A Figura 1 mostra o esquema da comunicação entre o sensor remoto e o servidor por meio do *gateway* IoT. Este trabalho foca na avaliação dos algoritmos de criptografia utilizados para comunicação entre o sensor remoto e o *gateway* IoT, conforme destacado pela linha tracejada na figura.

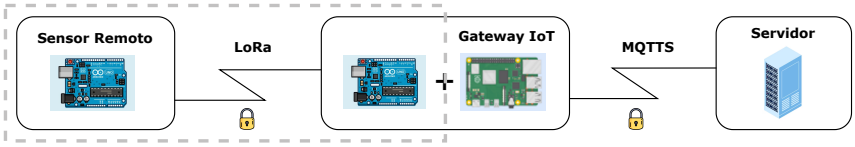


Figura 1. Esquema da comunicação. O sensor e o *gateway* se comunicam por meio de um módulo LoRa, com camada adicional de segurança para criptografia. A linha tracejada destaca a comunicação foco deste trabalho.

Avalia-se os algoritmos Speck e AES com diferentes tamanhos de chave e versões otimizadas. Utilizam-se as implementações de referência existentes na biblioteca `crypto`[1], desenvolvida para a plataforma Arduino. A família AES é composta pela implementação tradicional, AES, e otimizadas, AES Small e AES Tiny. Da mesma forma, a família Speck possui a versão tradicional, Speck, e as versões otimizadas, Speck Small e Speck Tiny. Utilizam-se chaves de 128, 192 e 256, sempre que possível. Destaca-se que as variantes "Tiny" não implementam função para descriptografia.

Os dados criptografados representam um pacote a ser transmitido utilizando o módulo LoRa. Esse pacote contém quatro diferentes tipos de variável que representam os dados ambientais coletados pelo sensor remoto. Cada variável tem 4 B, totalizando 16 B de dados a serem enviados. Os dados são transmitidos a cada 5 s e verificam-se a vazão da comunicação, tempo médio de execução, consumo de energia estimado, consumo de memória *flash* e consumo de memória SRAM. Na memória *flash* são armazenados o código, os arquivos das bibliotecas importadas, constantes literais e *strings*, enquanto na memória SRAM é armazenado temporariamente o código do programa em execução, variáveis, funções, e a pilha de execução do código. Dessa forma, a memória SRAM é a memória dinâmica, utilizada para execução do código. Em estado de repouso, o microcontrolador com todos os dispositivos que compõem o sensor remoto, utilizam em torno de 6 μA de corrente, considerando um tempo de observação de 10 segundos. A corrente é medida utilizando um multímetro Minipa ET-1400. O consumo de energia pode ser estimado por $v \times i \times \Delta t$, em que v é a tensão oferecida, i é a corrente consumida e Δt é o tempo de observação. Considerando que $v = 12\ V$, $i = 6\ \mu A$ e $\Delta t = 10/3600\ h$ há um consumo de energia estimado de $0{,}20\ \mu Wh$. O pico de consumo ocorre quando o dado é transmitido pelo módulo de comunicação, alcançando $i = 14\ \mu A$ em média durante $\Delta t = 0{,}480/3600\ h$ e, portanto, com uma estimativa de consumo de $0{,}02\ \mu Wh$.

5. Resultados e Discussão

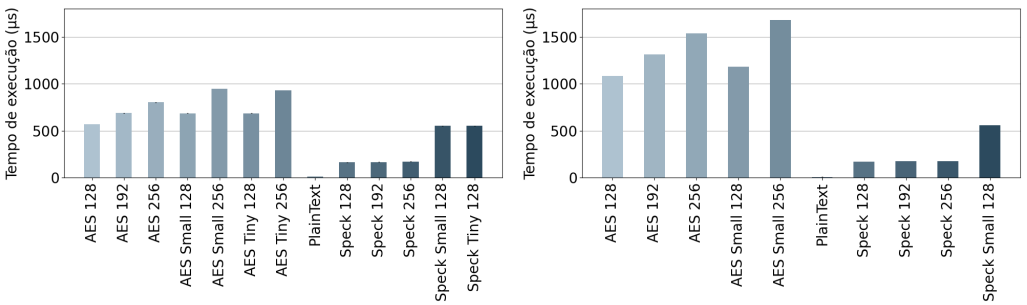
A comunicação entre o sensor remoto e o gateway IoT é avaliada para determinar o melhor algoritmo criptográfico para o cenário de monitoramento remoto estudado. É importante destacar que o módulo de comunicação LoRa utilizado implementa apenas a camada física, diferentemente de módulos LoRaWAN. Enquanto o protocolo LoRaWAN utiliza a criptografia AES para transmissão de dados, não há, por padrão, criptografia dos dados na camada física LoRa. Neste trabalho, os dados a serem transmitidos são transformados em um fluxo de bytes previamente criptografados para que sejam enviados pelo módulo LoRa. É importante destacar que em todos os resultados obtidos ao avaliar o receptor, as versões Tiny dos algoritmos são omitidas porque não há implementação da função de descriptografia, não sendo possível avaliar as métricas investigadas.

[1]Disponível em https://rweather.github.io/arduinolibs/crypto.html

Primeiramente, avalia-se a vazão, que representa a taxa de transferência de dados entre os dispositivos. Essa métrica é crucial para avaliar a capacidade de cada algoritmo em lidar com um fluxo contínuo de informações. Neste trabalho, a vazão é medida em pacotes por segundo. A vazão alcançada para transmissão dos dados é indiferente à criptografia usada, sendo igual a 3,4 pacotes por segundo. Isso ocorre porque o tempo necessário para execução dos algoritmos criptográficos é muito menor do que o intervalo entre envio de pacotes, não influenciando na quantidade total de pacotes enviados.

O tempo de execução dos algoritmos impacta diretamente na latência do sistema e no consumo de energia do dispositivo. A Figura 2 mostra o tempo de execução dos algoritmos. O algoritmo Speck 128 leva em média $162{,}4\ \mu s$ para criptografar ou descriptografar os dados. Ao utilizar a chave de 192 bits e 256 bits, observa-se um crescimento de $5{,}6\ \mu s$ e $9{,}6\ \mu s$ respectivamente. Mesmo com esse pequeno aumento, os algoritmos Speck sem otimização são 3 vezes mais rápidos do que as variantes Small e Tiny. Isso mostra que, apesar de ter o objetivo de reduzir a ocupação em memória, as implementações otimizadas elevam o tempo de execução. O algoritmo AES tradicional também apresenta tempos de execução menores do que suas versões Small e Tiny para tamanhos de chaves correspondentes. Observa-se também que quanto maior a chave utilizada, mais tempo os algoritmos levam para executar. Na descriptografia, não são mostrados os intervalos de confiança pois nas 5 rodadas realizadas, não houve variação no tempo de execução. É interessante notar que a descriptografia é um processo que leva mais tempo do que a criptografia para os algoritmos da família AES, enquanto para a família Speck o tempo necessário para ambas as ações é muito semelhante. Para o AES 128, o tempo de criptografia é de $569\ \mu s$, aumentando para $686\ \mu s$ com o AES 192 e $802\ \mu s$ com o AES 256. Apesar de haver uma variação grande entre os tempos de execução dos algoritmos, a influência desse tempo na vazão dos dados utilizando LoRa é insignificante. Dessa forma, quaisquer algoritmos seriam adequados. Caso seja necessário minimizar o tempo de execução, o algoritmo Speck com chave de 128 bits é o mais adequado.

Avalia-se também o tempo gasto para inicializar o dispositivo, preparando-o para transmissão ou recepção e desconsiderando as operações criptográficas. Essa avaliação é importante para verificar a influência dos algoritmos de criptografia na latência do sistema. A avaliação revela que no transmissor o tempo para inicialização é de aproximadamente $104\ ms$, enquanto o receptor necessita de aproximadamente $100\ ms$. O tempo



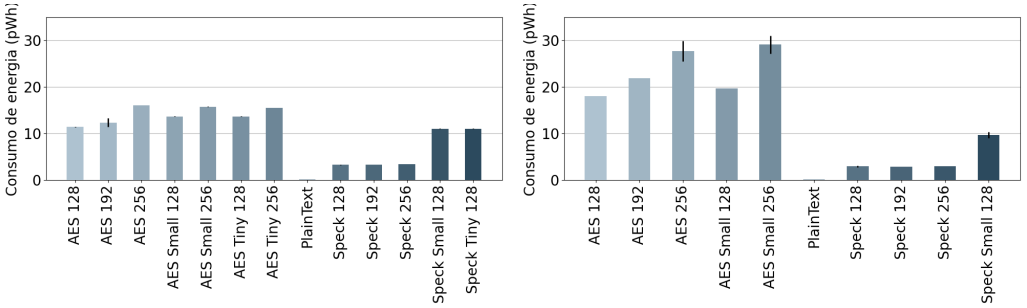
(a) Transmissor, criptografia.

(b) Receptor, descriptografia.

Figura 2. Tempo para criptografar e descriptografar. A família Speck supera a família AES, com diferenças de tempo significativas.

de inicialização e de execução dos algoritmos criptográficos influenciam diretamente a latência total da comunicação. Contudo, comparado ao tempo de inicialização, o tempo de execução dos algoritmos é desprezível. Essa avaliação corrobora o resultado obtido para a vazão, que é constante independentemente do algoritmo utilizado.

A estimativa de consumo de energia é feita com base em observações da corrente durante o tempo de criptografia, no transmissor, e descriptografia, no receptor. O experimento é repetido 5 vezes e o tempo de observação é extremamente curto, da ordem de microssegundos, de forma que muitas vezes não há variação da corrente durante o tempo observado. Por essa razão, alguns resultados não apresentam intervalo de confiança. O consumo de energia é extremamente baixo para todos os algoritmos, sendo ligeiramente menor para a operação de descriptografia. A menor estimativa obtida na criptografia, conforme mostra a Figura 3(a), ocorre para o algoritmo Speck 128, sendo que o uso de chaves de 192 bits e 256 bits nesse algoritmo aumenta levemente a estimativa do consumo de energia. Para a família AES, a menor estimativa é obtida para o algoritmo AES com chave de 128 bits. Na Figura 3(b), observa-se que a menor estimativa de consumo para descriptografar os dados é obtida também para o algoritmo Speck 128. Na família AES, a menor estimativa ocorre para o algoritmo tradicional com chave de 128 bits. Deve-se notar que o resultado para os algoritmos Speck Tiny 128 e AES Tiny 128 são omitidos porque ambos não implementam uma função de descriptografia. O resultado comprova a eficiência energética do algoritmo Speck para criptografar os dados. É interessante destacar que os algoritmos otimizados para uso de memória aumentam o consumo de energia em relação às implementações originais. Devido ao consumo estimado ser extremamente baixo, na ordem de picowatt-hora, qualquer um dos algoritmos poderia ser aplicado em um sistema restrito em energia. Contudo, caso seja necessário restringir o consumo ao máximo, o algoritmo ideal é o Speck com chave de 128 bits. Considerando a bateria de menor capacidade comumente usada em IoT, CR2032 de 3 V e 225 mAh [Smith et al., 2020], o tempo de vida do transmissor seria de aproximadamente 578×10^6 anos, e do receptor, 786×10^6 anos. Considerando apenas o uso da criptografia, os algoritmos Speck tradicionais reduzem o tempo de vida do transmissor para 0,04% do original, e do receptor para 3,4%. No transmissor, as versões otimizadas em memória e a família AES caem para 0,01%. No receptor, a versão Speck Small 128 cai para 1%, enquanto AES 128 e AES Small ficam em torno de 0,5%, AES 192 e AES 256 em 0,4%, e AES Small 256 em 0,3%.



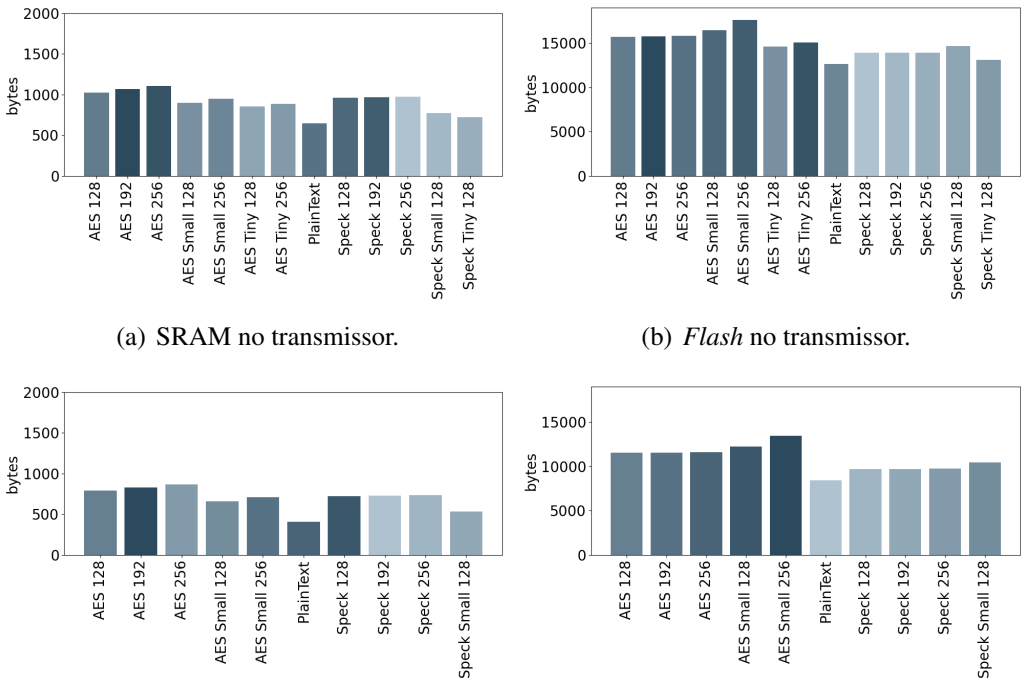
(a) Transmissor, criptografia.

(b) Receptor, descriptografia.

Figura 3. Estimativa de consumo de energia. Todos os algoritmos apresentam baixo consumo, sendo o menor consumo médio obtido para o Speck 128, considerando transmissor e receptor.

Em relação à ocupação da memória, avalia-se tanto a memória *flash* quanto a memória SRAM, a fim de verificar o impacto dos algoritmos no espaço de armazenamento disponível nos dispositivos. Destaca-se que essa ocupação é referente a todo o código, incluindo inicialização, criptografia e transmissão, no transmissor, e inicialização, recebimento e descriptografia, no receptor. A Figura 4 mostra os resultados obtidos. A ocupação da memória *flash* tende a ser menor para a família de algoritmos Speck, sendo minimizada na transmissão para o algoritmo Speck Tiny com chave de 128 bits. Já para a família AES, a menor ocupação é obtida para o algoritmo AES Tiny com chave de 128 bits. O algoritmo Speck Small com chave de 128 bits apresenta a segunda menor ocupação de memória tanto no transmissor quanto no receptor. Considerando a memória SRAM, a ocupação é minimizada no transmissor ao se utilizar o algoritmo Speck Tiny com chave de 128 bits. O algoritmo Speck Small com chave de 128 bits apresenta a segunda menor ocupação de memória tanto no transmissor quanto no receptor. O pior desempenho em termos de consumo de memória SRAM é obtido para o algoritmo AES com chave de 256 bits. Na família AES, o algoritmo com melhor desempenho é o AES Tiny com chave de 128 bits. Considerando que as variações Tiny dos algoritmos de criptografia não possuem uma função de descriptografia própria, no cenário de dispositivos restritos em recursos computacionais é mais vantajoso utilizar o algoritmo Speck Small. É válido destacar que a ocupação de memória *flash* e SRAM para o algoritmo Speck Small 128 é muito próxima da ocupação observada para o cenário em que não há criptografia (PlainText), reforçando a otimização de uso de memória dessa implementação do algoritmo.

Considerando todos os resultados obtidos, em um cenário de restrição de recursos



(a) SRAM no transmissor.

(b) *Flash* no transmissor.

(c) SRAM no receptor

(d) *Flash* no receptor.

Figura 4. Total de memória ocupada. A menor ocupação ocorre para Speck Tiny 128 no transmissor e Speck Small 128 no receptor. A ocupação para esses algoritmos é semelhante à do código sem criptografia (PlainText).

computacionais e de energia, o algoritmo criptográfico que apresenta o melhor desempenho em termos de consumo de memória para criptografar é o Speck Small com chave de 128 bits, enquanto para descriptografar é o Speck tradicional com chaves de 128, 192 ou 256 bits. Já o algoritmo com menor estimativa de consumo de energia é o Speck 128. A Tabela 1 resume os melhores algoritmos segundo cada métrica.

Tabela 1. Resumo dos melhores algoritmos por métrica.

	Consumo de energia	**Tempo de execução**	**Consumo de SRAM**	**Consumo de *flash***	**Vazão**
Transmissão	Speck 128	Speck 128	Speck Tiny 128	Speck Tiny 128	empate
Recepção	Speck 128	Speck 128	Speck Small 128	Speck 128/192/256	empate

6. Conclusão

Este trabalho avaliou o desempenho de algoritmos de criptografia simétricas em *hardware* limitado em recursos, a fim de permitir a construção de sistemas IoT de monitoramento remoto seguros. A avaliação foi realizada com experimentos práticos utilizando um transmissor e um receptor Arduino UNO R3, nos quais foram implementados os algoritmos AES e Speck, em suas versões originais e otimizadas para uso de memória, variando o tamanho da chave. Os resultados revelaram que em termos de consumo de energia, o algoritmo Speck tradicional com chave de 128 bits se destaca, sendo também o algoritmo com menor tempo de execução. Já o consumo de memória SRAM e *flash* foi menor para o Speck Tiny 128, seguido pelo Speck Small 128. As implementações otimizadas em memória tiveram um desempenho energético pior. Devido aos tempos de execução serem muito pequenos, houve pouca influência na vazão da transmissão, que alcançou no cenário avaliado 3,4 pacotes por segundo. Como trabalhos futuros, vislumbra-se a avaliação de outras técnicas criptográficas, incluindo funções de resumo criptográfico para garantia de integridade dos dados e outros algoritmos leves, como a família Ascon. Ademais, deve-se verificar o desempenho do *gateway* IoT ao receber dados de diversos sensores.

Referências

Abdullah, A. M. (2017). Advanced encryption standard (AES) algorithm to encrypt and decrypt data. *Cryptography and Network Security*, 16(1):11.

Albarello, R., Oyamada, M. e de Camargo, E. (2020). Avaliação de algoritmos de criptografia e implementação de um protocolo leve para comunicação entre dispositivos iot. Em *Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, p. 65–72, Porto Alegre, RS, Brasil. SBC.

Alvalá, R. C. S. e Barbieri, A. F. (2017). Desastres naturais. Em *Mudanças climáticas em rede: um olhar interdisciplinar*, volume 1, p. 203–230. Canal6Editora.

Beg, A., Al-Kharobi, T. e Al-Nasser, A. (2019). Performance evaluation and review of lightweight cryptography in an internet-of-things environment. Em *2nd International Conference on Computer Applications & Information Security (ICCAIS)*, p. 1–6.

Bogdanov, A., Khovratovich, D. e Rechberger, C. (2011). Biclique cryptanalysis of the full AES. Em Lee, D. H. e Wang, X., editors, *Advances in Cryptology – ASIACRYPT 2011*, p. 344–371, Berlin, Heidelberg. Springer Berlin Heidelberg.

de Paiva, B. D., de Souza, B. P. e Travassos, G. H. (2023). Cryptocomponent: um componente de criptografia para sistemas de software IoT de baixo custo. Em *Anais Estendidos do XIV Congresso Brasileiro de Software: Teoria e Prática*, p. 90–99. SBC.

El-hajj, M., Mousawi, H. e Fadlallah, A. (2023). Analysis of lightweight cryptographic algorithms on IoT hardware platform. *Future Internet*, 15(2).

Fotovvat, A., Rahman, G. M. E., Vedaei, S. S. e Wahid, K. A. (2021). Comparative performance analysis of lightweight cryptography algorithms for IoT sensor nodes. *IEEE Internet of Things Journal*, 8(10):8279–8290.

Guinelli, J. V., Aguiar, O. V. e Lazarin, N. M. (2018). Análise e comparação de algoritmos criptográficos simétricos embarcados na plataforma arduino. Em *Anais Estendidos do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, p. 167–176, Porto Alegre, RS, Brasil. SBC.

Hoffmann, R. B. e Griebler, D. (2023). Avaliando paralelismo em dispositivos com recursos limitados. Em *Anais da XXIII Escola Regional de Alto Desempenho da Região Sul*, p. 105–106. SBC.

Hossein Motlagh, N., Mohammadrezaei, M., Hunt, J. e Zakeri, B. (2020). Internet of things (iot) and the energy sector. *Energies*, 13(2).

Mouha, R. A. e Ait, R. (2021). Internet of things (IoT). *Journal of Data Analysis and Information Processing*, 09(2):77–101.

Panahi, P., Bayılmış, C., undefinedavuşoğlu, U. e Kaçar, S. (2021). Performance evaluation of lightweight encryption algorithms for IoT-based applications. *Arabian Journal for Science and Engineering*, 46(4):4015–4037.

Pereira, G. C. C. F., Alves, R. C. A., Silva, F. L. d., Azevedo, R. M., Albertini, B. C. e Margi, C. B. (2017). Performance evaluation of cryptographic algorithms over IoT platforms and operating systems. *Security and Communication Networks*, 2017:1–16.

Smith, R., Palin, D., Ioulianou, P. P., Vassilakis, V. G. e Shahandashti, S. F. (2020). Battery draining attacks against edge computing nodes in IoT networks. *Cyber-Physical Systems*, 6(2):96–116.

Song, L., Huang, Z. e Yang, Q. (2016). Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. Em Liu, J. K. e Steinfeld, R., editors, *Information Security and Privacy*, p. 379–394, Cham. Springer International Publishing.

Tawalbeh, L., Muheidat, F., Tawalbeh, M. e Quwaider, M. (2020). IoT privacy and security: Challenges and solutions. *Applied Sciences*, 10(12).

Vaz, Y., Mattos, J. e Soares, R. (2023). AES otimizado para uso em aplicações IoT. Em *Anais Estendidos do XIII Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, p. 31–36, Porto Alegre, RS, Brasil. SBC.

Zanon, V., Romancini, E. M., Manoel, B., Lau, J., Ourique, F. e Morales, A. (2022). Avaliação experimental de uma camada de segurança implementada em dispositivo vestível cardíaco para internet das coisas médicas. Em *Anais do XXII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, p. 97–110, Porto Alegre, RS, Brasil. SBC.