Otimização de esquemas de criptografia baseados em algoritmos pós-quânticos

Palavras-Chave: criptografia, algoritmos pós-quânticos, segurança da informação, IoT

Luiz Fernando Campos Ferro [UNICAMP]

Prof. Dr. Marco Aurélio Amaral Henriques (orientador) [UNICAMP]

1. Introdução

O desenvolvimento de computadores quânticos se aproxima cada vez mais da realidade e, com isso, os algoritmos de chave pública empregados na indústria estão ameaçados. Nos últimos anos, o National Institute of Standards and Technology (NIST) dos Estados Unidos, iniciou uma pesquisa para avaliar propostas de algoritmos resistentes ao ataque de computadores quânticos. Todas elas, no entanto, demandam alto poder computacional, seja em velocidade de processamento ou em espaço necessário para armazenamento de chaves. Há um agravamento no problema quando se introduz esses algoritmos em ambientes computacionais restritos, como sistemas embarcados e dispositivos voltados para IoT (Internet of Things) em geral. Muitas vezes, essas plataformas não oferecem os recursos necessários para a implementação de um algoritmo pós-quântico.

Sabendo da crescente inserção desses dispositivos de pequeno porte no dia a dia da sociedade, mostra-se necessário propor otimizações nos algoritmos de criptografia pós-quânticos de forma a viabilizar suas implementações nesses cenários restritos para que sua segurança seja mantida, mesmo após o surgimento de computadores quânticos práticos. Desse modo, este trabalho expõe e analisa diferentes técnicas de otimização para CRYSTALS-Kyber, um dos algoritmos de criptografia pós-quânticos finalistas da terceira rodada de avaliações coordenada pelo NIST, buscando reduzir seu consumo de memória e tempo de processamento de modo a facilitar sua implementação em um ambiente restrito.

2. Propriedades do CRYSTALS-Kyber

Kyber é um mecanismo de encapsulamento de chaves (key-encapsulation mechanism ou KEM) da família CRYSTALS (Cryptographic Suite for Algebraic Lattices) e é um algoritmo de criptografia baseada em reticulados. Encapsular uma chave pode ser entendido como o processo de cifrar esta chave para que trafegue protegida por canais inseguros. Os reticulados são conjuntos de pontos igualmente espaçados e dispostos em um espaço de n dimensões, sendo esses pontos gerados por meio da combinação linear de vetores linearmente independentes que constituem a base do respectivo reticulado. Durante a terceira rodada do NIST, dos quatro finalistas na área de pesquisa de criptografia pós-quântica, três se baseiam em reticulados. Por essa razão, reticulados se mostram mais promissores que a outra opção baseada em códigos como principal escolha para padronização [2].

Além disso, o Kyber possui uma propriedade segurança em termos de indistinguibilidade de texto cifrado (IND), o que garante que qualquer adversário não seja capaz de diferenciar pares de

mensagens encriptadas com base na mensagem original. Em outras palavras, o adversário não deve ter chances de sucesso no desencapsulamento significativamente maiores que a própria adivinhação aleatória. Caso contrário, há uma vantagem do atacante em identificar o texto e o algoritmo perde sua credibilidade. Essa propriedade pode ser representada por três tipos de segurança, onde cada uma confere ao adversário cada vez mais recursos de ataque, sendo elas, em nível crescente de segurança, denominadas CPA (*Chosen-Plaintext Attack*), CCA (*Chosen-Ciphertext Attack*) e CCA2 (*Adaptive Chosen-Ciphertext Attack*). No nível de segurança CPA (ataque de texto em claro escolhido), o adversário consegue encapsular diferentes mensagens da forma que desejar. Já no maior nível de segurança, CCA2 (ataque de texto cifrado escolhido adaptativo), o atacante recebe uma mensagem encapsulada e seu objetivo é obter o respectivo texto em claro. Ele tem acesso a um oráculo de desencapsulamento, para o qual envia mensagens arbitrárias encapsuladas e recebe o respectivo texto em claro, sem poder, no entanto, enviar ao oráculo a mensagem com o desafio que precisa ser desencapsulado. Esse tipo de consulta pode permitir que o adversário aumente suas chances de sucesso no ataque. O Kyber é um algoritmo com segurança do tipo IND-CCA2 e isso garante que ele também é seguro contra os demais tipos de ataques. O aumento do nível de segurança exige que o algoritmo faça mais cálculos com primitivas criptográficas, o que aumenta os requisitos de memória e processamento.

A construção do algoritmo de criptografia baseada em reticulados permite que o desenvolvedor crie combinações de parâmetros que configuram níveis de segurança. Dessa forma, a versão 3.01 de implementação do Kyber submetida à terceira rodada do NIST conta com três diferentes níveis, em ordem crescente de segurança: Kyber-512, Kyber-768 e Kyber-1024, sendo o número de dimensões empregado na base do reticulado a principal diferença entre essas versões. Essas variações permitem criar estimativas preliminares de segurança em cada nível, utilizando como referência a segurança de algoritmos de criptografia já conhecidos e empregados, como AES e SHA. Ainda existem outras implementações do Kyber buscando diferentes aplicações, mas que não conferem necessariamente mais segurança ao código. Uma delas, intitulada "Kyber-90s", abandona o padrão utilizado na versão de referência e utiliza funções alternativas para instanciar as primitivas do código, especificamente substituindo funções mais pesadas como SHAKE e SHA3 por outras mais leves (ou com implementações disponíveis em hardware) como AES e SHA2.

3. Condições de testes e resultados preliminares

É importante considerar que todas as versões do Kyber submetidas ao NIST realizam o encapsulamento de uma mensagem de tamanho fixado em 32 bytes, um padrão de entrada mantido em todos os testes. Dessa forma, buscando avaliar o desempenho da versão de referência do algoritmo, utilizou-se o software Valgrind para medição do consumo de memória total (stack e heap) e a ferramenta Perf para os ciclos de execução dos algoritmos em uma máquina Linux, processador Intel Core i5-4570 (Haswell) de 3.20GHz. Mediu-se o desempenho da versão de referência mais leve, Kyber512, para consumo de memória em KiB e números de ciclos de execução, separando os processos em geração de chaves, encapsulamento e desencapsulamento, conforme mostra a tabela 1.

Tabela 1. Consumo de memória e ciclos de CPU Intel i5 (versão de referência Kyber512)

Processo	Geração de chaves	Encapsulamento	Desencapsulamento
Consumo em KiB	19,7	21,6	23,9
Ciclos de CPU	65.310.796	201.642.131	214.875.107

4. Propostas de otimizações para o Kyber

Buscando reduzir os requisitos exigidos para o funcionamento do algoritmo, esta seção apresenta e avalia a aplicação de diferentes técnicas de otimizações para o CRYSTALS-Kyber com o intuito de facilitar sua implementação em um ambiente restrito. Para isso, é possível seguir duas abordagens: técnicas de otimização para consumo de memória e/ou redução no tempo de processamento. Um fator importante da construção do Kyber é o uso da biblioteca OpenSSL, que implementa e disponibiliza diversas rotinas criptográficas dos protocolos SSL e TLS. Apesar da grande variedade de funções fornecidas pela biblioteca, ela é utilizada no Kyber somente para uma função de geração de números aleatórios (RNG) que criam as sementes de geração das matrizes. Mesmo se tratando de uma funcionalidade extremamente importante para conferir segurança ao algoritmo e que precisa ter alta credibilidade, seu uso não é obrigatório e pode ser substituído por outros recursos de geração de números aleatórios disponíveis em cada dispositivo. Além disso, a biblioteca consome um grande espaço de memória. Dessa forma, avaliou-se o impacto na remoção da OpenSSL, utilizando (por enquanto) a função srand em C como método de geração de números aleatórios. Isso pode trazer impactos de segurança que precisarão ser avaliados para cada plataforma onde o algoritmo for usado.

Durante sua execução, Kyber também faz uso de leitura e escrita em arquivos, para acessar diferentes sementes definidas pelo NIST (para efeitos de teste) e registrar chave pública, privada e texto encapsulado. Eliminar a necessidade de acesso a arquivos é uma estratégia interessante para a aplicação do algoritmo em um ambiente restrito. Ao invés de utilizar arquivos, o algoritmo lida com chaves e textos somente em memória. O impacto da remoção de OpenSSL e acesso a arquivos em relação à versão de referência do NIST (chamada de "Ref.") é mostrado na Tabela 2, sendo essa nova versão intitulada "nRef". A tabela expõe também os dados comparativos para geração de chaves (Ger), encapsulamento (Enc) e desencapsulamento (Des) para a versão Kyber512. Visto o impacto que a retirada desses recursos causa no algoritmo, a versão que será utilizada como referência de agora em diante será Kyber-nRef.

Tabela 2. Impactos da remoção de OpenSSL e acesso a arquivos de Kyber512 (Intel i5)

Módulo	Ciclos de CPU	Variação em ciclos (%)	Consumo de memória (KiB)	Variação em memória (%)
Ger: Ref	65.310.796	-	19,7	-
Ger: nRef	13.055.197	-80,0%	10,5	-46,7%
Enc: Ref	201.642.131	-	21,6	-
Enc: nRef	16.376.449	-91,9%	12,4	-42,6%
Des: Ref	214.875.107	-	23,9	-
Des: nRef	20.423.529	-90,5%	17,1	-28,5%

A velocidade do Kyber é majoritariamente determinada pelo desempenho das primitivas criptográficas empregadas. O design do algoritmo em sua versão de referência foi implementado utilizando uma família de primitivas baseada no hash Keccak [4] e padronizada como SHA3. Esta função é extremamente rápida quando implementada em hardware dedicado ou programável; porém sua implementação em software pode ser lenta quando comparada a outras funções hash conhecidas [5]. Apesar da similaridade entre as estruturas da função SHA3 e SHAKE, a primeira foi

desenvolvida com parâmetros de segurança considerados excessivos e desnecessários atualmente [6]. Os dados expostos nas Tabelas 4 e 5 comparam o consumo máximo de memória e ciclos de CPU entre a versão adaptada de referência nRef e a versão de segurança do tipo CPA do Kyber-512.

Tabela 4. Variação de consumo de memória (KiB) do Kyber-512 nas versões nRef e CPA (Intel i5)

Módulo	Ger (Var.%)	Enc (Var.%)	Des (Var.%)
Kyber512-nRef	10,5	12,4	17,1
Kyber512-CPA	9,6 (-8,6)	12,3 (-0,8)	8,6 (-49,7)

Tabela 5. Variação de ciclos de CPU do Kyber-512 nas versões nRef e CPA (Intel i5)

Módulo	Ger (Var.%)	Enc (Var.%)	Des (Var.%)
Kyber512-nRef	13,055,197	16,376,449	20,423,529
Kyber512-CPA	11.381.727 (-12,8)	14.100.500 (-13,9)	6.160.822 (-69,8)

A versão que considera somente o esquema de segurança IND-CPA da versão de referência do Kyber é a mais rápida dentre as avaliadas para todos os níveis de segurança. Evidentemente, a versão CPA não tem o mesmo nível de segurança que a referência em termos de indistinguibilidade de texto cifrado. Porém, seu uso pode ser adequado em uma aplicação onde não é necessário segurança em nível IND-CCA2. A versão CPA pode apresentar qualidade suficiente em tal cenário e ter um melhor desempenho quando comparada às alternativas.

5. Implementação em ambiente restrito

As modificações propostas até então visavam reduzir os requisitos mínimos do algoritmo e viabilizar a implementação em plataformas restritas, que muitas vezes não dispõem dos recursos utilizados pela versão de referência do Kyber. Dessa forma, utilizou-se como alvo de testes a placa de desenvolvimento Freedom FRDM-KL25Z, equipada com MCU KL25Z128 com core ARM Cortex-M0+ com 128KB de memória FLASH, 16KB de memória SRAM e clock de 48MHz. Os testes foram realizados a partir da IDE MCUXpresso, desenvolvida pela NXP, que disponibiliza formas de medir o consumo de memória. Para medição dos ciclos consumidos na execução, implementou-se no algoritmo o contador de ciclos do CMSIS (Cortex Micro-Controller Software Interface and Standard).

Houve problemas com a medição de consumo de memória nesse ambiente e só foi possível avaliar na plataforma restrita até agora os custos de execução das versões nRef e CPA, conforme registrado na Tabela 6.

Tabela 6. Desempenho do Kyber-512 em ciclos de CPU (ARM M0+)

Módulo	Ger (Var.%)	Enc (Var.%)	Des (Var.%)
Kyber512-nRef	6.348.841	8.303.803	8.241.281
Kyber512-CPA	5.298.881 (-16,5)	5.763.296 (-30,6)	1.111.340 (-86,5)

Os resultados mostram que há uma redução nos ciclos de execução no ARM M0+ quando comparados aos dados obtidos no processador Intel i5. A razão dessa diminuição é explicada pela diferença de arquitetura dos dois processadores. Se os dois tivessem a mesma frequência de clock, a execução no ARM M0+ seria mais rápida na prática. Comparando os valores medidos apenas com a CPU ARM M0+, nota-se que a versão CPA exige um número menor de ciclos de CPU do que nRef, assim como foi observado para os dados da CPU Intel i5. Como já foi explicado, o Kyber é um algoritmo com segurança IND-CCA2 em sua versão nRef. A versão de menor segurança CPA utiliza menos cálculos com primitivas criptográficas do que nRef, o que consequentemente reduz o número de ciclos de CPU em ambos processadores.

6. Conclusões e trabalhos futuros

Com as modificações propostas para o Kyber, foi possível reduzir consideravelmente tanto o consumo máximo de memória exigido como os ciclos de CPU necessários para execução em uma plataforma Intel i5. Se o objetivo é executar o algoritmo no menor tempo e com a máxima segurança possível, a melhor opção é a versão nRef (INC-CCA2) em um processador mais poderoso. Por outro lado, se é desejado portar o código para um ambiente mais restrito e não há necessidade de garantir segurança em nível IND-CCA2, a versão mais interessante é a CPA. Trabalhos futuros são necessários de forma a viabilizar o nível de segurança IND-CCA2 com mais baixo custo em ambientes restritos.

Existem implementações do Kyber submetidas ao NIST desenvolvidas para ambientes com processadores de arquitetura ARM Cortex M4. No entanto, por se tratar de um dispositivo de custo elevado, uma pesquisa futura avaliando a aplicação desse algoritmo (e de outros finalistas do NIST) em plataformas de menor custo e com maiores restrições que o Cortex M4 seria interessante para comparar as vantagens de cada candidato em termos da relação custo-benefício.

7. Referências

[1] Regev, O. (2005), “The Learning with Errors Problem”, Courant Institute of Mathematical Sciences, New York University.

[2] Moody, D. (2021), “NIST Status Update on the 3rd Round”, Cryptography Technology Group, National Institute of Standards and Technology.

[3] Hedge, S., Nagapadma, R. (2019), “Number Theoretic Transform for Fast Digital Computation”, Department of Electronic and Communication Engineering, NIE Institute of Technology.

[4] Bertoni, G et al. (2011), “The Keccak reference”, Submission to the NIST SHA-3 Competition, https://keccak.team/files/Keccak-reference-3.0.pdf, January.

[5] Langley, A. (2017), “Maybe Skip SHA-3”, Blog post on ImperialViolet https://www.imperialviolet.org/2017/05/31/skipsha3.html, May.

[6] Bertoni, G. et al. (2017), “Is SHA-3 slow?”, Blog post on KeccakTeam, https://keccak.team/2017/is_sha3_slow.html, June.