Análise da Aplicação de Técnicas de Segurança na Comunicação de Dispositivos Voltados ao Controle de Frotas

Vinícius Andriani Mazera
Universidade do Vale do Itajaí
Ravex Equipamentos Automotivos
Itajaí, SC, Brasil
vinicius.mazera@edu.univali.br

Rubens Vicente de Liz Bomer
Ravex Equipamentos Automotivos
Itapema, SC, Brasil
rubens.bomer@ravex.com.br

Fabricio Bortoluzzi
Universidade do Vale do Itajaí
Itajaí, SC, Brasil
fb@univali.br

Paulo Roberto Oliveira Valim
Universidade do Vale do Itajaí
Itajaí, SC, Brasil
pvalim@univali.br

Douglas Rossi de Melo
Universidade do Vale do Itajaí
Itajaí, SC, Brasil
drm@univali.br

ABSTRACT

The increasing reliance on embedded systems for logistics management and fleet control demands secure communication, given the inherent computational constraints of these devices. This work comparatively implemented and analyzed three lightweight cryptographic ciphers: Speck, ChaCha20, and ECC to provide confidentiality to a fleet monitoring firmware. The analysis compared critical metrics, including memory usage, latency, and battery consumption. Each cypher was contrasted to the baseline, clear-text firmware version. The ChaCha20 algorithm provided the best cost-benefit ratio, with minimal latency overhead. In contrast, the ECC imposed the highest cost, with latencies exceeding 1.7 seconds, and the highest RAM/ROM memory overhead. The results confirm the viability of lightweight cryptography, providing an objective basis for upcoming design decisions aimed at achieving balanced performance and security in embedded logistics systems.

KEYWORDS

Embedded Systems. Information Security. Cryptography. Fleet Monitoring

1 INTRODUÇÃO

Sistemas embarcados são dispositivos de computação integrados a produtos específicos, desenvolvidos para desempenhar funções predeterminadas com alta eficiência e confiabilidade. Esses sistemas geralmente operam em tempo real e com recursos limitados, como capacidade de processamento e de memória reduzidas [1]. Com o aumento do uso em setores essenciais, surgem desafios, especialmente no que se refere à segurança. Como esses dispositivos processam dados sensíveis e se conectam a redes, a implementação de medidas de proteção para garantir a integridade, a confidencialidade e a autenticidade das informações torna-se fundamental [2].

A segurança em sistemas embarcados é essencial devido ao uso desses dispositivos em aplicações sensíveis. Esses sistemas estão expostos a ameaças variadas, incluindo ataques físicos e remotos, como a interceptação de redes [3]. Devido às limitações de processamento e de energia, a implementação de mecanismos de defesa robustos é desafiadora. As práticas de segurança em sistemas embarcados incluem a criptografia para proteção de dados e o controle de acesso [4]. Neste cenário, surge a necessidade de avaliar como métodos de segurança leve se comportam em protocolos de comunicação industriais, como o RS-485, amplamente utilizado em frotas pela sua robustez física, mas carente de segurança nativa.

A segurança da informação é tradicionalmente estruturada em cinco pilares: confidencialidade, integridade, disponibilidade, autenticidade e não repúdio. Este trabalho concentra-se especificamente na proteção da confidencialidade de dados em trânsito, aplicando métodos para garantir que as informações transmitidas no barramento que integra os diversos dispositivos do sistema de frotas (RS-485) não possam ser interceptadas ou acessadas por terceiros não autorizados [5].

O trabalho propôs e executou o desenvolvimento e a análise de três cifras criptográficas leves: Speck, ChaCha20 e ECC. O objetivo central foi avaliar o *trade-off* entre segurança e desempenho, quantificando o sobrecusto real de latência, de memória (RAM e ROM) e de potência que cada algoritmo impõe ao ciclo de comunicação serial RS-485 em um dispositivo embarcado restrito.

Este artigo está organizado em seis seções. A Seção 2 trata da fundamentação teórica, na qual são apresentados os conceitos de monitoramento de frotas e de criptografia. A Seção 3 trata dos trabalhos relacionados. A Seção 4 descreve a implementação dos algoritmos e do protocolo RS-485. A Seção 5 apresenta e discute os resultados das análises de latência, de memória e de consumo energético. Por fim, na Seção 6, apresentam-se as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, são apresentados os conceitos relacionados ao monitoramento de frotas, aos sistemas embarcados e à criptografia e, posteriormente, são apresentados os trabalhos relacionados a esta pesquisa.

2.1 Monitoramento de Frotas

O monitoramento de frotas consiste na aplicação de tecnologias e sistemas que acompanham, em tempo real, a movimentação de veículos comerciais, otimizando rotas, reduzindo custos e aprimorando a segurança [6]. A coleta contínua de dados de veículos permite transformar dados brutos em informações estratégicas. O ambiente de telemetria em um veículo é complexo e integrado, conforme ilustrado na Figura 1, que apresenta diversos

Figura 1: Equipamento de rastreamento e telemetria embarcado em um caminhão [7]

tipos de equipamentos para o monitoramento de frotas [7]. No Brasil, o modal rodoviário é responsável por mais de 60% do deslocamento de produtos (Confederação Nacional do Transporte, 2024). Equipamentos de rastreamento e telemetria embarcados se comunicam com diversos periféricos, que também são sistemas embarcados, muitos deles por meio do barramento RS-485.

2.2 Criptografia

A criptografia é uma disciplina fundamental da segurança da informação, que consiste no uso de algoritmos e técnicas matemáticas para cifrar e decifrar dados. Seu objetivo principal é garantir a confidencialidade das informações em trânsito e em repouso, assegurando que apenas partes autorizadas possam acessá-las, além de assegurar integridade e autenticidade. Os métodos criptográficos são a base para a proteção de dados em qualquer ambiente de comunicação.

A criptografia é o mecanismo padrão *de facto* para o provimento de confidencialidade na transferência de dados. Pode ser simétrica ou assimétrica. A criptografia de Curva Elíptica (ECC) é uma forma de criptografia assimétrica que utiliza as propriedades matemáticas de curvas elípticas, oferecendo segurança comparável à de métodos tradicionais que utilizam chaves menores [8]. Em contraste, as cifras leves Speck e ChaCha20 são desenvolvidas para oferecer segurança em dispositivos com recursos computacionais muito limitados. No entanto, esses algoritmos, quando usados isoladamente, oferecem apenas confidencialidade e não garantem a autenticidade dos dados. O Speck prioriza eficiência em implementações de software (firmware) [9], enquanto o ChaCha20 opera como cifra de fluxo que equilibra boa robustez criptográfica e desempenho aceitável [10].

3 TRABALHOS RELACIONADOS

Sopran et al. [11] realizaram um estudo comparativo focado no custo e no desempenho de implementações criptográficas. O autor explorou o compartilhamento entre hardware e software em sistemas embarcados. Os resultados evidenciaram que o uso combinado de hardware e software pode representar um bom equilíbrio entre custo, consumo e desempenho em cenários com restrições de processamento e de memória.

Hamadaqa et al. [12] propuseram uma técnica resistente à clonagem para sistemas de gestão de frotas. O modelo utiliza autenticação baseada em hardware e a criação de identidades conjuntas para prevenir clonagens e acessos não autorizados. A implementação envolveu cifras secretas e únicas (Secret Unknown Ciphers - SUCs) para proteger a identidade do dispositivo. Os resultados demonstraram resiliência contra ataques de clonagem sob baixa sobrecarga computacional.

Li et al. [13] apresentaram uma metodologia de modelagem de segurança para o design de sistemas embarcados. Essa abordagem integra a Análise de Requisitos e Ameaças, o Particionamento hardware/software e o Design de Software. O objetivo é identificar vulnerabilidades críticas desde as fases iniciais e orientar a aplicação sistemática de contramedidas. A integração de modelagem de ameaças à análise de riscos promove um equilíbrio entre segurança e desempenho.

Dhanda et al. [14] realizaram uma revisão teórica sobre criptografia leve, analisando os principais algoritmos aplicáveis ao ambiente IoT. O trabalho categorizou as primitivas criptográficas em cifras de bloco, cifras de fluxo, funções hash leves e criptografia assimétrica baseada em curvas elípticas (ECC). O estudo comparou algoritmos como PRESENT, SIMON e ECC, detalhando suas características e desempenho para ilustrar como algoritmos otimizados podem equilibrar segurança e eficiência energética.

Enquanto trabalhos como o de Sopran et al. [11] focam na otimização de custos por meio do particionamento hardware/software, e Li et al. [13] na modelagem de design seguro, o presente estudo se diferencia por sua natureza. A contribuição central desse trabalho é a Análise Prática e Experimental das cifras leves (Speck, ChaCha20, ECC) aplicadas ao barramento RS-485, fornecendo métricas de Latência, RAM, ROM e Potência que validam a viabilidade dessas soluções em ambientes logísticos restritos.

4 DESENVOLVIMENTO

O desenvolvimento deste trabalho teve como objetivo implementar criptografia nas comunicações em barramento RS-485 em um firmware de monitoramento de frotas. O foco principal foi analisar a relação entre custo e desempenho da implementação de diferentes técnicas de criptografia leve, a fim de garantir a segurança da comunicação sem comprometer a operação do sistema.

A Figura 2 ilustra o fluxo de dados e os pontos de segurança. Os Dados Confidenciais são coletados e protegidos por algoritmos de criptografia para serem transmitidos com segurança aos periféricos ou ao Destino. O modelo considera a ameaça de um terceiro malicioso que possa interceptar a comunicação. O objetivo é garantir a integridade e a confidencialidade das informações trocadas localmente via RS-485.

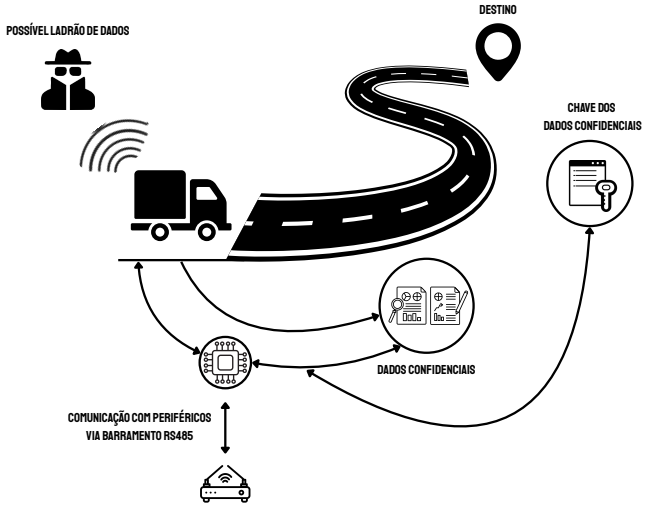

Figura 2: Visão geral do caminho dos dados

O microcontrolador utilizado foi o ESP32-wroom-32e, com o sensor INA226 e o Arduino Uno para a medição do consumo de energia. O firmware foi escrito em C/C++ utilizando o ESP-IDF. Para uma comparação justa, todos os algoritmos de criptografia (Speck, ChaCha20 e ECC) foram implementados sem recorrer a aceleradores de hardware do ESP32, com foco no custo intrínseco do software.

O algoritmo Speck foi implementado como cifra de bloco, adaptado para operar no modo ECB (Electronic Codebook). Esse modo foi escolhido para isolar e medir o custo computacional intrínseco das operações de round da cifra no firmware em C++, permitindo a medição da latência por bloco no microcontrolador, sem o sobrecusto de processamento de modos de encadeamento mais complexos. O ChaCha20 foi incluído como cifra de fluxo, dispensando o uso de mecanismos de padding e utilizando chaves de 256 bits e um nonce fixo. O módulo ECC utiliza a biblioteca MicroECC e o padrão ECIES (Elliptic Curve Integrated Encryption Scheme). Para garantir a segurança do esquema ECIES, o ECC foi configurado com a curva secp256r1 (P-256), utilizando a função nativa esp_random() do ESP32 para gerar números aleatórios para as chaves efêmeras. A derivação de chaves foi implementada com o algoritmo SHA-256, enquanto a cifra do payload utilizou o algoritmo AES-GCM com chaves de 256 bits.

Esta seção descreve os detalhes da implementação do módulo de criptografia e de sua integração ao protocolo de comunicação RS-485 no ambiente ESP-IDF/FreeRTOS. A arquitetura de software foi projetada com base nos princípios de modularidade para permitir a análise comparativa entre os algoritmos (Speck, ChaCha20 e ECC).

4.1 Fluxo do Protocolo

O fluxo de execução do firmware segue a Figura 3, que descreve a coleta de dados, a criptografia, a transmissão e o tratamento de falhas. O ilustra este ciclo de operação. O processo inicia-se com a Inicialização do sistema e avança para a fase de Coleta de Dados dos periféricos. Após a coleta, os dados passam pela fase de Criptografia, onde o algoritmo de segurança selecionado é aplicado ao payload. O sistema então tenta Transmitir Dados via barramento RS-485 e entra no estado de Aguardando Confirmação (ACK).

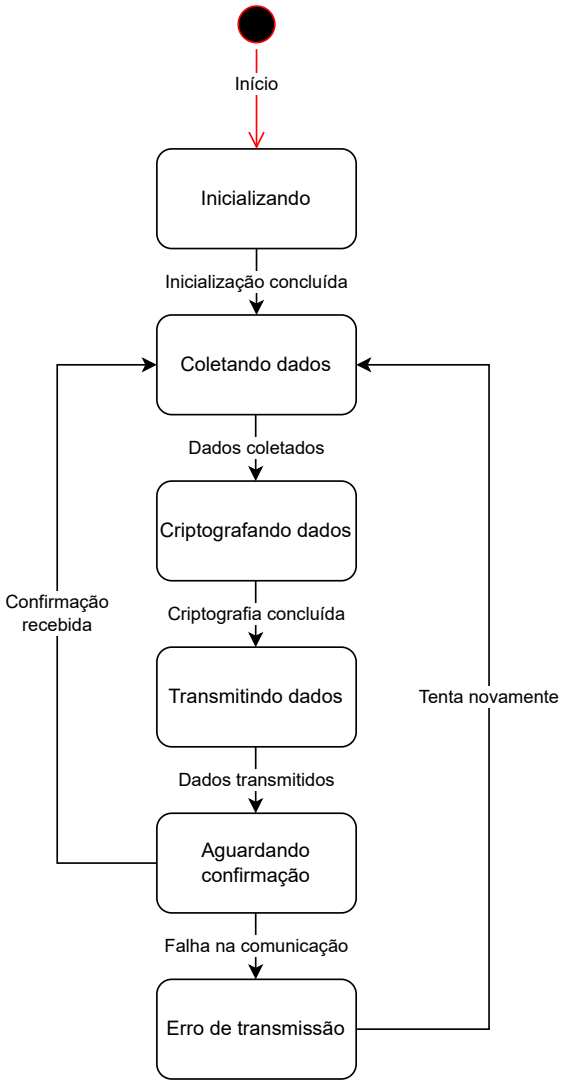

Figura 3: Diagrama de estados

A implementação do sistema de criptografia utiliza o Factory Pattern para desacoplar a lógica de rede da lógica algorítmica. Para este fim, foi definida uma interface básica que exige a implementação dos métodos polimórficos crypto e decrypto. A seleção do algoritmo específico é controlada por definições de pré-processamento, o que facilita a troca e os testes rápidos entre as diferentes opções de segurança.

4.2 Ciclo de comunicação

O ciclo de comunicação completo entre os módulos Mestre e Escravo, incluindo a criptografia dos dados e do Acknowledgement (ACK), é detalhado na Figura 4 . O Mestre inicia o ciclo criando o pacote de dados e o submetendo à etapa de Criptografia do payload do pacote, antes de enviá-lo pelo Barramento RS-485. Ao

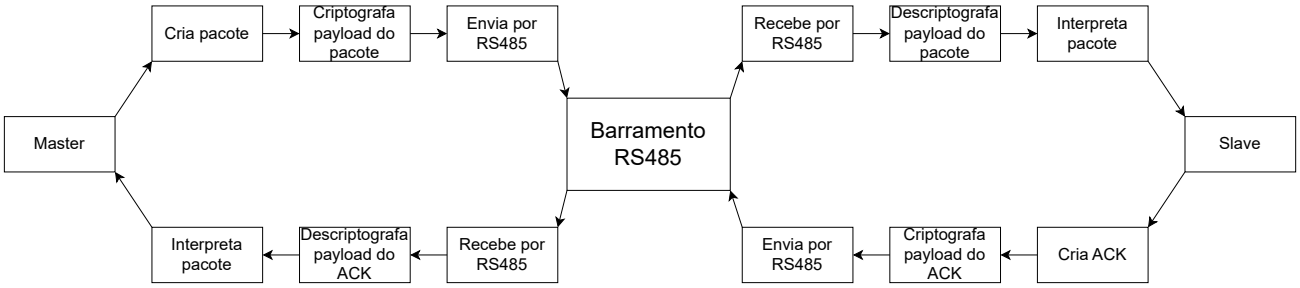

Figura 4: Ciclo de comunicação completo

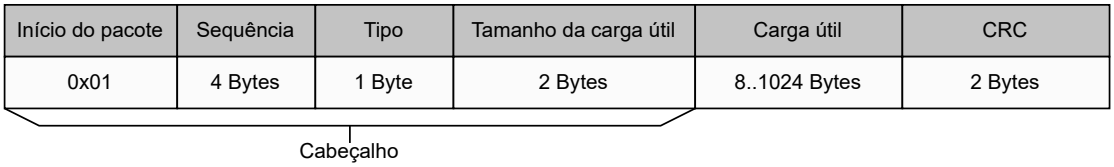

Figura 5: Estrutura do pacote

receber o pacote, o Escravo processa a Descriptografia do payload para Interpretar o pacote. Caso a interpretação seja bem-sucedida, o Escravo cria um ACK, que também é criptografado antes de ser enviado de volta pelo RS-485. Finalmente, o Mestre recebe, descriptografa o ACK e interpreta o pacote, concluindo um ciclo completo de transmissão segura.

4.3 Estrutura do Pacote

A comunicação RS-485 foi implementada no ambiente FreeRTOS. O pacote de comunicação para o processo de criptografia e medição de desempenho é demonstrado na Figura 5, ele é definido por três componentes principais: um Cabeçalho (que contém o número do pacote e o tamanho do payload), seguido pelo payload propriamente dito, e finalizado por um CRC (Cyclic Redundancy Check) para a garantia de integridade da transmissão . O protocolo foi configurado para operar a uma taxa de 115200 baud e utiliza o modo half-duplex. Para a confirmação de recebimento, utiliza-se o mecanismo de Acknowledgement (ACK).

4.4 Implementação dos Algoritmos de Criptografia

Todos os algoritmos de criptografia utilizados neste trabalho foram obtidos por meio de bibliotecas externas, garantindo que as implementações sigam padrões amplamente aceitos e sejam otimizadas para sistemas embarcados. A seguir, são detalhadas as especificidades de cada algoritmo:

- Speck: Foi implementado como uma Cifra de Bloco, adaptado para operar no modo ECB (*Electronic Codebook*) para criptografar *payloads* com mais de 16 bytes. O Speck é otimizado para eficiência em implementações de *software* (firmware).
- ChaCha20: Foi incluído como cifra de fluxo. Sua arquitetura é inerentemente sequencial e dispensa mecanismos de preenchimento (*padding*), permitindo criptografar ou descriptografar o *payload* em uma única chamada.
- ECC (Elliptic Curve Cryptography): O módulo utiliza a biblioteca MicroECC para o *setup* da chave e o padrão ECIES (*Elliptic Curve Integrated Encryption Scheme*) para a segurança do *payload*. O sistema executa o cálculo ECDH (*Elliptic Curve Diffie-Hellman*) em cada ciclo para derivar o segredo compartilhado, que é então utilizado como a chave simétrica para a criptografia do *payload*. O custo de latência é obtido por meio da chamada à função AESGCM.

5 RESULTADOS

Nesta seção, são apresentados e discutidos os resultados de síntese e simulação, com foco na quantificação do sobrecusto imposto pelos algoritmos de criptografia em relação ao cenário base (sem segurança).

Os experimentos foram conduzidos com diversos payloads, utilizando dois módulos ESP32 em comunicação serial, operando em modo half-duplex via barramento RS-485, no modelo Mestre-Escravo. As medições de latência e de consumo de recursos foram realizadas coletando dados de tempo de execução e de uso de memória (RAM e ROM) diretamente do microcontrolador. Além disso, o consumo energético foi monitorado e analisado com o sensor INA226 e o Arduino Uno, fornecendo uma visão completa do impacto da segurança no desempenho do hardware restrito.

O ambiente experimental utilizado para a coleta de dados é ilustrado na Figura 6, na qual são apresentados os materiais empregados no desenvolvimento do trabalho. A imagem apresenta as placas ESP32, nas quais o firmware foi gravado, as placas INA226 e Arduino UNO para medição e leitura da energia, e um sniffer para ler os dados transmitidos em tempo real.

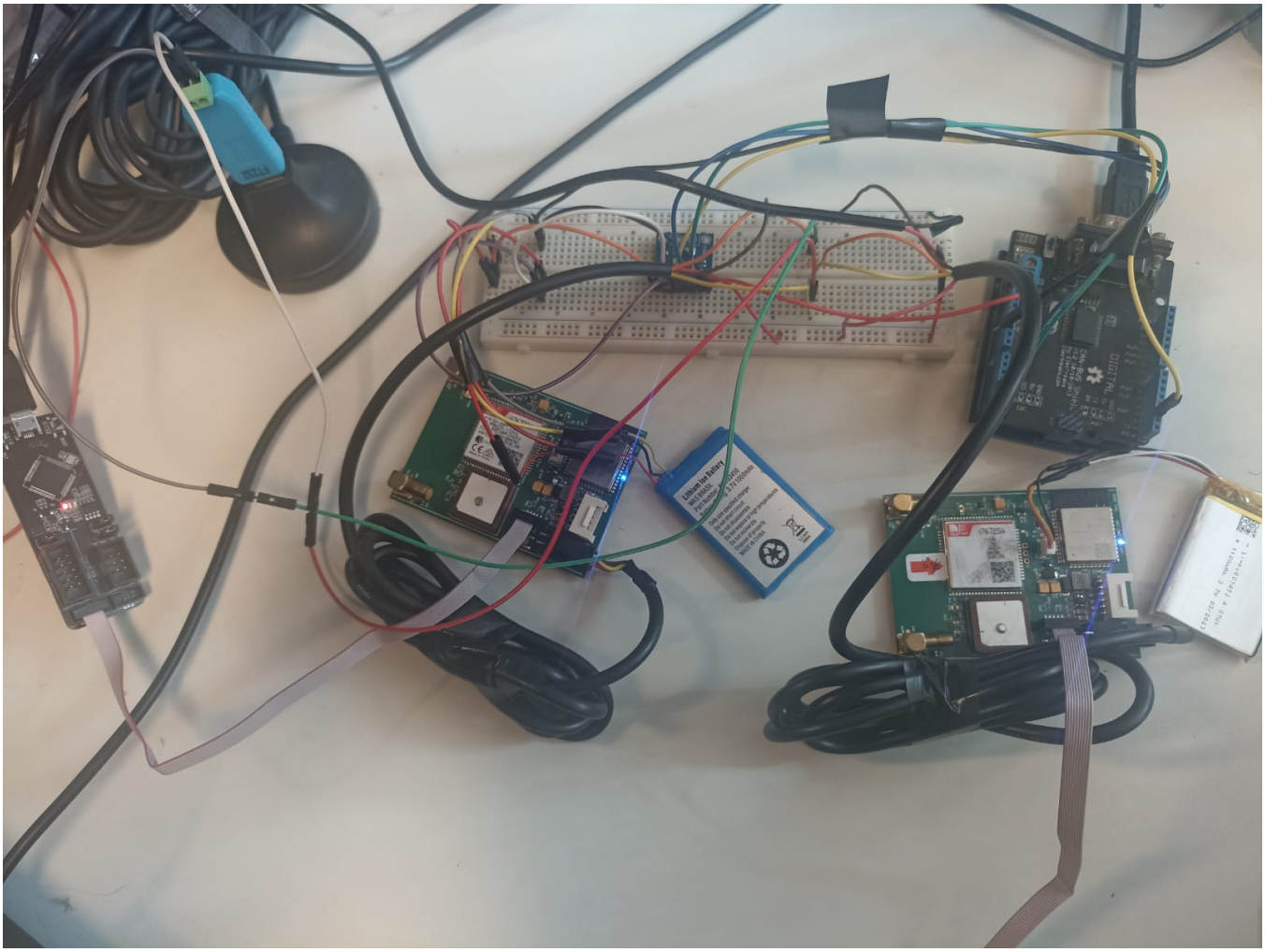

Figura 6: Bancada dos experimentos

5.1 Latência

A Tabela 1 detalha a latência média dos algoritmos em função do tamanho do payload. O algoritmo ChaCha20 se destacou por sua eficiência, registrando uma sobrecarga máxima marginal de apenas 0,95% para o payload máximo. Em contraste, o ECC impôs o maior sobrecusto de latência, iniciando em cerca de 1,02 segundos e escalando até 1,73 segundos. Este custo é impulsionado pelo algoritmo assimétrico.

A metodologia de coleta de dados seguiu um protocolo rigoroso de 100 repetições para cada tamanho de payload e algoritmo. Antes de cada bateria de testes, o dispositivo foi reiniciado para garantir que a memória RAM (heap e stack) estivesse em estado limpo, eliminando interferências decorrentes da fragmentação de memória. A coluna 'Base' na Tabela 1 representa o baseline do sistema: o tempo de execução do firmware realizando apenas a comunicação RS-485 via FreeRTOS, sem qualquer carga de processamento criptográfico, servindo como referência para o cálculo do sobrecusto.

Tabela 1: Latência Média (μs) dos Algoritmos de Criptografia

Payload (Bytes)	Base (μs)	Speck (μs)	ChaCha20 (μs)	ECC (μs)
8	46753	46783	45607	1018180
16	53591	53992	52651	1024860
32	63121	65742	62881	1034334
64	86152	94252	85138	1059500
128	128203	143625	127799	1102155
256	216483	246688	218972	1191910
512	392087	449120	395345	1370710
1024	744046	859568	751081	1729272

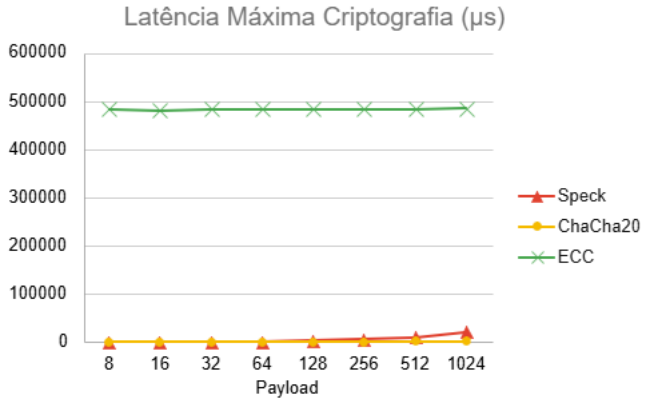

Figura 7: Latência máxima apenas da criptografia

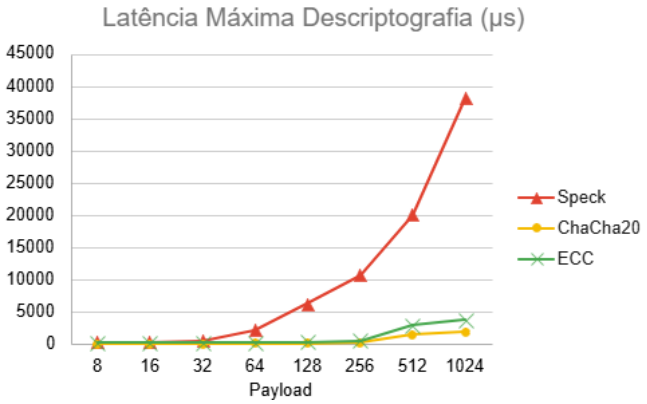

Figura 8: Latência máxima apenas da descriptografia

A análise da Latência Máxima de Criptografia e Descriptografia, apresentada nas Figuras 7 e 8, revela uma disparidade no custo de execução do algoritmo ECC. A Latência Máxima de Criptografia do ECC permanece em um patamar elevado, próximo a 485.000 µs, independentemente do tamanho do payload. Este custo fixo é imposto pelo cálculo inicial e complexo do ECDH (Elliptic Curve Diffie-Hellman), necessário para a geração da chave simétrica de sessão. Por outro lado, a Latência Máxima de Descriptografia do ECC apresenta um custo inicial drasticamente menor, próximo ao das cifras simétricas, com crescimento gradual até atingir aproximadamente 4.000 µs para um payload de 1024 bytes.

A diferença operacional decorre do modelo de Encapsulamento de Chave (KEM), em que a operação mais lenta e intensiva (ECDH) é executada predominantemente na fase de criptografia (onde o segredo é gerado e encapsulado), enquanto a fase de descriptografia se concentra apenas na rápida descriptografia do payload, com a chave já estabelecida.

5.2 Consumo de memória

Os custos de memória são apresentados nas Figuras 9 e 10. O ECC registrou o maior sobrecusto de memória volátil (RAM, 1,46%) e não volátil (ROM, 4,31%), atribuído à complexidade do algoritmo e à necessidade de alocar estruturas de dados para a aritmética de alta precisão. Em contrapartida, as cifras simétricas (ChaCha20 e Speck) apresentaram sobrecarga de memória desprezível (abaixo de 1%)

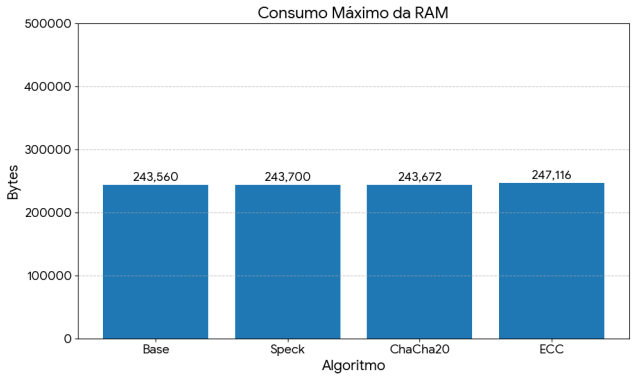

Figura 9: Consumo de memória RAM

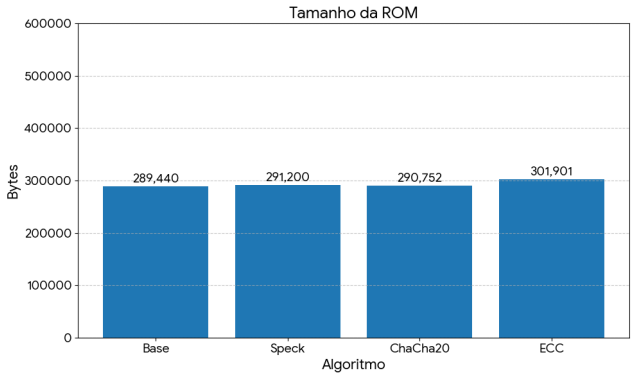

Figura 10: Tamanho da memória ROM

5.3 Consumo Energético

A análise do consumo médio de potência é apresentada na Figura 11. Embora os valores absolutos de potência (mW) sejam próximos devido à alta demanda base do ESP32, o algoritmo ECC impôs consistentemente o maior sobrecusto percentual em todos os payloads, com um pico de 18,92% em relação à Base.

5.4 Discussão

A Figura 12 apresenta a média das latências por algoritmo, que se mostrou o valor com maior discrepância entre as métricas coletadas, servindo como evidência central da carga de trabalho imposta ao sistema. Este gráfico consolida o trade-off entre a latência

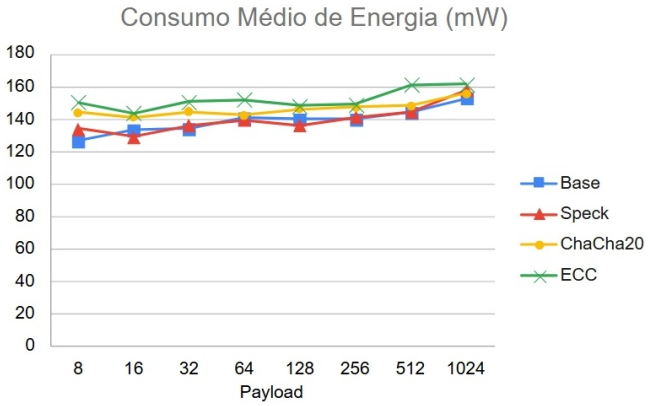

Figura 11: Consumo médio de potência (mW) por tamanho de payload

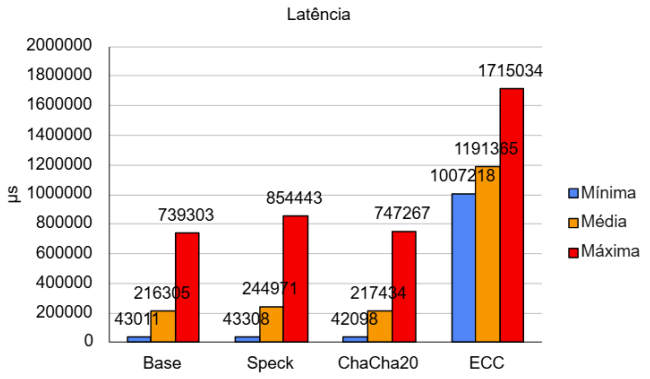

Figura 12: Latência média geral de todos os testes

e a complexidade do setup assimétrico e a eficiência das cifras simétricas ao longo de um ciclo completo de comunicação.

O algoritmo ECC se destaca como o fator de maior custo em todos os vetores de análise de recursos. Sua implementação demandou a inclusão de bibliotecas para aritmética de alta precisão, resultando em uma sobrecarga de memória ROM de 4,31% e o maior sobrecusto de RAM (1,46%). Este custo de setup alto e fixo é inerente à natureza da criptografia de chave pública, na qual a complexidade matemática exige mais código e mais memória volátil. A latência máxima na comunicação ratifica essa sobrecarga, com mais de 1,7 segundos no custo total do ciclo.

Em contraste, as cifras simétricas demonstram uma eficiência notável em recursos de memória. O Speck e o ChaCha20 confirmam ser altamente adequados para ambientes restritos, apresentando uma sobrecarga de ROM e RAM inferior a 0,1%. A sobrecarga de latência do ECC é o custo mais proibitivo para a aplicação em tempo real. Embora o custo de memória das cifras simétricas seja marginal, a análise de latência e de consumo de energia ratifica o ChaCha20 como a escolha de maior custo-benefício para a criptografia do payload, proporcionando desempenho sem comprometer significativamente a autonomia do dispositivo.

A análise do consumo energético baseou-se na medição da potência (mW) em 100 ciclos de comunicação para cada algoritmo, utilizando o sensor INA226 para medir a média do Bus Power. O impacto percentual do sobrecusto dos algoritmos simétricos no consumo médio de potência foi moderado, o que confirma que a carga base do sistema domina o consumo de energia. Embora o Consumo Médio de Potência (mW) das cifras simétricas e do ECC apresente uma diferença absoluta leve, a Latência é o fator dominante.

Embora o foco deste trabalho tenha sido a garantia de confidencialidade, é importante considerar os pilares de integridade e autenticidade no contexto das cifras analisadas. As implementações puras de Speck e ChaCha20 utilizadas neste experimento não garantem a autenticidade dos dados, tornando o sistema vulnerável a ataques de modificação de payload em trânsito. Para reduzir esse risco em aplicações reais, seria necessária a integração de códigos de autenticação de mensagens. Em contrapartida, o esquema ECIES utilizado no módulo ECC já provê nativamente esses pilares por meio do uso do AES-GCM. Este algoritmo de criptografia autenticada gera uma tag de autenticação que assegura tanto a integridade quanto a origem da mensagem, confirmando que o ECC, embora apresente maior custo computacional, oferece a camada de segurança mais robusta entre as técnicas avaliadas.

6 CONCLUSÃO

O aumento da dependência de sistemas embarcados para o monitoramento e o controle de frotas impõe a necessidade de soluções que garantam não apenas a eficiência operacional, mas também a segurança na comunicação de dados. Nesse contexto, este trabalho propôs e executou a implementação e a avaliação de camadas de segurança baseadas em criptografia leve em um firmware de rastreamento de frotas utilizando a plataforma ESP32.

O firmware foi estruturado de forma modular para coletar métricas de latência, uso de memória (RAM e ROM) e potência dissipada (mW), fornecendo dados objetivos para a análise comparativa. A análise dos resultados finais confirmou que o ChaCha20 é a solução de melhor desempenho para o cenário apresentado, com uma sobrecarga marginal em relação à Base em termos de latência e memória. O ECC demonstrou ser o fator mais limitante do sistema, com latências superiores a 1,7 segundos e o maior sobrecusto de RAM e de ROM.

Como trabalhos futuros, sugere-se ampliar a análise comparativa para mais algoritmos e para diferentes comunicações do RS-485, como CAN Bus ou redes sem fio. Adicionalmente, sugere-se avaliar em detalhes as métricas de latência e de consumo energético da etapa de troca de chaves (key exchange phase) na criptografia assimétrica, como no ECC.

AGRADECIMENTOS

Este trabalho foi financiado, em parte, pela Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina – FAPESC, Chamada 51/2024 (Contrato 2024TR001897), pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, e pela Ravex Equipamentos Automotivos Ltda.

REFERÊNCIAS

[1] Marilyn Wolf. *Computers as Components: Principles of Embedded Computing System Design, Second edition.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[2] Peter Marwedel. *Embedded System Design.* Springer, Dordrecht, Netherlands, 2011.

[3] Georgios Michail Makrakis, Constantinos Kolias, Georgios Kambourakis, Craig Rieger, and Jacob Benjamin. Industrial and critical infrastructure security: Technical analysis of real-life security incidents. *IEEE*, 2021.

[4] William Stallings and Lawrie Brown. *Computer Security: Principles and Practice.* Pearson, Upper Saddle River, NJ, USA, 2008.

[5] Kelce S. Wilson. Conflicts among the pillars of information assurance. *IEEE IT Professional*, 15(4):44–49, 2013.

[6] TT News. Fleet monitoring: Boosting efficiency, 2025. Accessed: 26 Mar. 2025.

[7] Ravex. Ravex: Soluções completas para logística, risco e agro. https://ravex.com.br/, 2025. Acessado em: 25 mar. 2025.

[8] Chao Luo, Yunsi Fei, and David Kaeli. Effective simple-power analysis attacks of elliptic curve cryptography on embedded systems. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018.

[9] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck families of lightweight block ciphers. *Cryptology ePrint Archive*, 2013(404), 2013.

[10] Jean-Philippe Aumasson. *Serious Cryptography.* No Starch Press, San Francisco, 2024.

[11] Robson Sopran, Douglas R Melo, Cesar A Zeferino, and Eduardo A Bezerra. Análise comparativa do custo e do desempenho de um algoritmo de criptografia explorando o particionamento hardware/software, 2016. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) Universidade do Vale do Itajaí, Itajaí, 2016.

[12] Emad Hamadaqa, Saleh Mulhem, Ayoub Mars, and Wael Adi. Clone-resistant joint-identity technique for securing fleet management systems. In *Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, USA, 2018. IEEE.

[13] Letitia W. Li, Florian Lugou, and Ludovic Apvrille. Security modeling for embedded system design. In *GraMSec 2017, LNCS 10744*, Sophia Antipolis, France, 2018. Springer International Publishing AG.

[14] Sumit Singh Dhanda, Brahmjit Singh, and Poonam Jindal. Lightweight cryptography: A solution to secure iot. *Wireless Personal Communications*, 111, 2020.