Avaliação de Desempenho de Algoritmos de Cifra de Fluxo para Sistemas Embarcados

Giovanni Schrickte Sartori[1], **Ana Cristina B. Kochem Vendramin**[1], **Juliana de Santi**[1], **Daniel Fernando Pigatto**[1]

[1]Departamento Acadêmico de Informática (DAINF)
Universidade Tecnológica Federal do Paraná (UTFPR). Curitiba – PR – Brazil

giovannisartori@alunos.utfpr.edu.br

{criskochem,jsanti,pigatto}@utfpr.edu.br

Abstract. *Embedded systems connected to industrial networks require cryptographic mechanisms compatible with their limited resources. This work experimentally compares the reference implementations of four standardized lightweight stream ciphers, ChaCha20, Trivium, Rabbit, and Enocoro-128v2, on a Raspberry Pi 4, measuring encryption time, throughput, cycles per byte, and memory footprint. Trivium reached 598 MiB/s, whereas ChaCha20, Rabbit, and Enocoro-128v2 reached 173 MiB/s, 232 MiB/s, and 57 MiB/s. The differences are explained by the operation count per keystream byte and the instruction-level parallelism each round function exposes to the Cortex-A72 pipeline.*

Resumo. *Sistemas embarcados conectados a redes industriais demandam mecanismos de criptografia compatíveis com seus recursos limitados. Este trabalho compara experimentalmente as implementações de referência de quatro cifras de fluxo leves padronizadas, ChaCha20, Trivium, Rabbit e Enocoro-128v2, em uma Raspberry Pi 4, medindo tempo de cifragem, vazão, ciclos por byte e footprint de memória. O Trivium atingiu 598 MiB/s, enquanto ChaCha20, Rabbit e Enocoro-128v2 obtiveram 173 MiB/s, 232 MiB/s e 57 MiB/s. As diferenças são explicadas pelo número de operações por byte de keystream e pelo paralelismo em nível de instrução que cada função de rodada expõe ao pipeline do Cortex-A72.*

1. Introdução

Sistemas embarcados são implementados como uma combinação de *software* e *hardware* que executa uma função específica, geralmente como parte de um sistema maior. Grande parte dos sistemas embarcados modernos atua direta ou indiretamente na transmissão de dados e, ao serem conectados à internet, podem se tornar alvo de ataques que visam à manipulação ou aquisição de informações sensíveis [Silva et al. 2016].

Algoritmos de criptografia são uma das maneiras mais eficazes de proteger dados transmitidos, mas a escassez de recursos em sistemas embarcados pode inviabilizar implementações tradicionais. A criptografia leve, desenvolvida para fornecer segurança adequada com baixo custo computacional e baixo consumo de recursos, deve ser analisada e comparada de acordo com requisitos e restrições da aplicação-alvo [Bühler et al. 2022].

Diante desse cenário, a seleção apropriada de uma cifra de fluxo para cenários de Internet das Coisas Industrial (IIoT, do inglês *Industrial Internet of Things*) requer

uma compreensão abrangente não apenas de suas barreiras matemáticas, mas de seu comportamento prático em *hardware* comercial. Embora a literatura apresente extensas análises teóricas, observa-se uma lacuna na avaliação experimental e comparativa de implementações de referência fornecidas diretamente por comitês de padronização em plataformas de computação de borda acessíveis. Este trabalho tem como objetivo apresentar uma avaliação sistemática de desempenho das cifras de fluxo leves ChaCha20, Trivium, Rabbit e Enocoro-128v2 executadas em uma plataforma Raspberry Pi 4. Pretende-se, através do levantamento de métricas de tempo de cifragem, vazão (*throughput*), ciclos por byte e *footprint* de memória, fornecer subsídios empíricos para o projeto de sistemas embarcados seguros e de alta eficiência.

O restante do artigo está assim organizado: a Seção 2 apresenta a fundamentação teórica sobre criptografia simétrica e as características das cifras de fluxo selecionadas; a Seção 3 descreve a metodologia experimental, incluindo o ambiente de testes, as ferramentas de medição e os parâmetros estatísticos utilizados; a Seção 4 apresenta e analisa os resultados obtidos; e a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Fundamentação teórica

A criptografia de dados é o principal mecanismo de proteção em redes de comunicação. Algoritmos criptográficos dividem-se em chave assimétrica (par pública/privada) e simétrica (chave única compartilhada). Os algoritmos simétricos, por sua vez, podem ser classificados em cifras de bloco e cifras de fluxo. As cifras de bloco aplicam uma transformação fixa a blocos do texto original, enquanto as cifras de fluxo realizam transformações variáveis sobre dígitos ou bits individuais [Robshaw 1995]. As cifras de fluxo são amplamente empregadas em sistemas embarcados, especialmente em cenários nos quais o tamanho do texto original é desconhecido ou em que a cifragem precisa ocorrer de forma contínua [Manifavas et al. 2016].

Foram selecionadas quatro cifras de fluxo amplamente referenciadas em criptografia leve para IIoT [Khan et al. 2025]: ChaCha20 (RFC 7539), Trivium e Rabbit (finalistas do portfólio eSTREAM[1]) e Enocoro-128v2 (ISO/IEC 29192-3). O critério de escolha foi a disponibilidade de implementação de referência publicada pelo desenvolvedor ou pelo comitê de padronização, garantindo reprodutibilidade e comparabilidade metodológica.

ChaCha20, projetado por Bernstein [Bernstein et al. 2008] e padronizado pela RFC 7539 [Nir and Langley 2018], é uma evolução da cifra de fluxo Salsa20, selecionada no projeto eSTREAM. O algoritmo utiliza uma chave de 256 bits e um vetor de inicialização (IV) de 96 bits. Seu funcionamento baseia-se em um estado interno de 512 bits, composto por dígitos de chave e constantes, organizados em uma matriz 4 x 4 de palavras de 32 bits. As transformações do estado são realizadas por meio da função *Quarter Round*, responsável por combinar operações de soma modular, XOR e rotações binárias.

Trivium é uma cifra de fluxo orientada a *hardware* com *design* focado em flexibilidade, padronizada pela ISO/IEC 29192-3:2012 e selecionada para o portfólio eSTREAM devido ao seu baixo custo [De Cannière and Preneel 2006]. Ela funciona em duas etapas, com uma chave K e um vetor IV, ambos de 80 bits, os quais são carregados em um estado interno s de 288 bits. Após a inserção da chave e do IV, os demais registradores do estado

[1] `https://competitions.cr.yp.to/estream.html`

são preenchidos com bits zero. Este é, então, rotacionado quatro vezes durante a fase de *warmup* e, em seguida, continua sendo rotacionado até que um fluxo de *keystream* de $N \leq 2^{64}$ bits z_i seja gerado.

Enocoro-128v2 é uma cifra de fluxo orientada a *hardware* proposta por Watanabe et al. [Watanabe et al. 2010], padronizada pela ISO/IEC 29192-3:2012 e recomendada pelo CRYPTREC[2]. Inspirada na estrutura PANAMA-like, recebe uma chave de 128 bits e um IV de 64 bits, produzindo 8 bits de *keystream* por rodada. Seu estado interno de 280 bits é dividido em um *buffer* b de 256 bits $(b_0, \ldots, b_{31})$, um estado a de 16 bits e um contador ctr de 8 bits. A atualização do estado decompõe-se em uma função ρ (substituição não-linear via S-box) sobre a e uma função λ (permutação linear via XORs) sobre b. A inicialização executa 96 rodadas sem produzir saída.

Rabbit é uma cifra de fluxo orientada a *software* proposta por Boesgaard et al [Boesgaard et al. 2008]. A cifra foi selecionada para o portfólio final eSTREAM (*Profile* 1) e posteriormente padronizada pela RFC 4503. A cifra recebe chave de 128 bits e IV de 64 bits, produzindo blocos de 128 bits de *keystream* por iteração. Seu estado interno de 513 bits é dividido em oito variáveis de estado $x_{j,i}$ de 32 bits, oito contadores $c_{j,i}$ de 32 bits e um bit de *carry*. A função de próximo-estado é construída a partir de oito instâncias acopladas da função não-linear $g(z) = LSB_{32}(z^2 \oplus (z^2 \gg 32))$, garantindo alta difusão. Além disso, os contadores asseguram um período mínimo de 2^{215}. Tanto a inicialização da chave quanto a do IV envolvem quatro iterações da função de próximo estado.

3. Metodologia

Os experimentos foram executados em uma Raspberry Pi 4 Model B (SoC Broadcom BCM2711), com processador ARM Cortex-A72 (ARMv8-A) de 1,5 GHz: núcleo superescalar de execução fora de ordem, três instruções por ciclo, 31 registradores de 64 bits e 32 KiB de cache L1 de dados por núcleo. Os algoritmos foram executas em sistema operacional Linux (Debian).

O volume processado foi padronizado em 1 MiB, em pacotes de 1 KiB, múltiplo exato dos blocos matemáticos de todas as cifras e residente por inteiro na cache L1, de modo que a hierarquia de memória não constitua gargalo e os tempos reflitam a capacidade das unidades de execução. O procedimento gera 1 KiB de dados aleatórios (`/dev/urandom`) e os processa em *loop* de 1024 iterações, sem acesso ao disco, garantindo que o tempo medido corresponda apenas ao processamento [Vaz et al. 2025]. O tempo de CPU foi medido pelo próprio programa, evitando flutuações de processos externos, e um *script* em Python executou $n = 50$ repetições independentes por cifra. A compilação foi realizada com o GCC 14.2.0 utilizando a *flag* -O3. O *footprint* foi avaliado pelo *sizeof* da estrutura de contexto e pelo estado mínimo especificado.

4. Resultados

4.1. Análise estatística e métricas derivadas

A Tabela 1 consolida os resultados. Com $n = 50$ execuções independentes, tem-se $df = n - 1 = 49$ graus de liberdade. Como o desvio padrão populacional é desconhecido e

[2] `https://www.nict.go.jp/publication/shuppan/kihou-journal/journal-vol63no2/journal-vol63no2-07-03.pdf`

estimado da própria amostra, adotou-se a distribuição t de Student em lugar da normal: para $df = 49$ e confiança bilateral de 95 %, $t_{crít} = 2{,}010$ e a semiamplitude é $IC = t_{crít} \cdot \sigma/\sqrt{n}$, escolha conservadora, pois o valor crítico normal $z = 1{,}960$ estreitaria o intervalo em 2,6 %. A maior semiamplitude obtida é 0,0175 ms, ou 0,44 % da média no pior caso, ao passo que o menor espaçamento entre médias adjacentes, entre Rabbit e ChaCha20, é 1,4617 ms, cerca de 56 vezes a soma das semiamplitudes envolvidas. Como os intervalos de confiança de 95 % não se sobrepõem, todas as diferenças entre as médias são significativas, dispensando testes adicionais.

Tabela 1. Cifragem de 1 MiB em *chunks* de 1 KiB ($n = 50$; IC de 95 % pela distribuição t de Student, $df = 49$, $t_{crít} = 2{,}010$). W e IPC (instruções por ciclo) são estimativas analíticas definidas na Seção 4.2 e devem ser lidos como ordens de grandeza.

Cifra	Tempo (ms)	σ (ms)	IC 95 % ($\pm$ ms)	CV (%)	Vazão (MiB/s)	cpb	ns/B	W	IPC
Trivium	**1,6701**	0,0260	0,0074	1,56	**598,77**	2,39	1,59	$\approx$6	$\approx$2,5
Rabbit	4,3001	0,0418	0,0119	0,97	232,55	6,15	4,10	$\approx$6	$\approx$1,0
ChaCha20	5,7618	0,0496	0,0141	0,86	173,56	8,24	5,49	$\approx$15	$\approx$1,8
Enocoro	**17,5180**	0,0614	0,0175	0,35	**57,08**	25,06	16,71	$\approx$16	$\approx$0,6

4.2. Fatores arquiteturais determinantes

A diferença de desempenho entre os algoritmos, com o Trivium alcançando uma vazão mais de dez vezes superior à do Enocoro, decorre da relação entre o trabalho elementar por byte de *keystream* (W) e o aproveitamento do *pipeline* superescalar do processador Cortex-A72, estimado por meio do IPC (instruções por ciclo). O Trivium, embora projetado para *hardware*, possui uma estrutura que permite agrupar até 64 rodadas independentes em palavras de 64 bits. Como exige apenas operações lógicas simples de latência unitária, a cifra maximiza o uso dos registradores e vias de despacho do ARMv8-A, mantendo seu estado totalmente na CPU e resultando em um IPC estimado de 2,5.

O ChaCha20 apresenta posição intermediária devido à margem conservadora de 20 rodadas, o que impõe uma maior carga de operações elementares por byte. No entanto, o alto grau de paralelismo interno de suas *Quarter Rounds* permite sustentar um IPC de 1,8, auxiliado pelo uso de instruções nativas de rotação do processador. O Rabbit, por sua vez, possui carga de trabalho semelhante à do Trivium, mas é duramente limitado por multiplicações de 32 bits e pela atualização em cadeia dos contadores de *carry*. Essa dependência estrita entre as iterações impede a CPU de ocultar latências por meio da execução fora de ordem, reduzindo o IPC para 1,0.

No extremo oposto, o Enocoro-128v2 é duplamente penalizado. Primeiro, gera apenas 8 bits por rodada, subutilizando severamente o caminho de dados de 64 bits. Segundo, os acessos sucessivos à sua *S-box* demandam consultas à memória (cache L1) que criam um caminho crítico que limita a exploração do paralelismo pelo *pipeline* e reduz o IPC a 0,6. Em *software*, essas escolhas de *design* representam lentidão, embora em *hardware* garantam a desejada redução de área física. O *footprint* reflete essas assimetrias: cifras de *hardware* exigem mais espaço transiente em *software* (como a representação paralela no Trivium e as tabelas no Enocoro) do que as cifras desenvolvidas nativamente para processamento de uso geral.

4.3. Ameaças à validade

Validade interna. O ruído inerente ao sistema operacional multitarefa foi mitigado pelo alto número de repetições e pelo rigor das medições, resultando em um coeficiente de variação (CV) $< 2\,\%$ em todos os casos. Contudo, permanecem limitações inerentes às condições de ensaio, como a estabilidade da frequência da CPU (exigindo ausência de *throttling*), a não fixação do processo a um núcleo isolado (sujeito a perdas de cache) e a sensibilidade do Trivium ao desempenho de otimizadores específicos do compilador, como o desenrolamento de laços e a alocação de registradores.

Validade externa. Os resultados obtidos refletem o comportamento de uma arquitetura superescalar de 64 bits com execução fora de ordem. Em microcontroladores *in-order* mais restritos, típicos de nós terminais da IIoT, as latências não podem ser mascaradas, o que prejudicaria o Rabbit e forçaria transferências lentas para a memória devido à escassez de registradores. Ademais, o teste em fluxo contínuo favorece cifras com alto custo de inicialização (o Trivium exige 1152 iterações de aquecimento). Em aplicações reais com pacotes curtos e rechaveamento frequente, essa vantagem de vazão tende a ser reduzida em relação ao ChaCha20, o qual se beneficiaria ainda mais de uma implementação vetorizada (NEON) não contemplada neste estudo.

Validade de construto. A métrica de tamanho baseada em *sizeof* não contabiliza a pilha e as constantes, podendo subestimar o tamanho real do Enocoro. No aspecto da segurança, a comparação envolve cifras em patamares desiguais: a liderança do Trivium apoia-se em uma chave de apenas 80 bits (abaixo dos padrões atuais do NIST) e limite de volume no *keystream*, contra 256 bits do ChaCha20. Em cifras de 128 bits, o Rabbit despontaria como o mais veloz. Por fim, a alta vazão não garante resiliência contra canais laterais; enquanto ChaCha20 e Trivium são nativamente de tempo constante, a estrutura em tabela do Enocoro o expõe a vulnerabilidades de análise de tempo de cache.

5. Conclusão e Trabalhos Futuros

Este trabalho avaliou quatro cifras de fluxo leves padronizadas em uma Raspberry Pi 4, obtendo ordenação estatisticamente inequívoca, com coeficientes de variação inferiores a $2\,\%$. O Trivium alcançou 598,77 MiB/s (2,39 cpb) e o Enocoro-128v2, 57,08 MiB/s (25,06 cpb), diferença explicada por dois fatores: o número de operações elementares por byte de *keystream* e o paralelismo em nível de instrução exposto pela função de rodada ao núcleo superescalar. A análise evidencia que a orientação declarada a *hardware* ou a *software* é preditor insuficiente de desempenho: Trivium e Enocoro, ambos padronizados pela ISO/IEC 29192-3 como cifras orientadas a *hardware*, situam-se em extremos opostos, e o critério determinante é a função de rodada admitir avaliação paralela em palavras.

As limitações da Seção 4.3 delimitam os trabalhos futuros: avaliação em microcontroladores de execução em ordem; caracterização do regime de mensagens curtas com rechaveamento por pacote, no qual os custos de inicialização podem inverter a ordenação obtida; comparação com implementações vetorizadas em NEON; e análise do consumo de energia por byte cifrado.

Declaração sobre uso de Inteligência Artificial

Os autores declaram que não utilizaram ferramentas de inteligência artificial na elaboração deste artigo.

Referências

Bernstein, D. J. et al. (2008). Chacha, a variant of salsa20. In *Workshop record of SASC*, volume 8, pages 3–5. Lausanne, Switzerland.

Boesgaard, M., Vesterager, M., and Zenner, E. (2008). The rabbit stream cipher. In *New Stream Cipher Designs: The eSTREAM Finalists*, pages 69–83. Springer.

Bühler, H., Walz, A., and Sikora, A. (2022). Benchmarking of symmetric cryptographic algorithms on a deeply embedded system. *IFAC-PapersOnLine*, 55(4):266–271. 17th IFAC Conference on Programmable Devices and Embedded Systems PDES 2022 — Sarajevo, Bosnia and Herzegovina, 17-19 May 2022.

De Cannière, C. and Preneel, B. (2006). Trivium specifications. `https://web.archive.org/web/20161020205326/http://www.ecrypt.eu.org/stream/ciphers/trivium/trivium.pdf`. Acessado em: 8 mai. 2026.

Khan, S., Ferreira Lopes martins, P. A., Sousa, B., and Pereira, V. (2025). A comprehensive review on lightweight cryptographic mechanisms for industrial internet of things systems. *ACM Computing Surveys*, 58(1):1–37.

Manifavas, C., Hatzivasilis, G., Fysarakis, K., and Papaefstathiou, Y. (2016). A survey of lightweight stream ciphers for embedded systems. *Security and Communication Networks*, 9(10):1226–1246.

Nir, Y. and Langley, A. (2018). Chacha20 and poly1305 for ietf protocols. Technical report.

Robshaw, M. J. (1995). Stream ciphers. *RSA Labratories*, 25.

Silva, N. B., Pigatto, D. F., Martins, P. S., and Branco, K. R. (2016). Case studies of performance evaluation of cryptographic algorithms for an embedded system and a general purpose computer. *Journal of Network and Computer Applications*, 60:130–143.

Vaz, Y. S., Mattos, J. C. B., and Soares, R. I. (2025). Lightweight cryptography for the internet of things: A multi-metric experimental assessment. In *2025 XV Symposium on Computing Systems Engineering (SBESC)*, pages 1–6.

Watanabe, D., Owada, T., Okamoto, K., Igarashi, Y., and Kaneko, T. (2010). Update on enocoro stream cipher. In *2010 International Symposium On Information Theory & Its Applications*, pages 778–783. IEEE.