Estudos de otimização do algoritmo de criptografia pós-quântica CRYSTALS-KYBER

Luiz F. C. Ferro, Felipe J. A. Rampazzo, Marco A. A. Henriques

Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas (Unicamp) – Campinas, SP

lfcferro01@gmail.com, felipejrampazzo@gmail.com, maah@unicamp.br

Abstract. *The development of quantum computers is getting closer to reality, and as a result the public key cryptographic algorithms used in industry are threatened. In recent years the National Institute of Standards and Technology (NIST) has been carrying out a research over promising quantum-safe cryptographic schemes. All of them demand high computational capability and large key sizes. This paper describes a study about possible optimizations on NIST third round candidate CRYSTALS-KYBER in order to reduce memory consumption and processing time in Intel i5 and ARM Cortex M0+ platforms.*

Resumo. *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) vem promovendo uma avaliação de propostas de algoritmos resistentes ao ataque de computadores quânticos. Todas elas, no entanto, demandam alto poder computacional e/ou exigem chaves de grandes dimensões. Este artigo descreve um estudo sobre possíveis otimizações no algoritmo CRYSTALS-KYBER, candidato da terceira rodada de avaliações do NIST, buscando reduzir seu consumo de memória e tempo de processamento tanto em plataformas Intel i5 como ARM Cortex M0+.*

1. Introdução

Os esquemas de criptografia assimétrica empregados hoje na indústria são considerados seguros, pois os computadores tradicionais não têm capacidade de resolver os problemas matemáticos que os sustentam. No entanto, esse argumento perde sua validade com o desenvolvimento dos computadores quânticos, pois existem algoritmos que conseguem solucionar tais problemas em tempo factível, como é o caso do algoritmo de Shor [1]. Como o advento desses novos computadores em maior escala é apenas uma questão de tempo, diversas entidades da área da segurança da informação têm se mobilizado em busca de modelos de criptografia resistentes a ataques quânticos e que sejam compatíveis com os computadores clássicos que hoje utilizamos.

Com esse propósito, o National Institute of Standards and Technology (NIST), dos Estados Unidos da América, propôs nos últimos anos um programa de pesquisa e competição para atualizar seus padrões e, com isso, incluir algoritmos de criptografia pós-quântica como alvo de estudo. Dada a influência política que esse país desempenha no mundo, muito provavelmente as padronizações definidas nesta área serão posteriormente adotadas pela indústria em escala global. Em 2021, a pesquisa se encontra em sua terceira rodada e traz como candidatos quatro diferentes algoritmos pós-quânticos, fundamentados em premissas distintas e parametrizados para atender diferentes níveis de segurança propostos pelo NIST. Três de quatro dessas opções se

baseiam em técnicas de reticulados e, por essa razão, elas são consideradas as propostas mais promissoras a serem selecionadas pelo NIST.

A implementação desses algoritmos, no entanto, traz dificuldades: todos demandam alto poder computacional, seja em velocidade de processamento ou em espaço necessário para armazenamento de chaves, que são maiores que aquelas dos algoritmos de criptografia de chave pública tradicionais. Nesse sentido, há um agravamento no problema quando é desejado introduzir 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 um algoritmo pós-quântico ou nem mesmo atendem requisitos mínimos para sua implementação e, por esse motivo, algumas pesquisas estudam adaptações nos algoritmos visando esse tipo de aplicação.

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, o presente artigo expõe e analisa diferentes técnicas de otimização para CRYSTALS-KYBER, um dos algoritmos de criptografia pós-quânticos candidatos da terceira rodada de avaliações coordenada pelo NIST, visando melhorar seu desempenho e facilitar sua implementação em ambientes restritos.

2. Conceitos de reticulados e criptossistemas associados

Um reticulado é um conjunto de pontos igualmente espaçados e dispostos em um espaço de n dimensões, sendo esses pontos gerados a partir da combinação linear de vetores linearmente independentes que constituem a base do reticulado. Diferentes bases podem gerar um mesmo reticulado, e é possível classificá-las de acordo com o tamanho e a ortogonalidade de seus vetores: uma base pode ser considerada boa se os vetores que a constituem forem pequenos e relativamente ortogonais, caso contrário, ela é dita uma base ruim, pois apresenta maior complexidade para gerar o reticulado do que a base boa. Na Figura 1, está ilustrado um exemplo de reticulado de duas dimensões gerado por diferentes bases, uma boa (representada pelos vetores em vermelho) e uma ruim (vetores em azul).

Alguns problemas matemáticos são elaborados a partir dos conceitos de reticulados, como é o caso do Closest Vector Problem (CVP) e Shortest Vector Problem (SVP). O primeiro consiste em, conhecendo a base do reticulado e um vetor w qualquer não pertencente a ele, encontrar um outro vetor u contido no reticulado e que minimize a norma entre os dois vetores, ou seja, encontrar o vetor mais próximo a w que esteja contido no reticulado [2]. O SVP pode ser considerado como um caso especial do CVP quando w é nulo, ou seja, é desejado encontrar o menor vetor não nulo pertencente ao reticulado quando somente sua base é conhecida. Existem alguns algoritmos que resolvem esses tipos de problemas, mas a eficiência é dependente da qualidade da base fornecida, ou seja, da ortogonalidade e da norma dos vetores que a compõem.

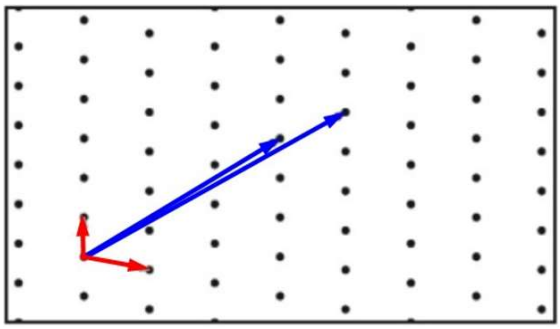

Figura 1. Um mesmo reticulado bidimensional gerado por duas bases distintas.

O GGH (Goldreich-Goldwasser-Halevi) é um dos primeiros criptossistemas fundamentados em reticulados [3] e possui uma similaridade com os problemas descritos anteriormente. Funcionando como um esquema de criptografia assimétrica, a base ruim do reticulado é atribuída a uma chave pública e a base boa, a uma chave privada. Apesar de ser possível cifrar um vetor de mensagem utilizando qualquer uma das bases, utilizar bases ruins para obter a mensagem original não garante a execução correta do sistema, pois somente bases relativamente ortogonais fazem isso.

Atualmente, a maioria dos esquemas de criptografia baseados em reticulados faz uso de uma implementação indireta e robusta do CVP chamada Learning With Errors (LWE), incorporando campos da matemática como anéis polinomiais e aritmética modular [4]. Seu funcionamento consiste na adição de erros (ruídos) em um sistema de equações modulares que compõem a chave pública, isto é, os vetores de uma base do reticulado, e a chave privada é associada à solução desse sistema. Sem a adição de erros, o sistema poderia ser facilmente resolvido a partir da eliminação de Gauss. A função de mão única desse esquema reside justamente nos vetores de ruído que são acrescentados ao sistema, tornando difícil a solução do mesmo sem se conhecer a chave privada.

Ainda existem outros modelos de criptossistemas baseados em reticulados que trazem mais variações do LWE, mas, de certa forma, ainda compartilham muita similaridade no esquema geral de funcionamento. Conforme mencionado anteriormente, esse tipo de criptografia se mostra mais promissor que seus concorrentes e será a principal escolha para padronização [5], pois dentre as quatro propostas de criptografia pós-quânticas finalistas da terceira rodada do NIST, três se baseiam em reticulados: CRYSTALS-KYBER, NTRU e Saber.

3. Segurança para diferentes modelos de ataques

Além dos três candidatos finalistas baseados em reticulados, muitos esquemas de encapsulamento são desenvolvidos com a propriedade de Indistinguibilidade de Texto Cifrado (IND), considerada requisito básico para conferir segurança efetiva aos algoritmos de criptografia de chave pública. Para um esquema ser considerado seguro em termos de indistinguibilidade, um adversário não deve ser capaz 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.

Esquemas modelados com essa propriedade são representados como uma espécie de jogo entre um desafiante e um adversário (máquina de Turing de tempo polinomial probabilística) [6], onde o segundo conhece a chave pública e pode realizar um número limitado de encriptações. Então, ele envia duas mensagens distintas para o desafiante, que realiza o encapsulamento de um trecho dessas mensagens e retorna a cifra ao adversário. Tendo em mãos a chave pública, as duas mensagens originais e o texto cifrado pelo desafiante, a máquina deve realizar as operações que julgar conveniente e, por fim, fornecer o valor do trecho que foi cifrado. Se o adversário não consegue vencer o jogo com uma vantagem desprezível sobre a escolha aleatória, o algoritmo é considerado seguro.

Existem definições de indistinguibilidade que caracterizam três níveis de segurança para um algoritmo, baseados em diferentes modelos de ataques de criptografia. Para todos os tipos de ataque, o esquema de indistinguibilidade do algoritmo é baseado no jogo proposto anteriormente, com pequenas diferenças que conferem algumas vantagens ao atacante. O primeiro nível de ataque, denominado "ataque de texto simples escolhido" (CPA), sugere que o adversário possa enviar diferentes textos em claro para um oráculo e consultar as respectivas mensagens cifradas de volta, assim como é descrito o jogo. No segundo nível, chamado "ataque sob texto cifrado escolhido não adaptativo" (CCA), o adversário agora é capaz de consultar um oráculo de desencapsulamento, enviando mensagens cifradas e obtendo como retorno as respectivas mensagens originais, no entanto, o acesso a essa consulta só é permitido até o momento em que o desafiante envia sua mensagem cifrada. Finalmente, no terceiro e último nível denominado "ataque sob texto cifrado escolhido adaptativo" (CCA2), o adversário pode consultar o oráculo de desencapsulamento mesmo após receber a mensagem cifrada do desafiante e realizar as operações que desejar. No entanto, não é permitido enviar ao oráculo a cifra que foi recebida em específico, caso contrário o jogo teria solução trivial.

Com base nos três diferentes tipos de ataque, a segurança em termos de indistinguibilidade é caracterizada, portanto, na seguinte ordem crescente de segurança: IND-CPA, IND-CCA e IND-CCA2. Garantir segurança de indistinguibilidade sob o último nível de ataque (CCA2) também confere segurança em relação aos demais níveis. Conforme dito, os algoritmos de criptografia pós-quântica baseados em reticulados e finalistas da terceira rodada do NIST, são todos desenvolvidos com segurança do tipo IND-CCA2. Para avaliar profundamente o funcionamento desses algoritmos e os requisitos exigidos para atender ao tipo de segurança CCA2, as seguintes seções se dedicam a explicar e propor mudanças para o CRYSTALS-KYBER, um dos três candidatos finalistas mencionados.

4. CRYSTALS-Kyber

Kyber é um mecanismo de encapsulamento de chaves (key-encapsulation mechanism ou KEM) da família CRYSTALS (Cryptographic Suite for Algebraic Lattices) do tipo IND-CCA2, cuja segurança é baseada na dificuldade de resolução de uma variação do LWE sobre reticulados. Ao invés de utilizar a modelagem exata do Learning With Errors, os vetores de n dimensões são substituídos por polinômios de grau menor que n compostos por elementos inteiros e operações modulares [7]. Essa

construção garante um aumento na segurança em relação à proposta original e é chamada de Module-LWE (também chamada de MLWE). Somente substituir os vetores por polinômios constituídos de elementos inteiros ao invés de modulares caracteriza uma outra variação do LWE, chamada de Ring-LWE ou RLWE, que tem eficiência similar ao MLWE.

A submissão do algoritmo ao NIST propõe três diferentes configurações de parâmetros, buscando atingir diferentes níveis de segurança, em ordem crescente: Kyber-512, Kyber-768 e Kyber-1024, sendo a dimensão empregada na base do reticulado como principal diferença entre essas versões. Durante as rodadas da competição, foi solicitado que os candidatos submetessem essas variações para criar estimativas preliminares da segurança de cada nível. Dessa forma, o NIST propôs cinco diferentes categorias utilizando como referência a segurança de algoritmos de criptografia já conhecidos e empregados, como o AES e SHA. Com isso, os desenvolvedores do Kyber modelaram e submeteram suas propostas para a terceira rodada, estimando níveis 1, 3 e 5 de segurança para as versões Kyber-512, Kyber-768 e Kyber-1024, respectivamente, como mostra a Tabela 1.

Tabela 1. Níveis de segurança para avaliação dos algoritmos candidatos do NIST

Nível	Descrição de Segurança
I	Tão difícil de quebrar quanto o AES128 (busca exaustiva)
II	Tão difícil de quebrar quanto o SHA256 (busca de colisão)
III	Tão difícil de quebrar quanto o AES192 (busca exaustiva)
IV	Tão difícil de quebrar quanto o SHA384 (busca de colisão)
V	Tão difícil de quebrar quanto o AES256 (busca exaustiva)

O processo de criptografia do Kyber é dividido em três partes: geração de chaves, encapsulamento e desencapsulamento. Inicialmente, esses passos são executados e constrói-se um esquema de encapsulamento com segurança do tipo IND-CPA que cifra mensagens de 32 bytes de comprimento. Em seguida, a partir de ajustes em um processo de conversão de esquemas do tipo CPA para CCA, denominado "Transformação Fujisaki-Okamoto" [8], uma segunda etapa é executada para geração, encapsulamento e desencapsulamento para construir um algoritmo com segurança IND-CCA2.

Ao longo do desenvolvimento do esquema, a versão de referência do Kyber faz uso de algumas funções hash como primitivas referenciadas pelo padrão FIPS 202 [9], publicado pelo NIST, sendo elas em sua maioria pertencentes à família SHA3 e SHAKE. Além disso, muitos cálculos são realizados a partir de uma implementação da transformada rápida de Fourier chamada Number Theoretic Transform (NTT) [10],

forma eficiente de realizar operações em anéis e em corpos finitos. Como os cálculos envolvem números grandes e graus altos de polinômios, a utilização dessa ferramenta facilita a implementação do Kyber e reduz seu tempo de execução consideravelmente.

Para a construção de chaves de segurança IND-CPA, o processo consiste na geração de uma matriz quadrada A de dimensão n (comprimento da entropia de entrada, fixado em 256 bits) no domínio NTT eleita como uma chave pública. Além disso, duas outras matrizes s e $e1$ distintas de dimensão $n \times k$ são geradas a partir das primitivas, representando chave privada e os ruídos, respectivamente (k define a dimensão do reticulado). Dessa forma, é realizada a operação $A*s + e1 = t$, onde o resultado t é uma matriz $n \times k$, e serve para referenciar uma outra chave pública que será utilizada para realizar uma parcela da cifragem, que consiste no encapsulamento de um dado (chave simétrica aleatória) de 32 bytes.

O encapsulamento ocorre em duas etapas. Inicialmente, é gerada uma matriz r de valores aleatórios e dimensão $k \times n$, utilizada como uma chave efêmera para realizar a operação $r*A + e1^T = u$, sendo a matriz u de dimensão $k \times n$ um trecho do encapsulamento. Na segunda etapa, é gerado um outro vetor de ruído $e2$ de mesma dimensão da mensagem, representada por uma matriz quadrada m de ordem k, para realizar a operação $r*t + e2 + (q/2)*m = v$. O valor de q é um número primo escolhido para possibilitar as operações do NTT e o resultado obtido v é uma matriz quadrada de ordem k, o segundo trecho do encapsulamento. Finalmente, obtém-se a mensagem encapsulada ao concatenar as matrizes u e v.

Para o desencapsulamento, o algoritmo calcula $u*s = w$ e, em seguida, faz a subtração v-w. O resultado dessa operação é aproximadamente igual a $(q/2)*m$. O algoritmo faz um arredondamento com base na norma desses valores: se v-w está mais próximo de $q/2$ do que de zero, a mensagem é um; caso contrário é zero.

Após a execução dessas três etapas, a abordagem de construção do algoritmo de segurança CCA2 basicamente implementa novamente essas funções, atribuindo como chave o resultado da concatenação dos valores obtidos da primeira execução e incluindo cálculos com funções hash das demais primitivas que foram instanciadas seguindo o padrão FIPS 202, como funções de derivação de chave, por exemplo.

Ainda existem duas 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 SHAKE por AES e SHA3 por SHA2. Outra versão do Kyber é referente a uma otimização que utiliza um conjunto de instruções vetoriais específicas AVX2 (Advanced Vector Extensions), presentes em microprocessadores Intel e AMD mais desenvolvidos e que reduz significativamente o tempo de execução do algoritmo. Como o foco deste trabalho é também buscar formas de otimizar o algoritmo para ambientes mais simples e limitados que não apresentam suporte para esse conjunto de instruções, essa versão AVX2 não será tratada. Para avaliar o desempenho do algoritmo, leva-se em consideração dois principais parâmetros: tempo de execução e memória máxima ocupada. A importância de cada uma dessas métricas pode ser ponderada de acordo com o cenário de aplicação e seu fator limitante.

5. Propostas de otimização 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 ou então para redução no tempo de processamento. Graças às eficientes operações de multiplicação no domínio NTT, a velocidade do Kyber é majoritariamente determinada pelo desempenho das primitivas empregadas.

O design do algoritmo necessita de uma função hash de saída extensível (XOF), duas funções hash H e G convencionais, uma função pseudo-aleatória (PRF) e uma função de derivação de chave (KDF). A versão de referência foi implementada utilizando uma única família de primitivas baseada no hash Keccak [12] e padronizada por meio do FIPS 202 como SHA3. Dessa forma, as primitivas utilizadas para as versões de referência e 90s estão expostas na Tabela 2.

Tabela 2. Primitivas instanciadas para diferentes versões do Kyber512

Versão	XOF	PRF	KDF	Hash H	Hash G
Ref.	SHAKE128	SHAKE256	SHAKE256	SHA3-256	SHA3-512
90s	AES256	AES256	SHA256	SHA256	SHA512
v1	AES256	AES256	SHA256	SHA3-256	SHA3-512
v2	SHAKE128	SHAKE256	SHAKE256	SHA256	SHA512

A função SHA3 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 [13]. Apesar da similaridade entre as estruturas da função SHA3 e SHAKE, a primeira foi desenvolvida com parâmetros de segurança excessivos e que são desnecessários atualmente [14]. Por isso, o NIST aprovou anos atrás as funções SHAKE, que flexibilizam essa configuração excessiva e torna o desempenho da função próximo ao SHA2 em processadores comuns.

Com isso, buscou-se avaliar o impacto no desempenho do Kyber criando versões com diferentes combinações das primitivas utilizadas nas versões Ref. e 90s. O objetivo dessa mudança é verificar qual primitiva tem maior influência na eficiência do algoritmo quando não se dispõe de nenhum recurso de aceleração em hardware para elas. A Tabela 2 exibe, além das primitivas das versões de referência e 90s, duas novas versões com diferentes combinações, intituladas “v1” e “v2”.

Outro 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 aplicação extremamente importante para conferir segurança ao algoritmo e de alta credibilidade, seu uso no algoritmo não é obrigatório e pode ser substituído. Além disso, carregar a biblioteca para uma única aplicação consome um espaço de memória indesejável. Dessa forma, avaliou-se o impacto na remoção da OpenSSL, utilizando agora 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 um fator 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.

6. Implementação em plataforma Intel i5

Esta seção analisa o desempenho de otimizações do algoritmo de referência em plataforma Intel i5, sem restrições de memória ou de capacidade de processamento. A versão 3.01 da implementação de referência do Kyber submetida à terceira rodada do NIST [11] é usada como versão de comparação inicial e é denotada por "Ref.". Para as avaliações nesta plataforma, utilizou-se o compilador GCC 11.0.1 e o software Valgrind como medidor do consumo de memória total (stack e heap) e a ferramenta Perf para medir os ciclos de execução dos algoritmos em uma máquina com processador Intel Core i5-4570 (Haswell) de 3.20GHz, executando o sistema operacional Linux Fedora 34. Um detalhe importante a mencionar é que não foram utilizados recursos de otimização em hardware para execução do algoritmo AES que estão disponíveis nesta plataforma, de maneira a facilitar uma comparação mais direta com uma implementação em ambiente restrito.

De modo a facilitar comparações com implementações em uma plataforma restrita, na qual não há disponibilidade de memória abundante e nem de sistema de arquivos, optou-se por remover da implementação de referência do NIST (Ref.) o uso da biblioteca OpenSSL (usada apenas na geração de números pseudo-aleatórios) e das chamadas de acesso a arquivos. A Tabela 3 exibe as variações de ciclos de CPU e de consumo máximo de memória quando se faz estas mudanças, sendo essa nova versão de referência intitulada "nRef". A tabela expõe os dados comparativos para geração de chaves (Ger), encapsulamento (Enc) e desencapsulamento (Des) para a versão de 512 bits de segurança do Kyber. As medições obtidas mostram que há um ganho de eficiência no algoritmo para ambas as métricas, principalmente para os ciclos de CPU: nos três módulos há uma redução entre 27,7% e 40,9% no número de ciclos em relação à versão Ref. Já para as medições de consumo de memória, há uma redução um pouco maior nos requisitos mínimos. Por ora, a geração de números pseudo-aleatórios está sendo feita com a função *rand()* da biblioteca da linguagem C, já que a mesma também está disponível para os testes em plataforma restrita, facilitando comparações. Com base nesses dados, foi possível realizar uma comparação de consumo máximo de memória e de ciclos de CPU entre a versão adaptada de referência nRef e as versões v1 e v2. Nas Tabelas 4 e 5, tais valores estão expressos para todos os tamanhos de chaves do Kyber (512, 768 e 1024). Além dessas comparações, também foram incluídos os resultados de

desempenho para a versão CPA do Kyber, isto é, a versão que não oferece o mesmo nível de segurança que IND-CCA2, mas que faz uso das mesmas primitivas da versão de referência. Isso pode ser de utilidade em aplicações que não exigem um nível tão elevado de segurança e, portanto, podem trabalhar com IND-CPA.

Tabela 3. 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	1.484.852	-	19,6	-
Ger: nRef	888.162	-40,2%	10,5	-46,4%
Enc: Ref	1.599.078	-	21,6	-
Enc: nRef	1.098.395	-31,3%	12,4	-42,6%
Des: Ref	2.410.518	-	26,8	-
Des: nRef	1.742.067	-27,7%	17,1	-36,2%

Tabela 4. Consumo de memória (KiB) das versões de Kyber (Intel i5)

Módulo	Ger (Var.%)	Enc (Var.%)	Des (Var.%)
Kyber512-nRef	10,5	12,4	17,1
Kyber512-v1	10,8 (+2,9)	13,4 (+8,1)	18,2 (+6,4)
Kyber512-v2	10,5 (0,0)	12,4 (0,0)	17,1 (0,0)
Kyber512-CPA	9,6 (-8,6)	12,3 (-0,8)	8,6 (-49,7)
Kyber768-nRef	15,6	18,3	25,1
Kyber768-v1	15,8 (+1,3)	19,4 (+6,0)	26,2 (+4,4)
Kyber768-v2	15,6 (0,0)	18,3 (0,0)	25,1 (0,0)
Kyber768-CPA	14,4 (-7,7)	18,3 (0,0)	11,6 (-53,8)
Kyber1024-nRef	21	25,3	34,6
Kyber1024-v1	22 (+4,8)	26,1 (+3,2)	35,4 (+2,3)
Kyber1024-v2	21 (0,0)	25,3 (0,0)	34,6 (0,0)
Kyber1024-CPA	19,4 (-7,6)	25,2 (-0,4)	15,1 (-56,4)

Conforme mostra a Tabela 4, ao realizar a alteração da versão nRef para v1, que utiliza somente primitivas AES e SHA2 sem instruções de aceleração em hardware, ocorre um aumento no consumo de memória para todos os processos de funcionamento do Kyber (geração de chaves, encapsulamento e desencapsulamento) em todos os níveis. Esse resultado indica que as primitivas SHAKE128 e SHAKE256, utilizadas na versão nRef, são mais leves que as funções utilizadas em v1. Para uma mesma versão, a geração de chaves se mostrou o processo com menor consumo de memória, exceto na versão CPA, onde o desencapsulamento é o módulo de menor consumo.

Tabela 5. Ciclos de CPU gastos em cada versão de Kyber (Intel i5)

Módulo	Ger (Var.%)	Enc (Var.%)	Des (Var.%)
Kyber512-nRef	888.162	1.200.971	1.742.067
Kyber512-v1	994.619 (+11,9)	1.289.576 (+17,4)	1.885.046 (+8,2)
Kyber512-v2	923.073 (+3,9)	1.129.635 (+2,8)	1.672.944 (-4,0)
Kyber512-CPA	835.574 (-5,9)	1.133.287 (-5,6)	1.334.033 (-23,4)
Kyber768-nRef	962.312	1.352.761	2.242.718
Kyber768-v1	1.137.623 (+18,2)	1.571.182 (+16,1)	2.509.356 (+11,9)
Kyber768-v2	1.103.698 (+14,7)	1.314.767 (-2,8)	2.230.337 (-0,6)
Kyber768-CPA	968.011 (+0,6)	1.359.693 (+0,5)	1.622.706 (-27,6)
Kyber1024-nRef	1,081,047	1.603.565	3.041.229
Kyber1024-v1	1.431.944 (+32,5)	2.038.091 (+27,1)	3.354.869 (+10,3)
Kyber1024-v2	1.002.317 (-7,3)	1.623.423 (+1,2)	2.941.116 (-3,3)
Kyber1024-CPA	1.070.510 (-1,0)	1.620.767 (+1,1)	1.993.409 (-34,5)

Agora, ao comparar a versão nRef com a versão v2, que se diferencia somente por utilizar SHA2 ao invés de SHA3, não há mudança alguma em nenhuma versão: os dados de consumo de memória são iguais para ambas as versões, indicando que não há vantagem em termos dessa métrica ao escolher entre SHA2 e SHA3. Para a versão CPA, nota-se que há uma redução no consumo de memória para todos os níveis de segurança, principalmente para o processo de desencapsulamento, que tem a memória reduzida aproximadamente pela metade. Naturalmente, isso ocorre porque esse esquema do Kyber não faz uso da transformação para CCA2,o que acarreta em menos chamadas de funções hash e, consequentemente, menos uso de memória. Na Tabela 5, estão expostos os dados referentes às mesmas modificações nos algoritmos, agora avaliando o impacto nos ciclos de CPU. Nela, os dados mostram que a versão v1 (que troca primitivas

SHAKE por SHA2 e por AES) causa um impacto negativo no desempenho de geração de chaves, encapsulamento e desencapsulamento para todos os níveis de segurança do Kyber. Esse impacto é ainda maior quando é avaliada a versão com maior nível de segurança, o Kyber1024. No entanto, quando são comparadas as implementações nRef com v2, que se diferem somente pelo uso do SHA2 ao invés de SHA3, nota-se que há impactos positivos e negativos nos ciclos de CPU. Logo, a adoção da versão v2 não traz vantagens no uso de memória e nem em ciclos de CPU que justifiquem sua adoção.

A versão que considera somente o esquema de segurança IND-CPA é a mais rápida dentre as avaliadas para todos os níveis de segurança, justamente por utilizar menos funções de hash. Evidentemente, essa última proposta não tem a mesma garantia de segurança que as demais em termos de indistinguibilidade de texto cifrado, porém ela pode ser usada, dependendo do contexto da aplicação. Para uma situação onde não é necessário segurança em termos IND-CCA2, o uso da versão CPA pode ser interessante por apresentar qualidade suficiente nesse cenário e ter um melhor desempenho no tempo de execução, além de também ter o menor consumo de memória quando comparado às outras propostas estudadas.

7. Implementação em ambiente restrito

Para testes em ambientes mais restritos, utilizou-se a placa de desenvolvimento Freedom FRDM-KL25Z, equipada com MCU KL25Z128, core ARM Cortex-M0+, 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 apenas. 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 (por limitações no ambiente de desenvolvimento e por estar trabalhando no limite da capacidade da placa), mas foi possível executar os módulos de geração de chaves, encapsulamento e desencapsulamento de chaves de sessão e avaliar os custos de execução das versões nRef, v1 e CPA do Kyber512, conforme registrado na Tabela 6. Neste ambiente não foi avaliada a versão v2. Mesmo considerando os 16KiB de memória SRAM disponíveis na placa, o limite real foi de cerca de 13 KiB e o sucesso na execução desses processos mostra que nenhuma etapa do Kyber teve consumo superior a esse valor. Apesar de existirem relatos de implementações em plataformas ARM Cortex M4 [15], entendemos ser interessante viabilizar o uso de Kyber em plataformas ainda mais restritas e de menor custo como a M0+, a fim de explorar os limites de otimização possíveis nestes ambientes.

Os resultados mostram inicialmente um grande aumento no número de ciclos de CPU necessários para a execução dos módulos de geração de chaves, encapsulamento e desencapsulamento, quando são comparados com as execuções em ambiente Intel i5, com aumentos de mais de 600% nos dois primeiros módulos citados e de mais de 300% no terceiro módulo. Um aumento no número de ciclos já era esperado com a troca de um processador CISC (Intel i5) para outro RISC (ARM Cortex M0+), o qual tem um conjunto de instruções mais simples e reduzido, exigindo assim mais instruções de máquina (e mais ciclos) para um mesmo conjunto de ações determinadas pelos

programas. Considerando ainda que a frequência de clock de um processador em ambiente restrito é significativamente mais baixa que aquela usada em ambientes sem muitas restrições (servidores, desktops e notebooks), conclui-se que o aumento no tempo de execução deste algoritmo pós-quântico em ambiente restrito poderá ser uma forte barreira à sua adoção.

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

Módulo	Ger (Var.%)	Enc (Var.%)	Des (Var.%)
Kyber512-nRef	6.318.033 (+611,4*)	8.272.196 (+653,1*)	8.242.426 (373,1*)
Kyber512-v1	13.940.229 (+120,6)	16.191.563 (+95,7)	16.175.480 (+96,2)
Kyber512-CPA	5.298.715 (-16,1)	5.764.486 (-30,3)	1.111.233 (-86,5)

*Comparação relativa à versão executada em Intel i5.

Comparando agora os valores medidos apenas com a CPU ARM Cortex M0+, nota-se que a versão v1 apresenta aumentos significativos em números de ciclos, praticamente dobrando o tempo de execução em relação à versão nRef. Estes aumentos são muito superiores aos observados na plataforma Intel i5 e precisam ser melhor avaliados para um entendimento mais completo de sua natureza. Já a versão CPA exige um número menor de ciclos de CPU do que nRef, assim como foi observado na 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. Entretanto, cabe notar que mesmo a redução entre 16,1% e 86,5% observados na versão CPA não é suficiente para compensar o grande aumento no número de ciclos causado pela troca da arquitetura de CPU. Isso mostra que novos esforços são necessários para melhorar o desempenho do algoritmo Kyber em ambientes restritos, onde predominam plataformas tipo RISC.

8. Conclusões e trabalhos futuros

Devido à presença majoritária dos esquemas baseados em reticulados nos finalistas para criptografia pós-quântica, é possível que o NIST padronize ao menos um desses esquemas ao final da terceira rodada. É importante que, além de ser eficiente e seguro, o algoritmo escolhido mostre potencial para aperfeiçoamentos futuros e aplicabilidade em cenários variados, garantindo segurança pós-quântica para qualquer ambiente, ainda que tenha restrições de processamento e de memória.

Com algumas das modificações propostas para o algoritmo Kyber, foi possível reduzir tanto o consumo máximo de memória como o de ciclos de CPU em plataformas Intel i5 e ARM Cortex M0+. Se o objetivo é executar o algoritmo no menor tempo e com a máxima segurança possível, a melhor opção é a versão Ref ou nRef (INC-CCA2) em um processador mais poderoso e com menos restrições de processamento e memória. 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 (baseada em nRef). Entretanto, deve ser notado que a maior popularidade de processadores RISC e a adoção de frequências de clock mais baixas em ambientes restritos causam um aumento significativo no tempo de execução nestes ambientes, o que traz uma maior dificuldade de adoção de algoritmos de criptografia pós-quântica nos mesmos.

Trabalhos futuros são necessários de forma a melhorar o desempenho do algoritmo Kyber em plataformas mais caras e em ambientes restritos, tanto do ponto de vista de consumo de memória como de ciclos de CPU. Também são necessários esforços para viabilizar a adoção de segurança IND-CCA2 com custos computacionais compatíveis com plataformas mais restritas. Além disso, análises e otimizações dos outros algoritmos em avaliação pelo NIST também são necessárias para se buscar melhores combinações entre software e hardware para diferentes tipos de aplicação.

Agradecimentos: agradecemos ao acadêmico Éric Ribeiro Daher pelas discussões iniciais sobre implementação do algoritmo em ambientes restritos.

9. Referências

[1] Shor, P. (1997), "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", SIAM Journal on Computing.

[2] Micciancio, D., Regev, O. (2008), "Lattice-based cryptography", CSE Department, University of California, San Diego.

[3] Goldreich, O., Goldwasser, S. & Halevi, S. (1997), "Public-key cryptosystems from lattice reduction problems", Lecture Notes in Computer Science, vol. 1294.

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

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

[6] Bogdanov, D. (2005), "IND-CCA2 secure cryptosystems", University of Tartu.

[7] Albrecht, M., Deo, M. (2017), "Large Modulus Ring-LWE ≥ Module-LWE", Information Security Group, Royal Holloway, University of London.

[8] Hofhein, D., Hövelmanns, K. & Kiltz, E. (2017), "A Modular Analysis of the Fujisaki-Okamoto Transformation", Karlsruhe Institute of Technology.

[9] May, W. (2015), "SHA-3 Standard: Permutation-Based and Extendable-Output Functions", Federal Information Processing Standards Publication, National Institute of Standards and Technology.

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

[11] Avanzi, R. et al. (2021), "CRYSTALS-Kyber - Submission to round 3 of the NIST post-quantum project", https://pq-crystals.org/kyber/resources.shtml, January.

[12] 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.

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

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

[15] Kannwischer, M. et al. (2019), “pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4”, Radboud University, Netherlands.