Estudo comparativo de desempenho em Assinaturas Digitais Pós-Quânticas para plataformas de IoT

Vitor Nagata[1], **Marco A. A. Henriques**[1]
nagatavit@gmail.com, marco@dca.fee.unicamp.br

[1]Research Group on Applied Security (ReGrAS)
Departamento de Engenharia de Computação e Automação Industrial (DCA)
Universidade Estadual de Campinas (UNICAMP) - Campinas, SP

Abstract. *With the fast advance of the quantum computing field, there has been a growing concern about the security of the current asymmetric cryptography schemes such as RSA and ECC. To maintain the security of the current protocols, the search for quantum resistant algorithms has already started. In this paper, we present some of the ideas behind the new post-quantum signature algorithms, and compare their performances in two contexts: One in a desktop environment and other tailored for IoT applications.*

Resumo. *Grandes avanços no campo da computação quântica têm aumentado a preocupação com a quebra de algoritmos criptográficos atuais, como RSA e ECC. Para manter a segurança e dos protocolos atuais, já estão sendo pesquisados algoritmos resistentes a ataques quânticos. Neste artigo, apresentamos algumas das ideias por trás dos novos algoritmos de assinatura pós-quânticos e comparamos seus desempenhos em dois contextos: um em ambiente de desktop e outro voltado para aplicações IoT.*

1. Introdução

O crescente avanço no ramo da computação quântica nos últimos anos traz consigo uma grande preocupação com a segurança dos esquemas criptográficos atuais, especialmente os baseados em primitivas assimétricas como o problema da fatoração de números inteiros (RSA), logaritmo discreto (Diffie-Hellman, Elgamal, DSA) e curvas elípticas (ECDH, ECDSA), devido ao algoritmo de Shor [Shor 1999]. De forma a prevenir os ataques possibilitados pela computação quântica, diversos esforços já estão sendo feitos, tais como, iniciativas de adoção de chaves maiores, propostas de novos protocolos baseados nas premissas atuais (Curvas Elípticas Super-singulares [Jao and De Feo 2011]) e principalmente, propostas de novas premissas criptográficas, os chamados Algoritmos Criptográficos Pós-Quânticos (*PQC - Post-Quantum Cryptography*).

No ramo da PQC, o principal esforço se concentra no processo de padronização sendo realizado pelo NIST [NISTPQC 2016] que, de forma similar aos processos realizados para o AES e SHA3, busca encontrar a(s) melhor(es) proposta(s) resistente(s) a ataques de algoritmos quânticos. Essas propostas contudo, possuem adversidades como grandes tamanhos de chaves e assinaturas, ou longos tempos de assinatura e verificação.

Em paralelo ao avanço da computação quântica, também estamos próximos de uma revolução em questão de conectividade com a proximidade de implementações das

tecnologias 5G. Isso significa que cada vez mais, mais dispositivos com baixo poder computacional e capacidade armazenamento estarão conectados em rede, muitas vezes trafegando dados sensíveis que necessitam de autenticação como sensores de biometria ou controles de acesso, na chamada Internet das Coisas (*IoT - Internet of Things*).

Neste trabalho, apresentamos as principais ideias por trás dos novos algoritmos pós-quânticos e a realização de alguns ensaios para verificar as vantagens e desvantagens das propostas de assinaturas digitais pós-quânticas em cenários mais específicos como o de IoT. Alguns trabalhos na literatura já foram realizados para a medição de desempenho de algoritmos pós-quânticos como o trabalho de Hyeongcheol An [An et al. 2018] e de projetos como eBACS [Bernstein and Lange 2019]. Nesses trabalhos o foco do estudo foi o desempenho geral das propostas ou sua viabilidade de uso em protocolos atuais. Nosso objetivo com este artigo é uma discussão de métricas práticas para o cenário de IoT como os requisitos mínimos necessários para que haja a viabilidade de implementação mas não será feita uma análise detalhada da segurança como está sendo realizado pelo NIST.

O artigo está estruturado da seguinte maneira: nas Seções 2 e 3 apresentamos uma breve contextualização do processo de padronização realizado pelo NIST e do ambiente a ser estudado; na Seção 4 são detalhadas as ideias por trás das novas primitivas criptográficas, na Seção 5 apresentamos as ferramentas utilizadas e o ambiente de testes; por fim, nas Seções 6, 7 e 8 temos a análise de resultados, conclusões e trabalhos futuros.

2. Chamada de contribuições NIST

Ao final de 2016, devido às perspectivas futuras para o avanço na computação quântica, o NIST resolveu iniciar seu processo de padronização para algoritmos pós-quânticos. A chamada pública foi motivada por dois fatores: uma rápida evolução da computação quântica nos últimos anos e a possível dificuldade de transição dos os algoritmos atuais para os pós-quânticos, dada as arquiteturas novas bem distintas das atuais.

2.1. Níveis de Segurança NIST

Na chamada de propostas do processo de padronização realizado pelo NIST [NISTPQC 2016], foi estabelecido um critério para o nível de segurança dos algoritmos a serem submetidos conforme a tabela 1.

Tabela 1. Níveis de Segurança (Adaptado de [NISTPQC 2016])

Nível de segurança	Descrição
I	Equivalente a um cifrador simétrico de 128 bits
II	Equivalente a colisão de um hash de 256 bits
III	Equivalente a um cifrador simétrico de 192 bits
IV	Equivalente a colisão de um hash de 384 bits
V	Equivalente a um cifrador simétrico de 256 bits

Por ser um ramo ainda em desenvolvimento, não é possível determinar a complexidade total dos ataques possibilitados pela computação quântica, ou a existência de vulnerabilidades novas. Como um critério mais concreto, foram estimados custos computacionais equivalentes aos algoritmos simétricos atuais como AES128 e SHA256. Apesar da comparação com algoritmos simétricos, os níveis de segurança também são válidos

para o cenário de assinaturas que será o nosso caso de estudo.

A recomendação inicial do NIST foi a de concentrar as submissões nos níveis de segurança I, II e III, os quais seriam suficientemente seguros para o futuro próximo mas, se possível, que fossem fornecidos parâmetros com provas de segurança acima do nível III para garantir a segurança no longo prazo. Além da segurança, outros critérios também serão julgados como a viabilidade em protocolos atuais como TLS e, para o caso de assinaturas, a formalização de *Existential Unforgeability under Chosen Message Attack*.

2.2. Candidatos da segunda rodada

No momento da escrita deste artigo, temos na segunda rodada do processo de padronização NIST, nove algoritmos PQC para assinaturas digitais. Não entraremos em detalhes neste artigo sobre o funcionamento de cada proposta, mas estão indicadas suas características principais na Tabela 2. Para detalhes do funcionamento de cada uma, é necessário consultar a documentação respectiva submetida ao processo do NIST.

Tabela 2. Candidatos da segunda rodada NIST PQC

Proposta	Características	Premissa
Crystals Dilithium	Uso de heurística *Fiat-Shamir with aborts* Baseada no problema do *LWE*[1] modular	Reticulados
Falcon	Estrutura de dados especial: *Falcon tree* Baseada no reticulado do *NTRU*	Reticulados
qTesla	Baseada em LWE[1] sobre aneis	Reticulados
GeMSS	Baseada em problemas *Hidden Field Equations*	Polinômios Multivariados
LuoV	Baseada no problema do *UOV*[2]	Polinômios Multivariados
MQDSS	Baseada no problema *Multivariate Quadratic* Utiliza heurística de *Fiat-Shamir*	Polinômios Multivariados
Rainbow	Generalização do *UOV*[4] Utiliza camadas de sistema de equações	Polinômios Multivariados
Picnic	Baseada em *Zero Knowlegde Proof*	Funções hash e Primitivas simétricas
Sphincs+	Utiliza estruturas *WOTS+*[3] e *FORS*[4]	Funções hash

3. Cenário de assinaturas em Internet das Coisas (*IoT - Internet of Things*)

Em um cenário de sistemas em Internet das Coisas, além da segurança, há uma grande preocupação com a dificuldade computacional de implementação da mesma. Devido às fortes restrições no processamento e capacidade de memória, há a necessidade do uso de algoritmos que economizem nos tamanhos ou tempos para assinatura e verificação a depender do que for mais crítico para a operação.

[1] *Learning With Errors*
[2] *Unbalanced Oil and Vinegar*
[3] *Winternitz one-time signature*
[4] *Forest Of Random Subsets*

Um cenário muito característico em sistemas IoT consiste normalmente de diversos nós sensores / atuadores e um nó agregador central que realiza o processamento dos dados. Nesse cenário, tamanhos e tempos de assinatura são os fatores mais críticos pelo consumo de banda e pelas restrições de processamento dos sensores. Já chaves públicas e tempos de verificação são fatores menos críticos, já que pela natureza do ambiente, o nó central já estaria configurado com as chaves públicas ou só seria necessário o envio das chaves na primeira conexão. Além disso, o nó central possui um poder computacional maior do que os outros nós e, portanto, a exigência no processo de verificação pode ser mais alta. Este será nosso cenário base para discussão dos resultados obtidos.

4. Premissas Criptográficas

Os algoritmos presentes na segunda rodada podem ser classificados em três categorias de primitivas criptográficas distintas:

- algoritmos baseados em reticulados;
- algoritmos baseados em polinômios multivariados;
- algoritmos baseados em hash ou primitivas simétricas.

Nelas, são explorados problemas matemáticos que são computacionalmente inviáveis de serem resolvidos mesmo com a presença de algoritmos quânticos conhecidos atualmente. Essas premissas são detalhadas a seguir.

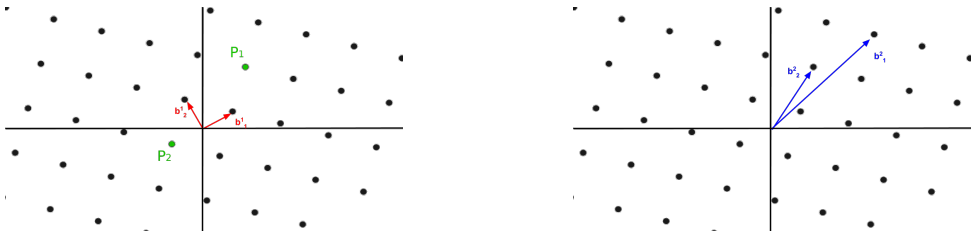
4.1. Assinaturas baseadas em reticulados

A definição matemática de um reticulado é dada pelo conjunto de pontos formados pela combinação linear de uma base. Matematicamente, temos:

$$L(\mathbf{b_1}, \mathbf{b_2}, \ldots) = \sum_{i=1}^{n} x_i \cdot \mathbf{b_i}, \ \forall x_i \in \mathbb{Z}$$

Onde o reticulado L de dimensão n é formado pela combinação linear da base $\mathbf{B}$ para quais quer valores inteiros de x_i

De forma intuitiva, podemos imaginar um reticulado como uma malha periódica de pontos no espaço $\mathbb{Z}^n$, onde os pontos são as combinações lineares de vetores da base como mostra a Figura 1(a) para o caso bidimensional.



(a) Reticulado formado pela base $\mathbf{B^1}$ (b_1^1, b_2^1) **(b) Reticulado formado pela base $\mathbf{B^2}$ (b_1^2, b_2^2)**

Figura 1. Reticulado formado por duas bases distintas

Na Figura 1(a), podemos observar que os pontos do reticulado mostrado podem ser obtidos pela combinação linear dos vetores da base $\mathbf{B^1}$. Como exemplo, podemos representar dois pontos desse reticulado por:

$$P_1 = 2 \cdot b_1^1 + 1 \cdot b_2^1 \quad e \quad P_2 = -1 \cdot b_1^1 + 0 \cdot b_2^1$$

É possível provar, através de transformações matriciais, que um mesmo reticulado pode ser formado por bases distintas, como por exemplo, com uso de uma matriz unimodular [Goldreich et al. 1997]. Essa característica é ilustrada visualmente pelas Figuras 1(a) e 1(b), onde um mesmo reticulado é construído a partir de duas bases diferentes.

A existência de múltiplas bases para um mesmo reticulado é uma das características que nos permite a criação de funções *trapdoor* (ou de mão única), onde é possível realizar uma operação mais facilmente com o uso de informações adicionais do que na ausência delas como é o caso do RSA, DH e ECC. Para reticulados é possível explorar o uso de bases ortogonais e não ortogonais formando um mesmo reticulado na resolução de problemas em que a complexidade dependa da facilidade de geração do reticulado a partir da base, como os problemas do *SVP* [Ajtai 1996], *CVP* [Ajtai 1996] e *LWE*[Regev 2009]. Esses problemas são melhores detalhados a seguir.

4.1.1. Reticulados aplicados a criptografia

Quando falamos sobre aplicações criptográficas dos reticulados, temos três principais problemas possíveis de serem utilizados como funções *trapdoor*:

- *SVP - Short Vector Problem*
  - achar o menor vetor (não nulo) pertencente ao reticulado;
- *CVP - Closest Vector Problem*
  - dado um ponto no espaço, achar o vetor pertencente ao reticulado mais próximo à esse ponto;
  - pode ser visto como uma variação do SVP com o centro do reticulado deslocado;
- *LWE - Learning with errors*
  - **Versão de busca:** dado um vetor corrompido intencionalmente com ruido, recuperar o vetor original;
  - **Versão de decisão:** dado um vetor corrompido intencionalmente com ruído, saber distinguir se o ruído é parte da codificação ou se é realmente algum ruído da comunicação;

A aplicação dos três problemas acima citados se baseia na característica de um mesmo reticulado poder ser representado por múltiplas bases distintas. A dificuldade computacional de se achar um ponto pertencente a um reticulado, ou achar o menor vetor pertencente ao reticulado próximo a um ponto arbitrário do espaço dependerá da base escolhida para a resolução do problema.

Para bases mais ortogonais, é mais fácil obter todos os pontos pertencentes ao reticulado e, consequentemente, encontrar o vetor mais próximo a um dado ponto arbitrário, enquanto que, para bases pouco ortogonais, é extremamente difícil garantir que um vetor é o menor possível no reticulado ou que este vetor é o mais próximo de um ponto arbitrário, mas é fácil conferir que o vetor pertence ao reticulado.

O enunciado do SVP diz que, dado uma base pouco ortogonal de um reticulado, é inviável de se obter o menor vetor pertencente a este reticulado para altas dimensões. Esta definição foi provada por Miklós Ajtai [Ajtai 1996] que mostrou que mesmo com o uso de uma redução quântica o problema continua inviável.

O problema do CVP é próximo ao SVP, sendo possível considerá-lo como uma o problema SVP com o centro do reticulado deslocado. O enunciado do CVP dita que, para uma base pouco ortogonal e um ponto arbitrário P, é inviável obter o vetor pertencente ao reticulado mais próximo ao ponto P. Algoritmos atuais para a resolução desse problema como o algoritmo de Babai [Babai 1986] não funcionam para bases não ortogonais.

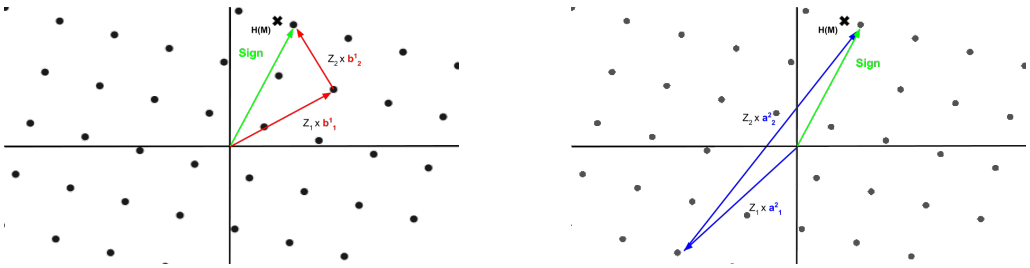
Por fim, o problema do LWE é uma aplicação do CVP, inspirada pelos códigos corretores de erros utilizados em telecomunicações. O LWE foi primeiramente proposto por Oded Regev [Regev 2009] com a ideia da utilização da adição de ruídos aos vetores do reticulado, de forma que apenas bases "boas" (i.e. ortogonais) conseguiriam recuperar o vetor original, enquanto que bases "ruins" (pouco ortogonais) não seriam capazes de obter o vetor mais próximo ao ruído de forma satisfatória.

4.1.2. Exemplo de assinatura utilizando CVP

Para ilustrar uma assinatura utilizando CVP, podemos imaginar um ponto aleatório no espaço $\mathbb{R}^2$ como sendo um *hash* da mensagem a ser assinada: podemos considerar, por exemplo, os primeiros $N/2$ bits do *hash* como a coordenada x e os últimos $N/2$ bits como a coordenada y, sendo N o número de bits do *hash* da mensagem.

Considerando as bases das Figuras 1(a) ($\mathbf{B^1}$) e 1(b) ($\mathbf{B^2}$) como respectivamente, a chave privada e a chave pública, e um ponto aleatório $H(X)$ como sendo o *hash* da mensagem X, é fácil determinar o vetor mais próximo do ponto $H(X)$ utilizando a base $\mathbf{B^1}$ por ser uma base ortogonal. Por outro lado, não é trivial determinar o menor vetor utilizando $\mathbf{B^2}$, mas é muito fácil verificar que o vetor achado por $\mathbf{B^1}$ é um ponto válido.

Portanto, um esquema de assinatura baseado em *CVP* pode ser construído dessas duas operações: a assinatura consiste em encontrar o ponto do reticulado mais próximo ao *hash* da mensagem x, sendo o ponto encontrado, a própria assinatura a ser enviada (Figura 2(a)) e, a verificação consiste em verificar se esse ponto pertence ao reticulado e está suficientemente próxima ao *hash* por algum critério de distância (Figura 2(b)).



(a) Assinatura a partir de uma base ortogonal **(b) Verificação com o uso de uma base pouco ortogonal**

Figura 2. Ilustração de assinatura utilizando o problema do CVP

4.2. Assinaturas baseadas em polinômios multivariados

Polinômios multivariados se utilizam de um problema há muito tempo estudado na matemática e computação que é a resolução de sistemas de equações. Um esquema de assinatura baseado em polinômios multivariados consiste, de forma geral, no uso de transformações não lineares dos dados através de sistemas de polinômios multivariados. Portanto, nesse sistema, temos a seguinte configuração:

- operações feitas sobre um corpo finito e.g. $GF(2^8)$
- chaves públicas e privadas definidas como um sistema de polinômios multivariados

Podemos então representar uma mensagem de k bits a ser assinada por

$$M = (x'_1, \ldots, x'_n)$$

sendo cada $x_i,\ i \in [1, n], n \leq k$ elementos de um corpo finito, e.g. $GF(2^8)$.

A chave pública é definida por um conjunto de m polinômios com n variáveis:

$$G(x_1, \ldots, x_n) = (G_1(x_1, \ldots, x_n), \ldots, G_m(x_1, \ldots, x_n))$$

Para realizar uma cifração da mensagem M, basta computar o polinômio da chave pública avaliado com o vetor da mensagem:

$$\begin{aligned} G(M) &= G(x_1, \ldots, x_n) \\ &= (G_1(x_1, \ldots, x_n), \ldots, G_m(x_1, \ldots, x_n)) \\ &= (y_1, \ldots, y_m) = Y \end{aligned}$$

onde $Y = (y_1, \ldots, y_m)$ é o texto cifrado obtido a partir da mensagem M

A chave privada pode ser então definida por um conjunto de polinômios de n equações e m variáveis "inversos" que retorne a mensagem original

$$G^{-1}(y_1, \ldots, y_m) = (G_1^{-1}(y_1, \ldots, y_m), \ldots, G_n^{-1}(y_1, \ldots, y_m))$$

$$G^{-1}(Y) = M$$

A cifração e decifração (ou a assinatura e verificação) nesses esquemas é dada pela simples avaliação da mensagem pelas chaves públicas e privadas. A segurança desses esquemas está na dificuldade com que esses polinômios públicos / privados são construídos.

O problema mais conhecido para a construção das chaves públicas e privadas é baseado no esquema *UOV - Unbalanced Oil and Vinegar* e seus derivados como proposto por Jacques Patarin et al [Patarin et al. 1999]. A ideia por trás desses esquemas é a construção de polinômios multivariados, normalmente de ordem quadrática, com mais variáveis do que equações. Para se obter uma solução desse sistema, é necessário fixar algumas das variáveis e é nessa característica que podemos montar as funções de *trapdoor*. Podemos montar as chaves privadas com a escolha aleatória de variáveis extras e resolver o sistema para determinar a pré imagem da chave pública. Além da escolha de variáveis para resolução do sistema, também é necessário a aplicação de uma função que "misture" os conjuntos de variáveis para que não seja possível a inversão sem o conhecimento dessas variáveis extras.

4.3. Assinaturas baseadas em Hash e primitivas simétricas

Outra primitiva utilizada nas novas propostas é o uso de funções de *hash* ou primitivas simétricas. A origem das assinaturas baseadas em *hash* remontam próximas ao início da criptografia assimétrica, sendo consideradas uma das primitivas mais antigas e maduras. As primeiras propostas desses esquemas [Lamport 1979] e [Merkle 1989] são

os primeiros modelos da utilização de funções de sentido único para assinaturas digitais e normalmente são utilizadas como bases para esquemas mais complexos.

Um dos grandes problemas desses primeiros esquemas de assinaturas baseadas em *hash* é o fato de serem do tipo *OTS - One Time Signature*, ou seja, para cada par de chaves públicas e privadas, só é possível realizar uma assinatura sem que haja comprometimento da segurança. Os esquemas mais recentes se utilizam de artifícios como a geração de diversas chaves privadas e a utilização de parte delas a cada assinatura, podendo ser realizadas múltiplas assinaturas ao invés de só uma. Um dos elementos principais nesses esquemas é o uso de árvores de *Merkle*, que são estruturas úteis para resumir uma grande quantidade de chaves em uma única. A grande desvantagem desses contornos é o aumento significativo no tamanho das chaves e tempos de assinatura.

O uso de primitivas simétricas não chegou a ser estudado de forma aprofundada neste trabalho, mas há uma proposta na segunda rodada baseada em provas de conhecimento zero com o uso de cifradores simétricos e funções *hash* [Chase et al. 2017].

5. Avaliação de desempenho dos algoritmos da segunda rodada

Após estudar e entender as novas premissas criptográficas, buscamos avaliar o desempenho prático das propostas concorrendo a padronização NIST. Para tal, optamos pelo uso da biblioteca libOQS [Stebila and Mosca 2016], que possui implementação dos principais algoritmos da segunda rodada e um conjunto de testes para medir a velocidade e tamanhos de chaves e assinaturas, o qual foi a base para os nossas discussões.

Apesar de facilitar o processo das medições de tempos, fornecendo um programa comum para os algoritmos implementados, o uso desta biblioteca possui algumas desvantagens, dentre elas: não implementar alguns dos candidatos do processo, ser uma biblioteca de uso geral, não específica para ambientes embarcados e não possuir testes para medição de tamanhos de chaves e assinaturas. As informações obtidas pelos testes do libOQS, são as especificadas nas respectivas documentações dos algoritmos, e não os valores medidos na prática.

Antes da escolha dessa biblioteca para nossos testes, nossa primeira tentativa de estudo foi com o uso das implementações de referência submetidas ao NIST, já que gostaríamos de avaliar a performance dos algoritmos puros, sem a utilização de otimizações específicas, uma vez que este possivelmente seria o caso das plataformas em internet das coisas. Contudo, devido a heterogeneidade no formato das submissões, como o uso de algumas dependências externas, não foi possível a criação de um ambiente de testes comum a todas as submissões no período de nosso estudo.

5.1. Ambiente de testes

Como citado anteriormente, os testes foram realizados com o auxilio da libOQS em duas plataformas distintas. Os primeiros testes foram realizados com a seguinte configuração: Intel i7-8550U (1.8-4.0 GHz), 8 GB RAM, Arch Linux, libOQS (v0.4.0-dev) compilado com gcc (v10.1.0) e uso de funções auxiliares (AES, SHA-2 SHA-3) do OpenSSL (1.1.1g).

Os testes embarcados foram realizados em uma raspberry pi 3 B, com um processador ARM 64-bits, quad-core (1.2 GHz), 1 GB RAM, Raspbian (v8.0), com libOQS (v0.4.0) compilado sem optimizações de cpu (GCC 8.4.0), e sem funções do OpenSSL.

6. Resultados

Os resultados do conjunto de testes da libOQS são mostrados a seguir. Os gráficos mostram a comparação dos algoritmos em seus diferentes níveis de segurança especificados pelo NIST. O teste é realizado medindo-se os tempos de assinatura para vetores aleatórios de 50 bytes. Alguns algoritmos estão ausentes nos gráficos da Raspberry por falta de implementação da própria biblioteca e no caso das medições no i7, o *Sphincs+* se encontra ausente pela falta de aprofundamento que tivemos durante nosso período de estudo, não conseguindo especificar os níveis de segurança dessa proposta para realizar uma comparação justa. Um ponto a ser destacado é que os gráficos estão em escala logarítmica dado que algoritmos de classes diferentes possuem resultados bem discrepantes.

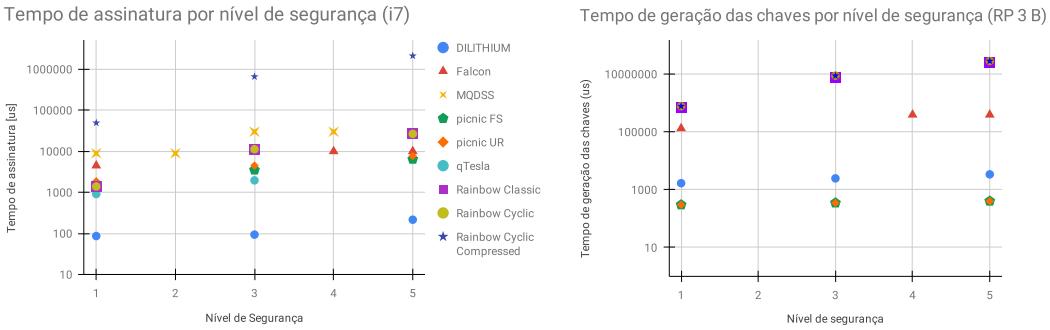


Figura 3. Picnic possui os menores tempos de geração de chave para todos os níveis, enquanto que o Rainbow possui os maiores.

De todos os parâmetros estudados, o tempo de geração de chaves é o menos relevante em nosso contexto dado o número de gerações muito inferior ao de assinaturas e verificações em um cenário IoT. Em nossos testes, observamos um ótimo desempenho no tempo de geração de chaves no Picnic, que se manteve abaixo dos 10 μs para testes no i7 e em torno dos 300 μs para raspberry em todos os níveis de segurança. Esse tempo reduzido muito possivelmente se dá ao uso de primitivas simétricas, facilitando sua geração. No outro espectro, vimos que o Rainbow possui os maiores tempos de geração, obtendo tempos entre 100 ms até a ordem de alguns segundos em seu último nível de segurança.

Além das medições de tempos de geração de chaves, outros parâmetros mais relevantes, como os tempos de assinatura e verificação, são mostrados nas Figuras 4 e 5. Ao comparar essas figuras, é possível observar que a maioria das propostas mantém consistentes seus tempos de assinatura e verificação, com a única exceção do Falcon, onde há uma assimetria. Essa assimetria nos tempos de assinatura e verificação pode ser útil em algumas situações onde a quantidade de assinaturas é muito inferior à de verificações como em aplicações cujo *payload* é de alto valor agregado. Para o nosso contexto, acreditamos que o desejável seja o oposto: menores tempos de assinatura com um possível *tradeoff* para a verificação. Isso considerando o cenário proposto na Seção 3, onde há múltiplos nós assinando dados e apenas um nó central que os verifica.

Vale destaque também para o algoritmo DILITHIUM que possui os menores tempos tanto para assinatura quanto para verificação. Ao mesmo tempo, vemos que os dois algoritmos multivariados presentes nos testes, são os mais lentos em ambas as categorias, o que pode ser crítico para determinadas aplicações em tempo real como por exemplo, em uma rede de sensores atrelado a um controlador de uma usina.

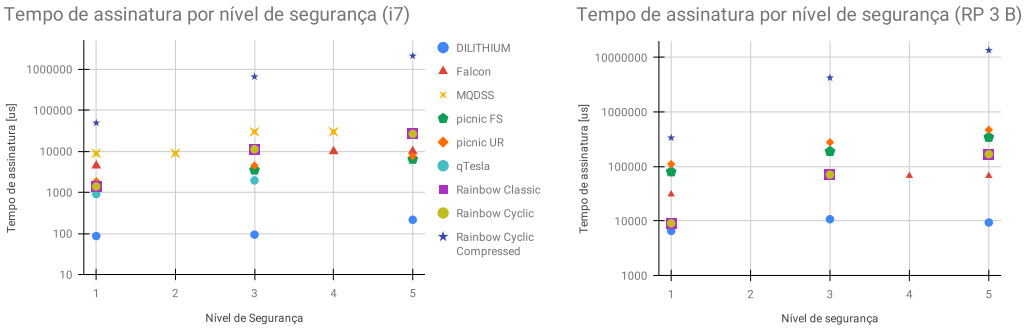


Figura 4. Menores tempos de assinatura em todos os níveis para DILITHIUM e maiores tempos para MQDSS, Falcon e Rainbow Cyclic Compressed

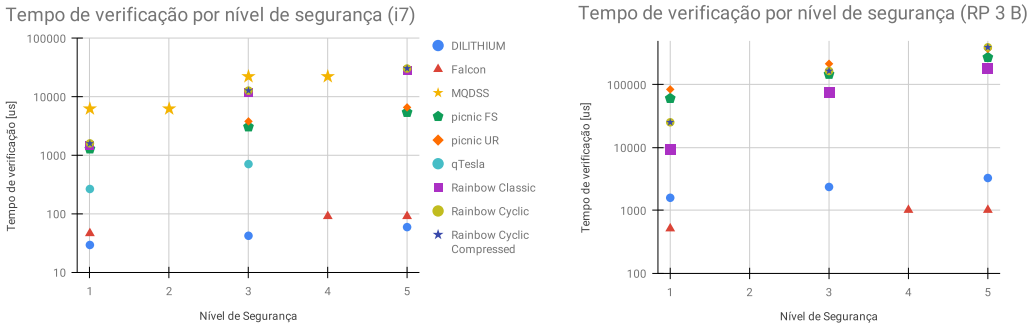


Figura 5. Menores tempos de verificação são dados pelo DILITHIUM e Falcon e maiores tempos pelo MQDSS, Rainbow e picnic (para o caso embarcado)

A partir das medições de tempos realizados no processador i7 e na rasberry pi, é possível traçar um gráfico da correlação entre eles para observar a relação entre a carga dos algoritmos e a diferença de tempo nas duas plataformas. Na Figura 6, temos a relação entre os tempos medidos em a escala *log-log*.

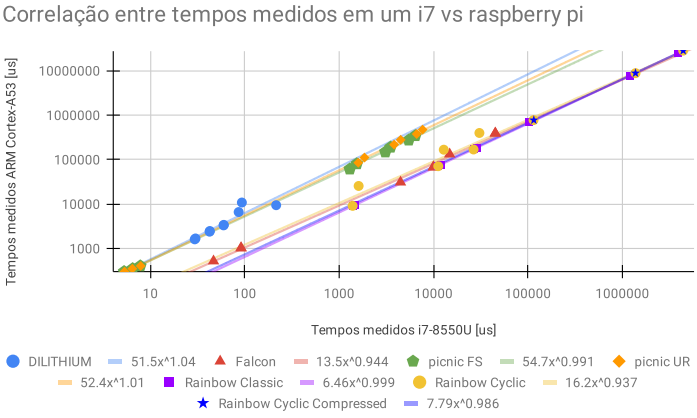


Figura 6. Regressão entre tempos medidos no i7 e na raspberry pi indicam uma característica quase linear, com expoentes variando em torno de um.

Na Figura 6 é possível observar a relação entre as medições nos dois ambientes, além das expressões para as curvas quase lineares que melhor aproximam os pontos de cada algoritmo. Os tempos mostrados nesse gráfico correspondem a todos os tempos

medidos, ou seja, os tempos de geração, assinatura e verificação dos algoritmos implementados em ambos os ambientes, sempre considerando o mesmo nível de segurança.

Este gráfico traz alguns detalhes interessantes sobre os algoritmos pesquisados. Podemos constatar que para os algoritmos Falcon e Rainbow há uma redução de aproximadamente dez vezes na velocidade de execução quando trocamos o i7 pelo ARM mas, ao mesmo tempo, temos uma redução de quase cem vezes para os algoritmos Dilithium e Picnic. Isso mostra claramente outras limitações da plataforma restrita que vão além de sua frequência de clock mais baixa. Essa distinção possivelmente está relacionada com uma maior demanda de memória RAM por parte desses algoritmos e poderá ser estudada mais a fundo em trabalhos futuros que auxiliarão no entendimento das limitações em ambientes restritos quando comparados a um hardware mais robusto.

Com relação às características de consumo de espaço nesses algoritmos, os testes no libOQS fornecem os valores tamanhos máximos correspondentes as especificações das propostas submetidas ao NIST. Na Figura 7, temos os tamanhos de assinaturas, chaves públicas e chaves privadas em relação aos níveis de segurança.

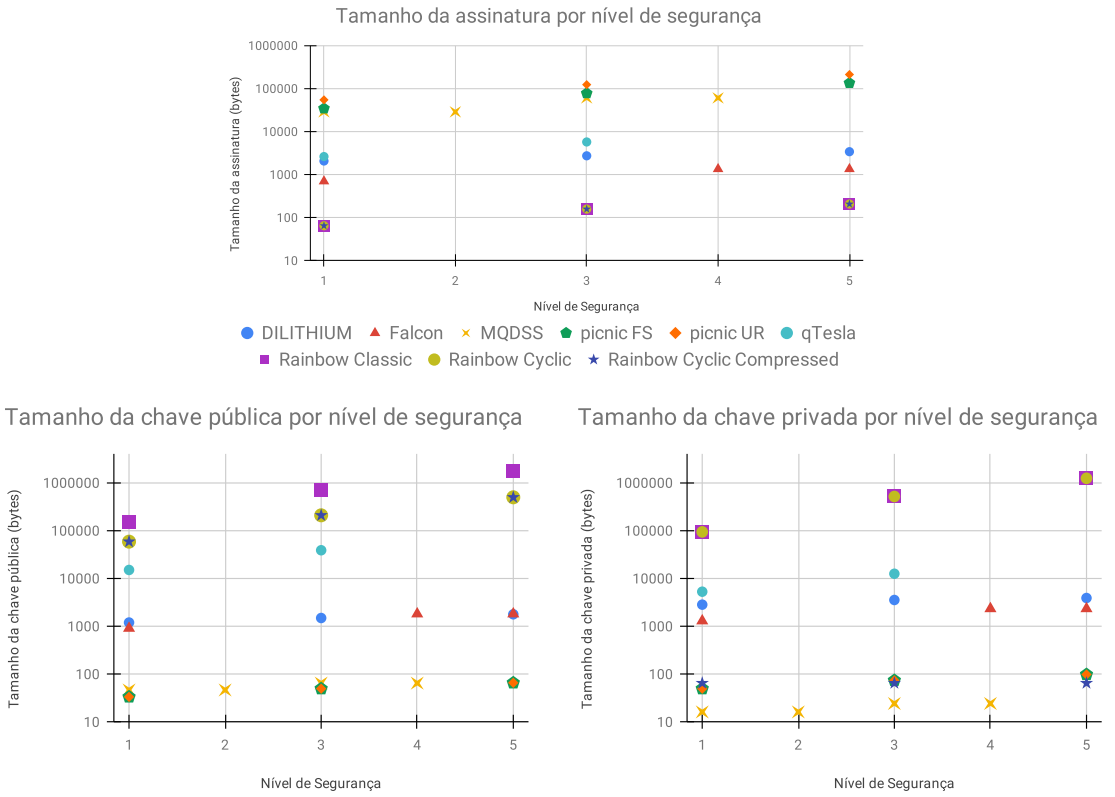


Figura 7. Menores assinaturas no algoritmo Rainbow mas com as maiores chaves, exatamente o oposto do que ocorre com MQDSS.

Com base nos tamanhos apresentados pela Figura 7, é possível notar alguns padrões em relação às escolhas de projeto das propostas. Quando observamos os tamanhos das **chaves** do Rainbow, vemos que é o pior caso em ambas as categorias (com exceção da variação *Rainbow Cyclic Compressed* que possui uma das menores chaves privadas). Em contrapartida ele possui a menor assinatura de todos, para todos os níveis de segurança. Já no Picnic, e MQDSS vemos o oposto: as menores chaves mas as maiores

assinaturas entre os algoritmos analisados.

Os três algoritmos de reticulados (DILITHIUM, Falcon e qTesla) possuem um balanço entre os três tamanhos, o que pode ser desejável em determinados cenários, como situações onde a capacidade de armazenamento entre os nós é igualmente limitada, e.g redes *peer-to-peer* sem um nó central.

Duas últimas análises relevantes que podemos realizar são: a comparação entre as somas de tamanhos de chaves públicas e assinaturas e a comparação entre o tamanho de assinaturas e o tempo necessário para assinar.

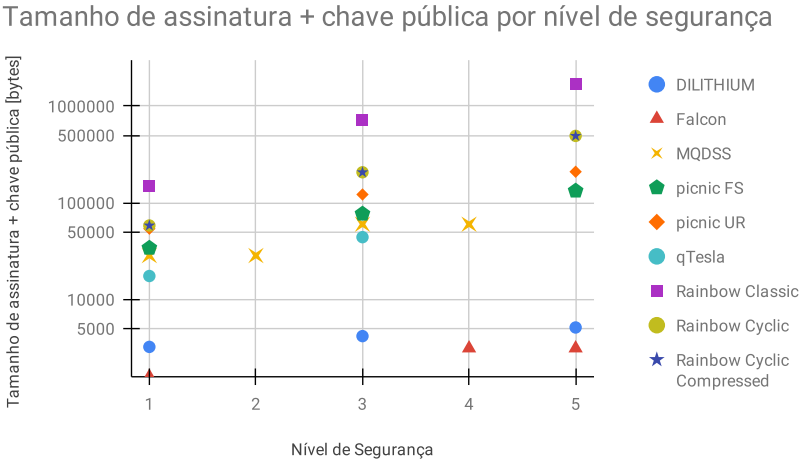


Figura 8. O Rainbow é custoso quando considerado o tamanho da chave pública junto a sua assinatura. Já o DILITHIUM e Falcon são os que possuem os menores tamanhos nessa métrica.

Pela Figura 8 é possível notar que, apesar das pequenas assinaturas do Rainbow, quando elas são somadas à sua chave pública ele se torna o pior nesta métrica. Contudo, como comentado na Seção 3, em nosso cenário de estudo há um nó agregador central que já estaria configurado previamente com os certificados dos outros nós, permitindo portanto, que o Rainbow ainda seja viável neste cenário. Em outros cenários IoT onde não exista um nó centralizador, ainda é possível que o envio do certificado não seja um fator tão prejudicial, visto que o envio do certificado seria necessário apenas no início do estabelecimento da primeira conexão em situações com redes estáticas, como normalmente são as redes de sensoreamento locais.

A última análise a ser feita é baseada na Figura 9, onde é possível estudar eventuais fontes de gargalos em sistemas distribuídos em rede como arquiteturas IoT, uma vez que, nestes cenários, estamos preocupados com *overheads* tanto em tamanho de pacote, quanto em taxas de transmissão.

Podemos observar a partir da Figura 9 que há dois grandes destaques no cenário de IoT. A depender do fator mais crítico para cada aplicação, podemos optar pelo uso do DILITHIUM em cenários onde o tempo de assinatura deve ser baixo, mas o tamanho das assinaturas é aceitável, ou optar pelo uso do Rainbow onde o tempo não é crítico, mas é necessário uma assinatura menor. Dentre os piores desempenhos, estão o desempenho do Picnic e do MQDSS, mostrando um forte contraste com seus destaques anteriores por terem chaves pública e privada pequenas.

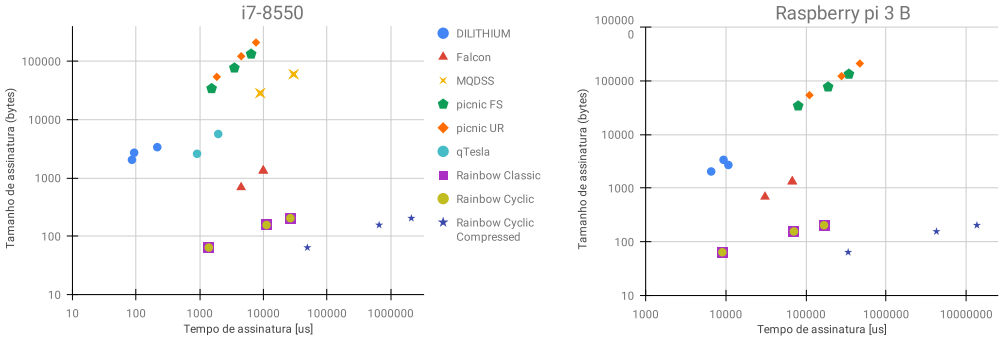


Figura 9. Em questão de gargalo para transmissão em redes, o DILITHIUM possui os melhores compromissos entre tamanho e tempo de assinatura. O Rainbow, por outro lado, possui as menores assinaturas, mas gastando um tempo considerável para criá-las.

7. Observações finais

Pouco antes da submissão deste artigo, no dia (22/07/2020), foram publicados os candidatos escolhidos pelo NIST que avançaram para a terceira rodada do processo. Os escolhidos pelo NIST foram classificados em duas categorias: "finalistas" e "alternativos". A categoria "finalistas" corresponde aos que o NIST considera mais promissores e que possivelmente serão escolhidos como um dos padrões. Os finalistas estão: CRYSTALS-DILITHIUM, Falcon e Rainbow. Já os algoritmos "alternativos" são os algoritmos que ainda poderão ser padronizados, mas possivelmente não na terceira rodada. Nesta categoria, para assinatura digitais, constam: GeMSS, Picnic e SPHINCS+.

Nossa análise de desempenho também apontou o DILITHIUM e o Rainbow como opções promissoras para o cenário em estudo, especialmente dependendo da criticidade das aplicações. Dos algoritmos de assinaturas que ficaram fora da terceira rodada, temos: LUOV, MQDSS, qTESLA, já que em todos foram achadas falhas de segurança críticas.

8. Conclusão e Trabalhos futuros

Neste artigo apresentamos algumas das ideias por trás das novas premissas dos algoritmos criptográficos pós-quânticos para a realização de assinaturas digitais, bem como alguns testes de desempenho destes algoritmos em uma plataforma desktop (Intel i7) e em uma platafoma restrita (Raspberry pi 3).

Com o uso da biblioteca libOQS, comparamos o desempenho e os *tradeoffs* dos algoritmos de assinaturas implementados, com grande destaque para os tamanhos de chaves e tempos de assinaturas em reticulados, em especial do DILITHIUM. Para o cenário IoT, onde o envio de cadeias de certificados não costuma ser o padrão, vale destacar os tamanhos de assinaturas do Rainbow que, apesar de tempos e chaves maiores, pode possuir um espaço para aplicações IoT em certos cenários.

Em trabalhos futuros, pretendemos resolver algumas lacunas encontradas durante a realização deste trabalho, tais como o aprofundamento de algumas das propostas como Sphincs+ e Picnic, e a realização de testes e implementações em ambientes IoT completos para medição de performance em sistemas com condições mais próximas às reais, bem como com a presença de atrasos na rede e limitação de banda. Também serão buscadas formas de otimizar os códigos para ambientes IoT a fim de amenizar a queda de desempenho quando comparado a um ambiente desktop.

9. Agradecimentos

Este projeto de pesquisa foi realizado dentro do Programa Institucional de Bolsas de Iniciação Científica - PIBIC realizado pela Universidade Estadual de Campinas - UNICAMP e financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, processo número 124609/2019-8.

Referências

Ajtai, M. (1996). Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108.

An, H., Choi, R., Lee, J., and Kim, K. (2018). Performance evaluation of liboqs in open quantum safe project (part i). In *2018 Symposium on Cryptography and Information Security (SCIS 2018)*. IEICE Technical Committee on Information Security.

Babai, L. (1986). On lovász'lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13.

Bernstein, D. J. and Lange, T. (2019). ebacs: Ecrypt benchmarking of cryptographic systems. URL: `bench.cr.yp.to/`, Acessado em 13/09/2020.

Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., and Zaverucha, G. (2017). Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *Proceedings of the 2017 acm sigsac conference on computer and communications security*, pages 1825–1842.

Goldreich, O., Goldwasser, S., and Halevi, S. (1997). Public-key cryptosystems from lattice reduction problems. In *Annual International Cryptology Conference*, pages 112–131. Springer.

Jao, D. and De Feo, L. (2011). Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer.

Lamport, L. (1979). Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International.

Merkle, R. C. (1989). A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer.

NISTPQC (2016). Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. *https://csrc.nist.gov*.

Patarin, J., Kipnis, A., and Goubin, L. (1999). Unbalanced oil and vinegar signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 206–222. Springer.

Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40.

Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332.

Stebila, D. and Mosca, M. (2016). Post-quantum key exchange for the internet and the open quantum safe project. In *International Conference on Selected Areas in Cryptography*, pages 14–37. Springer.