Viabilidade de Criptografia Pós-Quântica em Dispositivos IoT: Estudo de Caso Múltiplo com ESP32, ESP8266 e ARMv7

Wesley Rodrigues da Silva[1], **Eduardo Takeo Ueda**[1,2]

[1] Instituto de Pesquisas Tecnológicas do Estado de São Paulo (IPT)
Av. Prof. Almeida Prado, 532 – 05508-901 – São Paulo – SP – Brasil

[2]Departamento de Computação
Universidade Federal de São Carlos (UFSCar) – Sorocaba – SP – Brasil

wrodrigues.principal@gmail.com, eduardo.takeo@ufscar.br

Abstract. *The standardization of post-quantum cryptography (PQC) by NIST creates pressure to migrate the large installed base of low-cost IoT devices, yet related work usually reports timing and energy figures without recording whether the algorithms actually run on constrained hardware. This paper presents a multiple case study on the feasibility of running NIST-standardized digital signature algorithms on five units: two ARMv7 boards, an ESP32-S3, an ESP8266, and a high-end reference processor. We compare classical ECDSA against ML-DSA, FN-DSA, and SLH-DSA at NIST security level 3. The central finding is a concrete feasibility barrier: the ESP32 and ESP8266 could not execute ML-DSA or FN-DSA because both rely on SHAKE-256, while SLH-DSA, when it ran, took tens of seconds per signature. On ARMv7, in contrast, ML-DSA was even faster and more energy-efficient than ECDSA. Selecting PQC for an IoT product therefore requires checking support for the underlying hash primitive, not only the nominal security level.*

Resumo. *A padronização da criptografia pós-quântica (PQC) pelo NIST cria pressão para migrar a ampla base instalada de dispositivos IoT de baixo custo. Contudo, trabalhos correlatos costumam relatar tempo e energia sem registrar se os algoritmos de fato executam em hardware restrito. Este artigo apresenta um estudo de caso múltiplo sobre a viabilidade de executar algoritmos de assinatura digital padronizados pelo NIST em cinco unidades: duas placas ARMv7, um ESP32-S3, um ESP8266 e um processador de referência de alto desempenho. Comparamos o ECDSA clássico com ML-DSA, FN-DSA e SLH-DSA no nível de segurança 3 do NIST. O achado central é uma barreira concreta de viabilidade: o ESP32 e o ESP8266 não conseguiram executar ML-DSA nem FN-DSA, pois ambos dependem do SHAKE-256, enquanto o SLH-DSA, quando executou, levou dezenas de segundos por assinatura. Em ARMv7, por outro lado, o ML-DSA foi até mais rápido e mais eficiente energeticamente que o ECDSA. Escolher PQC para um produto IoT exige, portanto, verificar o suporte à primitiva de hash subjacente, e não apenas o nível de segurança nominal.*

Palavras-chave: *Internet das Coisas, Criptografia Pós-Quântica, Assinatura Digital, Estudo de Caso Múltiplo, Dispositivos Restritos.*

1. Introdução

A base instalada de dispositivos da Internet das Coisas (*Internet of Things*, IoT) cresce em ritmo acelerado: estimativas indicam a passagem de cerca de 5 bilhões de dispositivos conectados em 2018 para quase 40 bilhões em 2028 [Statista 2024]. Muitos desses dispositivos protegem suas comunicações com criptografia de chave pública, como o ECDSA, cuja segurança repousa sobre problemas que um computador quântico de larga escala poderá resolver em poucos anos [Iqbal and Zafar 2023]. Em resposta, o Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST) conduziu um processo de padronização de criptografia pós-quântica (*Post-Quantum Cryptography*, PQC) e publicou, em 2024, os padrões de assinatura digital ML-DSA e SLH-DSA [NIST 2024a, NIST 2024c], além do mecanismo de encapsulamento de chaves ML-KEM [NIST 2024b].

A transição para PQC, no entanto, esbarra em uma característica desses dispositivos: os novos algoritmos exigem mais processamento, memória e banda do que os clássicos [Fernández-Caramés 2020]. A RFC 7228 classifica os nós restritos em classes C0, C1 e C2 conforme memória e conectividade [Bormann et al. 2014], e dispositivos populares como o ESP32 e o ESP8266 situam-se na fronteira entre essas classes, o que torna incerta sua real capacidade de executar PQC.

Trabalhos correlatos medem tempo de execução e consumo de energia de algoritmos PQC em plataformas embarcadas [Prantl et al. 2021, Schöffel et al. 2021], mas raramente relatam *falhas* de execução ou explicam *por que* elas ocorrem. Esta lacuna motiva a pergunta de pesquisa deste artigo: dispositivos IoT restritos, como o ESP32 e o ESP8266, conseguem de fato executar os algoritmos de assinatura digital padronizados pelo NIST? Para respondê-la, conduzimos um estudo de caso múltiplo com cinco unidades de hardware e quatro algoritmos, priorizando a documentação das barreiras práticas de execução, e não apenas a comparação de desempenho.

2. Metodologia: Estudo de Caso Múltiplo

Adotou-se o estudo de caso múltiplo como estratégia metodológica, seguindo Yin [Yin 2018]. A escolha se justifica porque a pergunta de pesquisa é do tipo "como" e "por que" um fenômeno ocorre (a execução ou não de PQC), sobre um conjunto contemporâneo de artefatos reais que não podem ser abstraídos em um modelo genérico sem perda das particularidades de hardware. Cada dispositivo é tratado como um caso independente, e a comparação entre casos (*cross-case analysis*) permite distinguir o que é limitação de um chip específico daquilo que é um padrão recorrente entre plataformas restritas. Não se trata, portanto, de um *survey* nem da proposta de um *framework* genérico, mas da investigação empírica de unidades concretas.

2.1. Unidades de análise

Foram selecionadas cinco unidades que representam faixas distintas de capacidade computacional, descritas na Tabela 1. As duas placas ARMv7 (Rockchip RV1103G1 e RV1106G2, arquitetura ARMv7 combinada com RISC-V) executam GNU/Linux e representam nós IoT de médio porte. O *m5stack* (ESP32-S3) e o *nodemcu* (ESP8266) representam os microcontroladores de baixo custo mais difundidos em aplicações residenciais e industriais. O *ryzen* (AMD Ryzen AI 9 HX 370) atua como caso de referência (*baseline*): por ser um processador não restrito, permite observar o comportamento esperado de cada algoritmo em um cenário sem limitação de recursos.

Tabela 1. Características das unidades de análise. Aceleradores indica presença de blocos de hardware para AES, ECC, RSA, SHA e RNG.

Dispositivo	Classe	Cores	MHz	Bits	RAM (MB)	Aceleradores
arm_pequeno	C2	1	1200	32	64	Sim
arm_grande	C2	1	1200	32	128	Sim
nodemcu	C1	1	80	32	0,064	Não
m5stack	C2	2	240	32	0,52	Sim
ryzen	NR	24	5000	64	32000	Sim

2.2. Algoritmos e nível de segurança

Comparou-se o ECDSA clássico (secp256r1 com SHA-256) com três algoritmos padronizados pelo NIST: ML-DSA (baseado em reticulados), FN-DSA (Falcon-512, também em reticulados) e SLH-DSA (SPHINCS+, baseado em funções de *hash*). Todos foram configurados no nível de segurança 3 do NIST, garantindo uma comparação justa. A Tabela 2 resume seus parâmetros. Destaca-se a função de *hash* subjacente: ML-DSA e FN-DSA empregam SHAKE-256, ao passo que ECDSA e a variante escolhida de SLH-DSA usam SHA-256 — distinção determinante para os resultados observados.

Tabela 2. Parâmetros dos algoritmos avaliados. PK: chave pública; SK: chave privada; Assin.: assinatura (tamanhos em *bytes*).

Algoritmo	Hash	Família	PK	SK	Assin.
ECDSA (secp256r1)	SHA-256	Curvas elípticas	64	32	128
ML-DSA-44	SHAKE-256	Reticulados (LWE)	1312	2528	2420
FN-DSA (Falcon-512)	SHAKE-256	Reticulados (NTRU)	897	1281	666
SLH-DSA (SPHINCS+-128s)	SHA-256	Funções de hash	32	64	7856

2.3. Configuração experimental

Os algoritmos PQC foram implementados com o projeto PQClean [Kannwischer et al. 2022], que oferece implementações em C padronizadas e testadas, sem alocação dinâmica de memória; o ECDSA usou a biblioteca Micro-ECC (uECC) [MacKay 2025], voltada a plataformas restritas e aderente aos formatos do SECG [Certicom Research 2009]. Nas placas ARMv7, o código foi compilado com GCC, gerando um binário ELF de 32 bits transferido via SSH. O ESP32 foi programado com o ESP-IDF e o ESP8266 com o SDK baseado em FreeRTOS, ambos por meio do PlatformIO; nesses microcontroladores o firmware é recompilado a cada troca de algoritmo.

As medições elétricas de corrente e voltagem foram feitas com um módulo INA226 controlado por um Arduino via I2C, conectado em série com uma fonte estabilizada; a temperatura ambiente foi mantida em $25\,^{\circ}\mathrm{C}$. Cada operação (geração de chaves, *hash*, assinatura e verificação) foi repetida múltiplas vezes para estabilizar as medianas reportadas, com meta de 1.000 execuções por algoritmo em cada dispositivo.

3. Execução e Problemas Encontrados

O achado prático central deste estudo emergiu já na fase de execução, antes de qualquer análise de desempenho: nem todos os dispositivos conseguiram rodar todos os algoritmos. A Tabela 3 sintetiza o resultado de execução por algoritmo e dispositivo.

Tabela 3. Resultado de execução por algoritmo e dispositivo. Completa: todas as iterações concluídas; Parcial: execução incompleta; Inviável: concluída com tempo impraticável; Falha: erro/interrupção antes de concluir.

Algoritmo	Hash	arm_pequeno	arm_grande	m5stack	nodemcu	ryzen
ECDSA	SHA-256	Completa	Completa	Completa	Completa	Completa
ML-DSA	SHAKE-256	Completa	Completa	Falha	Falha	Completa
FN-DSA	SHAKE-256	Completa	Completa	Falha	Falha	Completa
SLH-DSA	SHA-256	Parcial	Parcial	Inviável	Falha	Completa

ESP32 e ESP8266 e a barreira do SHAKE-256. O *m5stack* (ESP32) e o *nodemcu* (ESP8266) não conseguiram executar ML-DSA nem FN-DSA. Em ambos os casos, o processamento foi interrompido com mensagens de erro. A causa comum é a dependência dos dois algoritmos do *hash* SHAKE-256, cujas necessidades de CPU e memória excedem o que esses microcontroladores oferecem. Trata-se de uma limitação da primitiva de *hash*, e não do paradigma criptográfico em si: o SLH-DSA, que na variante avaliada usa SHA-256 (com aceleração de hardware disponível no ESP32), chegou a executar no *m5stack*.

SLH-DSA: executa, mas é inviável. No *m5stack*, o SLH-DSA rodou sem falhas de memória ou de CPU, porém com tempos proibitivos: a mediana foi de cerca de 10 segundos para gerar chaves e de aproximadamente 78 segundos para produzir uma única assinatura, contra frações de milissegundo do ECDSA no mesmo dispositivo. Não houve *crash*, mas o custo torna o algoritmo inviável para aplicações reais. No *nodemcu*, ainda mais restrito, o SLH-DSA não concluiu.

ARMv7 executou quase tudo. As placas ARMv7 (*arm_pequeno* e *arm_grande*) executaram ECDSA, ML-DSA e FN-DSA integralmente, tendo apresentado apenas execuções parciais do SLH-DSA (o tempo elevado impediu completar todas as iterações planejadas). Somente o *ryzen*, o caso de referência, executou os quatro algoritmos por completo.

Esse contraste é o cerne da contribuição: uma barreira de viabilidade que não aparece em *benchmarks* conduzidos apenas em servidores ou PCs, nos quais todos os algoritmos simplesmente executam. Em hardware restrito, a pergunta relevante deixa de ser "qual algoritmo é mais rápido?" e passa a ser "qual algoritmo consegue, de fato, executar?".

4. Resultados Quantitativos

Para os casos em que houve execução, a Tabela 4 apresenta as medianas de tempo por operação e o consumo elétrico (voltagem e corrente). Os tempos estão em milissegundos. As linhas ausentes correspondem às combinações que falharam (Tabela 3).

Tabela 4. Medianas de tempo (ms) por operação e consumo elétrico por algoritmo e dispositivo.

Algoritmo	Dispositivo	Chaves	Hash	Assin.	Verif.	V	mA
ECDSA	arm_pequeno	6,562	0,001	6,960	7,702	5,29	81,71
ECDSA	arm_grande	6,077	0,002	6,452	7,151	5,22	102,30
ECDSA	m5stack	0,002	0,005	0,213	114,945	5,21	163,86
ECDSA	nodemcu	0,012	0,001	0,149	0,121	5,37	23,65
ML-DSA	arm_pequeno	2,095	0,020	4,828	2,031	5,20	84,03
ML-DSA	arm_grande	1,925	0,019	4,432	1,881	5,22	77,53
FN-DSA	arm_pequeno	136,987	0,033	39,194	0,607	5,20	80,89
FN-DSA	arm_grande	125,806	0,032	36,328	0,565	5,21	100,87
SLH-DSA	arm_pequeno	730,655	0,002	5545,040	5,522	5,21	81,68
SLH-DSA	arm_grande	678,616	0,002	5148,876	5,160	5,21	102,60
SLH-DSA	m5stack	10240,835	0,000	77828,994	76,160	5,23	79,97

Os números confirmam a inviabilidade do SLH-DSA em hardware restrito: no *m5stack*, a assinatura tem mediana próxima de 78 segundos, cerca de cinco ordens de grandeza acima da assinatura ECDSA no mesmo dispositivo. Nas placas ARMv7, o SLH-DSA também é o mais custoso, com assinaturas na faixa de 5 segundos.

Um achado adicional, porém, aponta na direção oposta e é igualmente relevante. Nas placas ARMv7, o ML-DSA superou o ECDSA em quase todas as operações: foi mais rápido na geração de chaves (por exemplo, 1,925 ms contra 6,077 ms no *arm_grande*), na assinatura e na verificação, sendo apenas ligeiramente mais lento na função de *hash*. Além disso, consumiu corrente igual ou menor (77,53 mA contra 102,30 mA no *arm_grande*). Ou seja, quando o dispositivo suporta a primitiva de *hash* exigida, o PQC baseado em reticulados pode ser não só viável, mas vantajoso frente ao algoritmo clássico, tanto em tempo quanto em energia.

5. Discussão e Implicações Práticas

O conjunto de resultados leva a uma implicação de projeto direta: a escolha de um algoritmo PQC para um produto IoT não pode se basear apenas no "nível de segurança" nominal. Dois algoritmos no mesmo nível 3 do NIST — ML-DSA e SLH-DSA — tiveram destinos opostos nos mesmos microcontroladores, e a diferença decisiva foi a primitiva de *hash*: o SHAKE-256, exigido por ML-DSA e FN-DSA, simplesmente não executou no ESP32 e no ESP8266. Verificar o suporte efetivo à primitiva subjacente é, portanto, um requisito prático tão importante quanto o nível de segurança.

A partir dos casos observados, delineiam-se três caminhos para quem precisa de assinatura pós-quântica em dispositivos restritos. O primeiro é incorporar um coprocessador criptográfico dedicado, capaz de acelerar SHAKE-256 e as operações mais custosas. O segundo é investir em otimização de biblioteca, adaptando implementações às limitações de memória e ao conjunto de instruções do alvo. O terceiro, mais conservador, é reconhecer que dispositivos de classe C1 segundo a RFC 7228 [Bormann et al. 2014], como o ESP8266, podem simplesmente não ser candidatos adequados à PQC de assinatura no estado atual das bibliotecas, permanecendo com soluções clássicas enquanto durar a viabilidade destas.

O resultado favorável do ML-DSA em ARMv7 reforça que a barreira não é intrínseca ao PQC, e sim ao acoplamento entre algoritmo, biblioteca e hardware. Onde há recursos e aceleradores adequados, a migração pode ocorrer sem penalidade de desempenho — e até com ganho de eficiência energética.

6. Conclusão

Este estudo de caso múltiplo investigou a viabilidade de executar algoritmos de assinatura digital padronizados pelo NIST em cinco unidades de hardware, do ESP8266 a um processador de alto desempenho. Mais do que medir desempenho, a estratégia metodológica permitiu identificar uma barreira concreta de viabilidade: o ESP32 e o ESP8266 não executam ML-DSA nem FN-DSA, e a hipótese fortemente suportada pelos resultados é que isso ocorre por dependerem do SHAKE-256. O SLH-DSA, embora execute, é inviável por seu tempo longo. Em contraste, as placas ARMv7 executaram quase todos os algoritmos, e o ML-DSA chegou a superar o ECDSA em tempo e energia.

A principal contribuição não é um ranking de velocidade, e sim a evidência de que decisões de migração para PQC em IoT devem considerar o suporte às primitivas subjacentes e as reais capacidades do hardware no campo, e não apenas as classificações teóricas ou o nível de segurança nominal. Como trabalhos futuros, pretende-se avaliar coprocessadores dedicados e implementações otimizadas de SHAKE-256 para dispositivos de classe C1 e C2.

Referências

Bormann, C., Ersue, M., and Keranen, A. (2014). Terminology for Constrained-Node Networks. RFC 7228.

Certicom Research (2009). Standards for efficient cryptography SEC 1: Elliptic curve cryptography. Technical report, Certicom Corp. Version 2.0; Contact: Daniel R. L. Brown.

Fernández-Caramés, T. M. (2020). From pre-quantum to post-quantum iot security: A survey on quantum-resistant cryptosystems for the internet of things. *IEEE Internet of Things Journal*, 7(7):6457–6480.

Iqbal, S. S. and Zafar, A. (2023). A survey on post quantum cryptosystems: Concept, attacks, and challenges in iot devices. In *Proceedings of [...]*, pages 460–465, New Delhi, India. IEEE.

Kannwischer, M. J., Schwabe, P., Stebila, D., and Wiggers, T. (2022). Improving software quality in cryptography standardization projects. In *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022*, pages 19–30, Los Alamitos, CA, USA. IEEE Computer Society.

MacKay, K. (2025). kmackay/micro-ecc.

NIST (2024a). *Module-Lattice-Based Digital Signature Standard*. Number FIPS 204. Gaithersburg, MD.

NIST (2024b). *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Number FIPS 203. Gaithersburg, MD.

NIST (2024c). *Stateless Hash-Based Digital Signature Standard*. Number FIPS 205. Gaithersburg, MD.

Prantl, T., Prantl, D., Bauer, A., Iffländer, L., Dmitrienko, A., Kounev, S., and Krupitzer, C. (2021). Benchmarking of pre- and post-quantum group encryption schemes with focus on IoT. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–10. ISSN: 2374-9628.

Schöffel, M., Lauer, F., Rheinländer, C. C., and Wehn, N. (2021). On the energy costs of post-quantum KEMs in TLS-based low-power secure IoT. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, IoTDI '21, pages 158–168. Association for Computing Machinery.

Statista (2024). Internet of things (IoT) connected devices installed base worldwide from 2015 to 2025.

Yin, R. K. (2018). *Case Study Research and Applications: Design and Methods*. SAGE Publications, Thousand Oaks, CA, 6 edition.