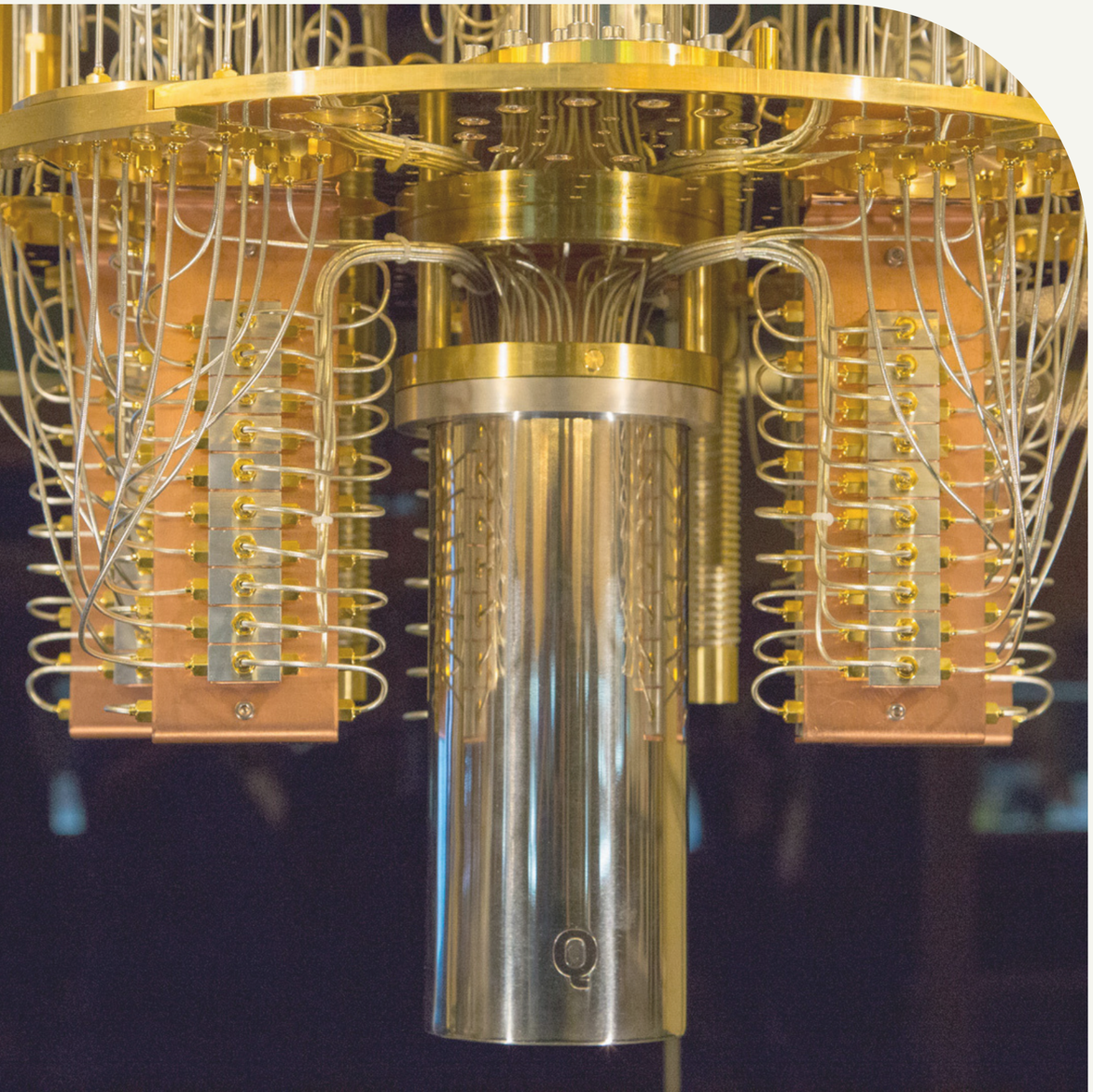
Impacto da otimização de funções hash no desempenho do algoritmo de assinaturas digitais pós-quântica CRYSTALS-Dilithium

Rodrigo Duarte de Meneses
Marco Aurélio Amaral Henriques

Introdução

— PARTE 1



O que é a criptografia pós-quântica?

- Computação quântica põe em risco a segurança dos esquemas de criptografia clássica;
- **Criptografia pós-quântica:** esquemas criptográficos resistentes aos ataques de um computador quântico;
- Processo de padronização do NIST para algoritmos pós-quânticos.

Assinaturas digitais

- Buscam promover a verificação da autenticidade de mensagens e/ou documentos digitais.

- Consistem em 3 etapas:

1. Geração de chaves (**Gen**);
2. Procedimento de assinatura (**Sign**);
3. Verificação (**Verify**)

CRYSTALS-Dilithium

(CRYPTOGRAPHIC SUITE FOR ALGEBRAIC LATTICES)

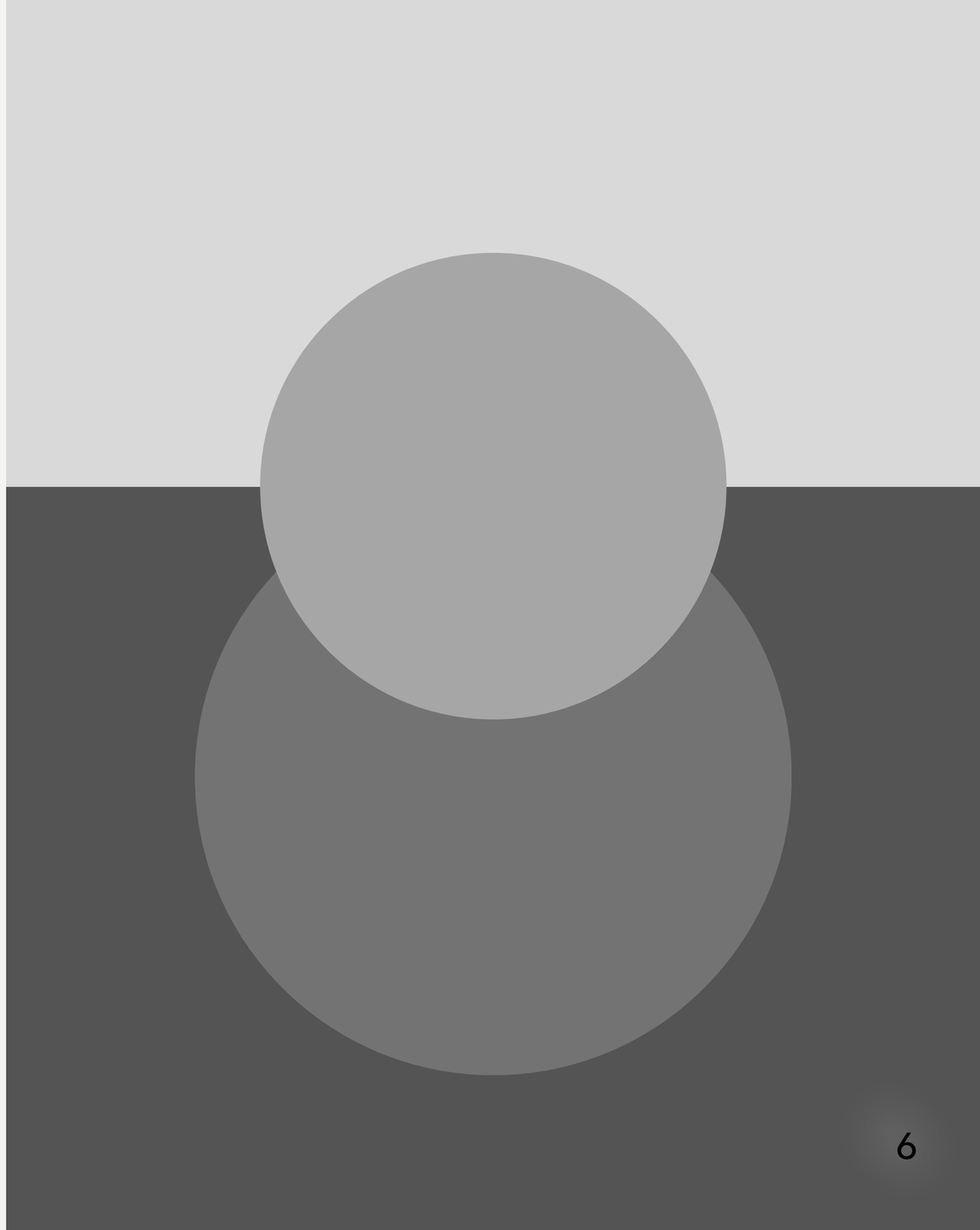
Um dos menores tamanhos de chave e assinaturas comparado aos outros finalistas do processo de padronização do NIST.

Segurança dependente do problema MLWE e independente de implementação de aritmética de ponto flutuante.

Arquitetura modular, a variação da segurança pode ser feita alterando as dimensões das matrizes e vetores de polinômios.

Mecanismo de funcionamento

— PARTE 2

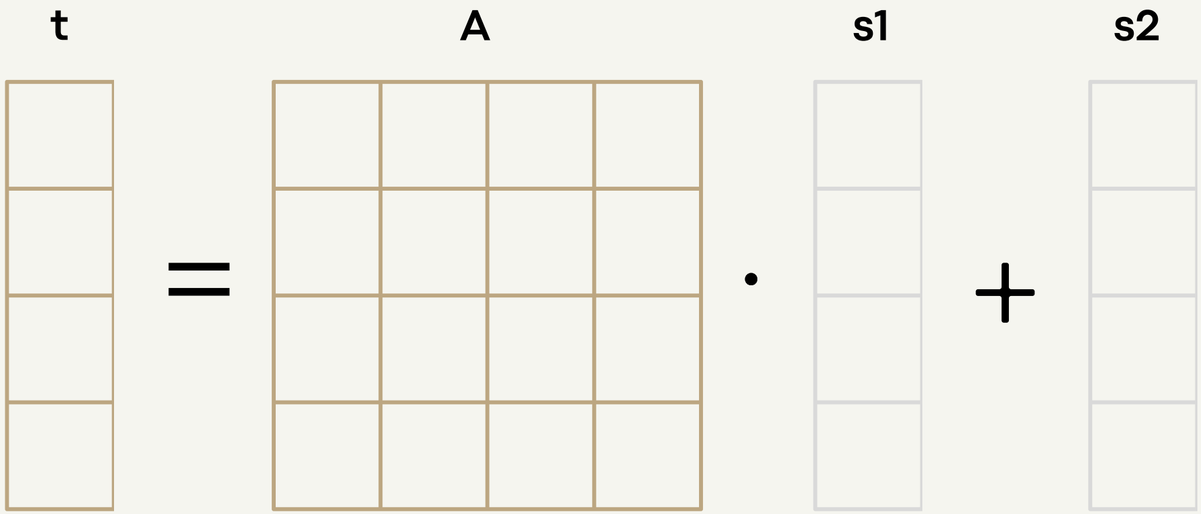


- **Problema baseado em reticulados**

Chave pública = (**A, t**)

Chave privada = (**s1, s2**)

- Vetor de polinômios **t**;
- Matriz de polinômios **A**;
- Vetores secretos de polinômios, **s1** e **s2**.

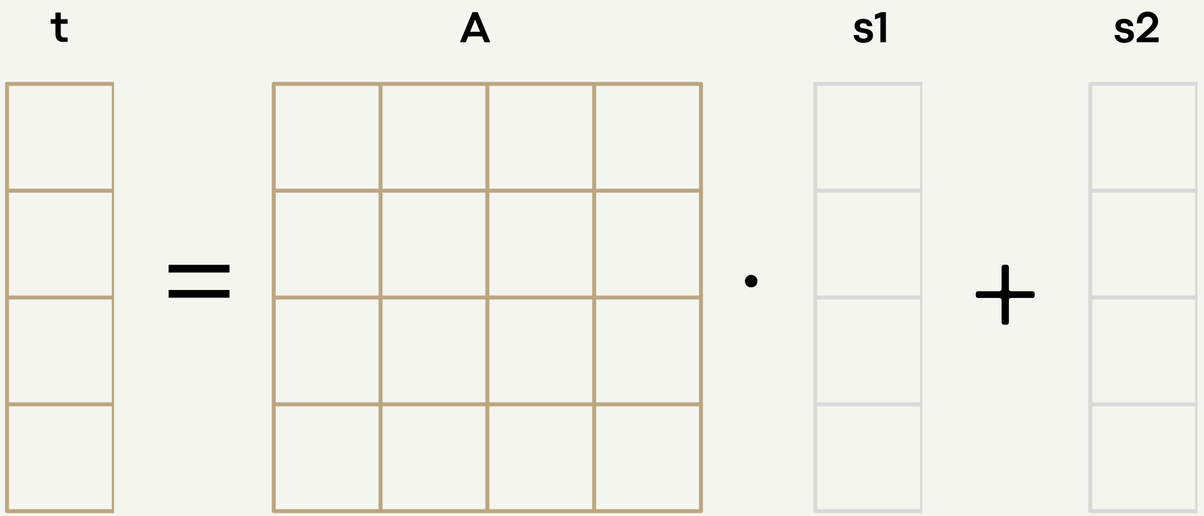


- **Problema baseado em reticulados**

- Vetor de polinômios **t**;
- Matriz de polinômios **A**;
- Vetores secretos de polinômios, **s1** e **s2**.

Chave pública = (**A, t**)

Chave privada = (**s1, s2**)



- A dificuldade consiste em distinguir um vetor aleatório **t** de um vetor **t** = **A** . **s1** + **s2**.
- Cada entrada dos vetores ou matriz é um polinômio de 256 coeficientes.
- Esse problema é denominado MLWE (Module Learning With Errors).

Problema

- O Dilithium exige o armazenamento de vetores e matrizes grandes.

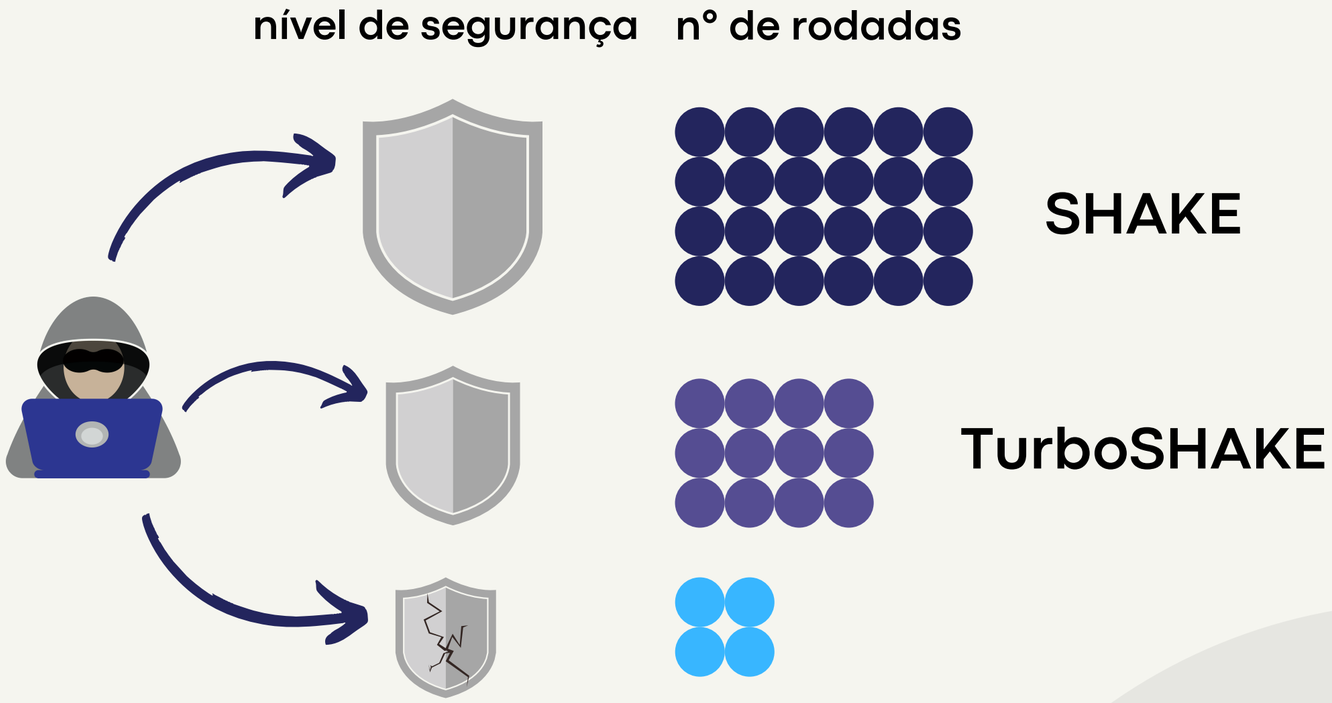
Problema

- O Dilithium exige o armazenamento de vetores e matrizes grandes.

Solução

- Geramos os parâmetros sob demanda a partir de uma seed ζ.
- Isso é feito a partir de funções hash de saída extensível (XOFs).
- O Dilithium utiliza funções padronizadas pelo NIST, SHAKE128 e SHAKE256.
- A utilização dessa funções corresponde a maior parte dos custos computacionais associados ao uso do Dilithium.

SHAKE vs TurboSHAKE



- SHAKE utiliza 24 rodadas para obter sua segurança.
- Redução do número de rodadas até um certo limite não compromete a segurança do algoritmo para nenhum dos ataques conhecidos.
- **TurboSHAKE**: versão de rodadas reduzidas do SHAKE (12 rodadas).

Métodos e resultados

— PARTE 3

Materiais utilizados

- Utilizou-se um *laptop* com processador Intel Core i5-8265U CPU 1.6 GHz, SO Ubuntu 20.04.6 LTS e memória RAM de 8GB.

Materiais utilizados

- Utilizou-se um *laptop* com processador Intel Core i5-8265U CPU 1.6 GHz, SO Ubuntu 20.04.6 LTS e memória RAM de 8GB.

Método de análise

ABORDAGEM EXPERIMENTAL UTILIZADA

Medimos a média e mediana de ciclos de processador para cada uma das funções principais para as versões v.0 (SHAKE) e v.1 (TurboSHAKE).

A utilização de *rejection sampling* implica na presença de alguns *outliers* para a função **Sign.**

Materiais utilizados

- Utilizou-se um *laptop* com processador Intel Core i5-8265U CPU 1.6 GHz, SO Ubuntu 20.04.6 LTS e memória RAM de 8GB.

Procedimento experimental

- Medições para 1500 iterações do algoritmo para cada nível de segurança (2, 3 e 5) descritos na documentação do Dilithium.

- Em cada iteração é feita a geração de chaves, assinatura e verificação para uma mensagem aleatória de 59 bytes.

Método de análise

ABORDAGEM EXPERIMENTAL UTILIZADA

Medimos a média e mediana de ciclos de processador para cada uma das funções principais para as versões v.0 (SHAKE) e v.1 (TurboSHAKE).

A utilização de *rejection sampling* implica na presença de alguns *outliers* para a função **Sign.**

Utilização de memória no CRYSTALS-Dilithium

- Comparando as versões v.0 (SHAKE) e v.1 (TurboSHAKE), foi possível atestar que não houve aumento na utilização de memória RAM do algoritmo em nenhum dos níveis de segurança.

Tabela 1. Uso máximo de RAM (bytes) para níveis de segurança 2, 3 e 5

Pico de RAM para CRYSTALS-Dilithium v.0 / v.1			
Nível de segurança	KeyGen	Sign	Verify
Dilithium 2	42.488 B	59.648 B	43.936 B
Dilithium 3	67.128 B	90.272 B	68.344 B
Dilithium 5	105.496 B	134.960 B	106.448 B

Utilização de memória no CRYSTALS-Dilithium

- Comparando as versões v.0 (SHAKE) e v.1 (TurboSHAKE), foi possível atestar que não houve aumento na utilização de memória RAM do algoritmo em nenhum dos níveis de segurança.

Tabela 1. Uso máximo de RAM (bytes) para níveis de segurança 2, 3 e 5

Pico de RAM para CRYSTALS-Dilithium v.0 / v.1			
Nível de segurança	KeyGen	Sign	Verify
Dilithium 2	42.488 B	59.648 B	43.936 B
Dilithium 3	67.128 B	90.272 B	68.344 B
Dilithium 5	105.496 B	134.960 B	106.448 B

Otimizações de processamento - Dilithium

Tabela 2. Média de ciclos de CPU para cada função do Dilithium

Média de ciclos de CPU para o CRYSTALS-Dilithium			
Nível de segurança	Função	v.0	v.1 (variação %)
Dilithium 2	KeyGen	418.522	325.304 (-22.3%)
	Sign	1.844.839	1.674.893 (-9.2%)
	Verify	447.904	362.306 (-19.1%)
Dilithium 3	KeyGen		
	Sign		
	Verify		
Dilithium 5	KeyGen		
	Sign		
	Verify		

Otimizações de processamento - Dilithium

Tabela 2. Média de ciclos de CPU para cada função do Dilithium

Média de ciclos de CPU para o CRYSTALS-Dilithium			
Nível de segurança	Função	v.0	v.1 (variação %)
Dilithium 2	KeyGen	418.522	325.304 (-22.3%)
	Sign	1.844.839	1.674.893 (-9.2%)
	Verify	447.904	362.306 (-19.1%)
Dilithium 3	KeyGen		
	Sign		
	Verify		
Dilithium 5	KeyGen		
	Sign		
	Verify		

Otimizações de processamento - Dilithium

Tabela 2. Média de ciclos de CPU para cada função do Dilithium

Média de ciclos de CPU para o CRYSTALS-Dilithium			
Nível de segurança	Função	v.0	v.1 (variação %)
Dilithium 2	KeyGen	418.522	325.304 (-22.3%)
	Sign	1.844.839	1.674.893 (-9.2%)
	Verify	447.904	362.306 (-19.1%)
Dilithium 3	KeyGen	740.820	566.839 (-23.4%)
	Sign	2.781.223	2.525.570 (-9.2%)
	Verify	705.726	556.641 (-21.1%)
Dilithium 5	KeyGen	1.144.005	842.012 (-26.4%)
	Sign	3.523.184	3.255.138 (-7.6%)
	Verify	1.173.812	899.562 (-23.3%)

Otimizações de processamento - Dilithium

Tabela 2. Média de ciclos de CPU para cada função do Dilithium

Média de ciclos de CPU para o CRYSTALS-Dilithium			
Nível de segurança	Função	v.0	v.1 (variação %)
Dilithium 2	KeyGen	418.522	325.304 (-22.3%)
	Sign	1.844.839	1.674.893 (-9.2%)
	Verify	447.904	362.306 (-19.1%)
Dilithium 3	KeyGen	740.820	566.839 (-23.4%)
	Sign	2.781.223	2.525.570 (-9.2%)
	Verify	705.726	556.641 (-21.1%)
Dilithium 5	KeyGen	1.144.005	842.012 (-26.4%)
	Sign	3.523.184	3.255.138 (-7.6%)
	Verify	1.173.812	899.562 (-23.3%)

Otimizações de processamento - Dilithium

Tabela 3. Mediana de ciclos de CPU para cada função do Dilithium

Mediana de ciclos de CPU para o CRYSTALS-Dilithium			
Nível de segurança	Função	v.0	v.1 (variação %)
Dilithium 2	KeyGen	411.942	321.624 (-21.9%)
	Sign	1.490.671	1.279.631 (-14.2%)
	Verify	446.843	361.137 (-19.2%)
Dilithium 3	KeyGen		
	Sign		
	Verify		
Dilithium 5	KeyGen		
	Sign		
	Verify		

Otimizações de processamento - Dilithium

Tabela 3. Mediana de ciclos de CPU para cada função do Dilithium

Mediana de ciclos de CPU para o CRYSTALS-Dilithium			
Nível de segurança	Função	v.0	v.1 (variação %)
Dilithium 2	KeyGen	411.942	321.624 (-21.9%)
	Sign	1.490.671	1.279.631 (-14.2%)
	Verify	446.843	361.137 (-19.2%)
Dilithium 3	KeyGen		
	Sign		
	Verify		
Dilithium 5	KeyGen		
	Sign		
	Verify		

Otimizações de processamento - Dilithium

Tabela 3. Mediana de ciclos de CPU para cada função do Dilithium

Mediana de ciclos de CPU para o CRYSTALS-Dilithium			
Nível de segurança	Função	v.0	v.1 (variação %)
Dilithium 2	KeyGen	411.942	321.624 (-21.9%)
	Sign	1.490.671	1.279.631 (-14.2%)
	Verify	446.843	361.137 (-19.2%)
Dilithium 3	KeyGen	724.807	562.345 (-22.4%)
	Sign	2.230.749	2.080.019 (-6.8%)
	Verify	704.122	553.911 (-21.3%)
Dilithium 5	KeyGen	1.119.261	836.480 (-25.3%)
	Sign	3.026.118	2.665.533 (-11.9%)
	Verify	1.168.728	897.902 (-23.2%)

Conclusões

& PERSPECTIVAS DE TRABALHOS FUTUROS

- Redução significativa no número de ciclos de CPU para todos os níveis de segurança do Dilithium.
- Sem comprometer a segurança e integridade do algoritmo.

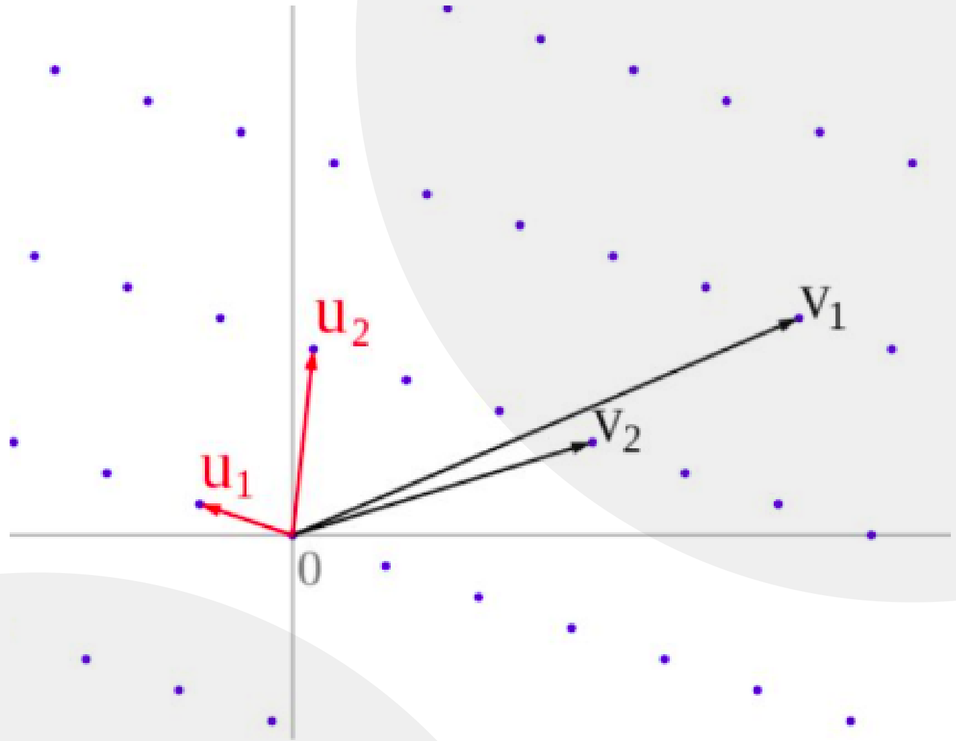
- Buscar possíveis compromissos entre velocidade de processamento e espaço de memória utilizado.
- Avaliar a implementação do algoritmo para dispositivos restritos.
- Aplicar a otimização obtida neste trabalho para outros algoritmos pós-quânticos que façam uso do SHAKE.

OBRIGADO!

Rodrigo de Meneses
FEEC – Unicamp
rodrigoduartem@protonmail.com

O que são reticulados?

- Os reticulados são estruturas algébricas que consistem em um conjunto S de pontos em um espaço n-dimensional.
- Os pontos são gerados a partir de uma base de vetores contida em S.
- A criptografia baseada em reticulados utiliza de problemas matemáticos baseados nessas estruturas.



Pseudocódigo simplificado

KeyGen()

$$\mathbf{A} \leftarrow R^{n \times m}$$
$$\mathbf{s_1} \leftarrow R^m, \ \mathbf{s_2} \leftarrow R^n$$
$$\mathbf{t} = \mathbf{A}\mathbf{s_1} + \mathbf{s_2}$$
$$p_k = (\mathbf{A}, \mathbf{t}), \ s_k = (\mathbf{s_1}, \mathbf{s_2})$$

Sign(p_k, s_k, μ)

$$\mathbf{y} \leftarrow R^m$$
$$\mathbf{w} = \mathbf{A}\mathbf{y}$$
$$c = \mathrm{H}\left[\mathrm{High}(\mathbf{w}), \mu\right]$$
$$\mathbf{z} = c\mathbf{s_1} + \mathbf{y}$$
$$\mathrm{RejectionSample}(p_k, s_k, \mathbf{z})$$
$$\sigma = (\mathbf{z}, c)$$

Verify(μ, σ, p_k)

$$\tilde{c} = \mathrm{H}\left[\mathrm{High}(\mathbf{A}\mathbf{z} - c\mathbf{t}), \mu\right]$$
$$\textbf{If} \quad c = \tilde{c} \quad \textbf{and} \quad |z| \,\mathrm{is\,small}, \quad \textbf{accept}$$