Avaliando a Prova de Posse de chaves KEM baseada em Geração Verificável

Gabriela Strieder Ramos[1], **Andrei Filipim Esteves**[1], **Alexandre Augusto Giron**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal 85902-490 – Toledo – PR – Brazil

gabrielaramos,andreifilipim}@alunos.utfpr.edu.br

Abstract. *In response to the threat that quantum computers pose to security, the migration to quantum-resistant cryptography using Key Encapsulation Mechanisms (KEMs) for Internet authentication is proposed, aiming for better performance. Thus, this work aimed to analyze the feasibility of PoPs for KEMs and compare their performance, focusing on the FrodoKEM algorithm. The comparative analysis between implementations in C and Go showed that, although C is faster, the main problem is the computational cost of the "Proof of Possession", which is orders of magnitude higher than the base mechanism. This problem could affect scalability and prevents its use in high-security scenarios demanding agility.*

Resumo. *Em resposta à ameaça dos computadores quânticos à segurança, surge a proposta de migração para a criptografia resistente ao ataque quântico usando Key Encapsulation Mechanisms (KEMs) para autenticação na Internet, visando melhor desempenho. Assim, este trabalho teve por objetivo analisar a viabilidade de PoPs para KEMs e comparar seu desempenho, focando-se no algoritmo FrodoKEM. A análise comparativa entre implementações em C e Go mostrou que, embora C seja mais rápida, o principal problema é o custo computacional da "Prova de Posse"ser em ordens de magnitude superiores ao mecanismo base. Isto pode afetar a escalabilidade, dificultando seu uso em cenários de alta segurança que demandam agilidade.*

1. Introdução

A segurança da economia digital, atualmente baseada em algoritmos como RSA e ECC, está sob ameaça iminente devido ao avanço dos computadores quânticos, que podem quebrar essas cifras utilizando o algoritmo de Shor [Shor 1994]. Como resposta, surge a criptografia pós-quântica, onde os Mecanismos de Encapsulamento de Chaves (KEMs) se destacam como uma solução versátil e de bom desempenho para compartilhar chaves de forma segura e até mesmo para realizar autenticação [Güneysu et al. 2022], com propostas para sua integração em protocolos como TLS e VPNs.

Embora a hipótese seja de que os KEMs pós-quânticos avaliados pelo NIST [NIST 2022], como o FrodoKEM, ofereçam um balanço entre segurança e eficiência, um desafio crítico impede sua adoção ampla: a certificação de suas chaves. Protocolos de certificação automatizada, como o ACME [Barnes et al. 2019], não possuem suporte nativo para KEMs. Existem propostas para uma "Prova de Posse"(PoP) que resolveria essa questão [Güneysu et al. 2022], porém, falta uma análise de viabilidade, com implementações em diferentes linguagens e testes de desempenho para avaliar sua eficácia.

O objetivo deste trabalho é avaliar o desempenho das Provas de Posse (PoP) na geração de chaves do FrodoKEM, contribuindo com uma implementação em Go que integra a

técnica proposta por Gunyesu et al. [Güneysu et al. 2022] para permitir seu uso futuro em implementações de ACME. Ao analisar o custo computacional desta implementação, o estudo busca fornecer a base técnica e os dados de desempenho necessários para o futuro desenvolvimento de ferramentas, como plugins para Certbot ou OpenSSL, que viabilizem a emissão de certificados X.509 resistentes a ataques quânticos.

A escolha do algoritmo FrodoKEM neste trabalho se da por conta de ele ter sido estabelecido como uma opção recomendada por autoridades europeias, como o BSI da Alemanha [Federal Office for Information Security 2023] e a ANSSI da França [Agence Nationale de la Sécurité des Systèmes d'Information 2023], para assegurar a confidencialidade de dados a longo prazo. Mesmo não tendo sido selecionado na padronização final do NIST, sua abordagem conservadora ainda atende os requisitos das agências mencionadas.

O artigo está organizado da seguinte forma. A Seção 2 apresenta os fundamentos teóricos deste trabalho, incluindo os conceitos de criptografia pós-quântica, mecanismos de encapsulamento de chaves (KEMs) e provas de posse (PoPs), além de uma explicação sobre o algoritmo FrodoKEM. A Seção 3 detalha a metodologia utilizada para analisar a viabilidade e comparar o desempenho das PoPs para KEMs. A Seção 4 discute os resultados obtidos com os experimentos de desempenho e as implicações para a certificação de chaves KEM. Finalmente, a Seção 5 conclui o trabalho e sugere direções para pesquisas futuras.

2. Referencial Teórico

Esta seção detalha os conceitos relacionados aos KEMs, ao processo de padronização de criptografia pós-quântica do NIST, o trabalho-base sobre Provas de Posse (PoP) de chaves KEM[Güneysu et al. 2022] e possibilidades de integração.

2.1. Key Encapsulation Mechanism (KEM) e Assinatura Digital

O Mecanismo de Encapsulamento de Chaves (KEM), essencial para protocolos modernos como o TLS, é um esquema criptográfico focado em trocar chaves simétricas de forma segura em canais não confiáveis, ao encapsular uma chave temporária que o destinatário recupera com sua chave privada. Em contraste, a assinatura digital não se concentra na confidencialidade, mas sim em garantir a autenticidade e a integridade das mensagens, utilizando um par de chaves para validar a identidade do remetente e confirmar que o conteúdo não foi alterado.

2.2. Algoritmo: FrodoKEM

O FrodoKEM é um algoritmo de criptografia pós-quântica do tipo KEM, fundamentado no problema de Aprendizagem com Erros (LWE), conforme descrito por sua organização [FrodoKEM 2025]. O algoritmo apresenta diferentes variantes (como FrodoKEM-640, -976 e -1344), que correspondem a níveis de segurança crescentes (NIST 1, 3 e 5). Basicamente, essas variantes mostram um trade-off: um nível de segurança mais elevado exige chaves e textos cifrados maiores, o que, por sua vez, demanda mais recursos computacionais e de armazenamento.

2.3. Geração de Chaves com Prova de Posse

O artigo de PoP [Güneysu et al. 2022] descreve uma abordagem para gerar chaves e uma provas de posse para Mecanismos de Encapsulamento de Chaves (KEMs), com foco na criptografia pós-quântica. A ideia central é que o detentor de um par de chaves possa fornecer uma prova que autentica a posse da chave privada de forma não interativa, o que é crucial em cenários

como a emissão de certificados para garantir a propriedade da chave sem comprometer a segurança. O artigo detalha como o protocolo pode ser ajustado para diferentes KEMs, como o FrodoKEM, e analisa o custo computacional e os parâmetros de segurança envolvidos na geração e verificação dessa prova.

3. Metodologia

Para avaliar o desempenho e as características das diferentes variantes do Frodo, será adotada uma metodologia de testes quantitativa. O foco principal dos testes será a avaliação das variantes dos algoritmos Frodo, especificamente Frodo-640, Frodo-976 e Frodo-1344. Os testes abrangerão as operações fundamentais de cada esquema, incluindo geração de chaves, encapsulamento e desencapsulamento. O objetivo é analisar o comportamento de cada variante em termos de tempo de médio de execução e desvio padrão.

Para a execução do algoritmo FrodoKEM, foi configurada uma máquina virtual (VM) na plataforma Google Cloud, localizada na região de São Paulo (southamerica-east1). A instância, do tipo n2-standard-4, foi equipada com 4 vCPUs, 2 núcleos e 16 GB de memória, utilizando o sistema operacional Ubuntu 24.04 LTS Minimal em um disco de 100 GB. As configurações de rede foram ajustadas para permitir tráfego HTTP e HTTPS, facilitando a comunicação com repositórios e bibliotecas externas.

Os experimentos serão conduzidos na linguagem C e GO com o objetivo de avaliar o desempenho das versões do FrodoKEM dependendo dos parâmentros utilizados. Essa avaliação em Go é necessária para facilitar a integração prática da técnica com implementações ACME existentes e para entender seus custos antes do surgimento de computadores quânticos que possam quebrar a criptografia atual.

Durante os testes, são avaliadas variações nos parâmetros `N` e `TAU` dos esquemas ZKPOP para Frodo. Esses parâmetros influenciam diretamente o tamanho da prova e o nível de segurança do protocolo. O parâmetro `N` está relacionado ao tamanho do problema matemático subjacente, como a dimensão das matrizes em Frodo, impactando a complexidade e o custo computacional da prova. Por sua vez, o parâmetro `TAU` ajusta a força das provas dentro do esquema ZKPOP, afetando o nível de segurança e o tamanho final da prova gerada. Ambos os parâmetros podem ser configurados por meio dos comandos utilizados para executar o código nos repositórios.

3.1. Testes do FrodoKEM

Para avaliar o tempo médio e o desvio padrão das operações do FrodoKEM, foi utilizados os seguintes repositórios de código-fonte:

- Repositório PQCrypto-LWEKE [Microsoft Corporation 2025]: Contém o código-fonte oficial do FrodoKEM.
- Repositório KEM-NIZKPoP [Chair for Security Engineering 2025]: Contém os artefatos referentes ao artigo "Proof-of-possession for KEM certificates using verifiable generation"[Güneysu et al. 2022]. Este repositório oferece uma implementação provas de posse para certificados KEM baseada nas versões originais dos algoritmos FrodoKEM em C.
- Repositório FrodoKEM em GO [Ramos et al. 2025]: Implementa um biding do algoritmo FrodoKEM em GO. Ao executar o algoritmo do Frodo, gera dados sobre a prova de posse do FrodoKEM.

Para gerar os dados apresentados na Tabela 1, foram compilados e executados dois repositórios distintos em C, cada um focado em um aspecto diferente do KEM Frodo.

Repositório PQCrypto-LWEKE e Repositório KEM-NIZKPoP: Foram realizados testes de desempenho nestes dois repositórios distintos para avaliar as variantes 640, 976 e 1344 do FrodoKEM. O repositório PQCrypto-LWEKE gera as operações padrão de geração de chaves, encapsulamento e desencapsulamento. Já no repositório KEM-NIZKPoP, o foco foi a Prova de Posse (NIZKPoP), com funções de geração "KeyGen"e verificação da prova "Verify". Todos os testes usaram os parâmetros N=31 e TAU=26, para 1000 execuções, e os resultados de ambas as avaliações foram consolidados na Tabela 1.

Tabela 1. Resultados de desempenho agrupados por repositório em C

Repositório: PQCrypto-LWEKE			
Modelo	**Operação**	**Tempo médio (s)**	**Desvio padrão (s)**
FrodoKEM-640	Geração de chaves	0.000473385	0.000007645
	Encapsulamento	0.000572191	0.000047390
	Desencapsulamento	0.000539776	0.000013270
FrodoKEM-976	Geração de chaves	0.000994658	0.000010958
	Encapsulamento	0.001081776	0.000010558
	Desencapsulamento	0.001038351	0.000010784
FrodoKEM-1344	Geração de chaves	0.001741351	0.000011767
	Encapsulamento	0.001783777	0.000015288
	Desencapsulamento	0.001710627	0.000015530
Repositório: KEM-NIZKPoP			
Versão Frodo	**Operação**	**Tempo médio (s)**	**Desvio padrão (s)**
FrodoKEM-640	NZKPoP KeyGen	0.226718	0.001318
	NZKPoP Verify	0.229480	0.000983
	NZKPoP Total	0.228099	0.002301
FrodoKEM-976	NZKPoP KeyGen	0.418372	0.003908
	NZKPoP Verify	0.367920	0.003984
	NZKPoP Total	0.393146	0.007892
FrodoKEM-1344	NZKPoP KeyGen	0.723120	0.006458
	NZKPoP Verify	0.728452	0.006467
	NZKPoP Total	0.725786	0.012925

Repositório FrodoKEM em GO: Foram realizadas 1000 execuções com as três variantes do FrodoKEM (640, 976 e 1344), em quatro etapas principais: a geração de chaves, o encapsulamento, o decapsulamento e a geração e verificação de provas ZKPoP. No teste, foi usado os parâmetros $N = 31$ e $\tau = 26$.Os dados podem ser observados na tabela 2 .

Além disso, foi feito outro teste com 100 execuções para o Frodo 640, com $N = 65536$ e $\tau = 8$. Os dados podem ser observados na Tabela 3.

4. Resultados e Discussão

A Tabela 4, compila os resultados de todas as versões do FrodoKEM para os tempos médios e desvios-padrão em segundos, para os parâmetros $N = 31$ e $\tau = 26$, para 1000 execuções. Posteriomente, são apresentados os resultados do FrodoKEM 640, com os tempos médios e desvios-padrão em segundos, para os parâmetros $N = 65536$ e $\tau = 8$, para 100 execuções.

Tabela 2. Resultados de desempenho do repositório em GO

Versão Frodo	Operação	Tempo Médio (s)	Desvio padrão (s)
Frodo-640	Geração de chaves	0.000519	0.000050
	Encapsulamento	0.000570	0.000033
	Desencapsulamento	0.000536	0.000010
	ZKPoP KeyGen	0.264781	0.001002
	ZKPoP Verify	0.220503	0.001223
	ZKPoP Total	0.242642	0.002225
Frodo-976	Geração de chaves	0.001057	0.000088
	Encapsulamento	0.001079	0.000041
	Desencapsulamento	0.000994	0.000014
	ZKPoP KeyGen	0.438160	0.002790
	ZKPoP Verify	0.380957	0.003376
	ZKPoP Total	0.409559	0.006166
Frodo-1344	Geração de chaves	0.001796	0.000067
	Encapsulamento	0.001811	0.000065
	Desencapsulamento	0.001696	0.000017
	ZKPoP KeyGen	0.801847	0.004511
	ZKPoP Verify	0.754695	0.004198
	ZKPoP Total	0.778271	0.008709

Tabela 3. Resultados de desempenho em GO com N=65536 e TAU=8 — 100 execuções

Versão Frodo	Operação	Tempo Médio (s)	Desvio padrão (s)
Frodo-640	KeyPair	0.000535	0.000036
	Encaps	0.000588	0.000081
	Decaps	0.000560	0.000008
	ZKPoP KeyGen	159.483000	0.511570
	ZKPoP Verify	145.882621	0.515489
	ZKPoP Total	152.682811	0.7263

4.1. Análise Comparativa de Desempenho: C vs. Go

A Tabela 4 apresenta uma análise comparativa do desempenho, em termos de tempo médio de execução e desvio-padrão, entre as implementações de referência em C e uma implementação em Go para o KEM FrodoKEM e suas respectivas provas de conhecimento (ZKPoP). A análise revela um cenário de *trade-offs* de desempenho entre as duas linguagens.

De modo geral, a implementação em C demonstra ser mais performática na maioria das operações avaliadas. No entanto, a implementação em Go exibe vantagens pontuais, superando a versão em C em operações específicas, o que sugere otimizações direcionadas ou ainda limitações na medição, sendo operações muito rápidas nas duas linguagens.

Desempenho por Operação

- **Geração de Chaves (KeyPair):** A superioridade da implementação em C é consistente em todos os modelos. Contudo, a diferença de desempenho diminui à medida que o nível de segurança do esquema aumenta. Em Go, a operação foi 9,64% mais lenta para

Tabela 4. Comparação dos tempos médios e desvios-padrão entre implementações em C e Go para FrodoKEM

Modelo	Operação	Tempo médio (C)	Desvio Padrão (C)	Tempo médio (Go)	Desvio Padrão (Go)
Frodo-640	KeyPair	0.000473385	0.000007645	0.000519	0.000050
	Encaps	0.000572191	0.000047390	0.000570	0.000033
	Decaps	0.000539776	0.000013270	0.000536	0.000010
	ZKPoP Total	0.225807	0.001101	0.242642	0.002225
Frodo-976	KeyPair	0.000994658	0.000010958	0.001057	0.000088
	Encaps	0.001081776	0.000010558	0.001079	0.000041
	Decaps	0.001038351	0.000010784	0.000994	0.000014
	ZKPoP Total	0.416975	0.004025	0.409559	0.006166
Frodo-1344	KeyPair	0.001741351	0.000011767	0.001796	0.000067
	Encaps	0.001783777	0.000015288	0.001811	0.000065
	Decaps	0.001710627	0.000015530	0.001696	0.000017
	ZKPoP Total	0.715405	0.006302	0.778271	0.008709
Frodo-640 (N=65536) (TAU=8)	KeyPair	-	-	0.000535	0.000036
	Encaps	-	-	0.000588	0.000081
	Decaps	-	-	0.000560	0.000008
	ZKPoP Total	-	-	152.682811	0.7263

o Frodo-640, 6,27% para o Frodo-976 e apenas 3,14% para o Frodo-1344, mostrando que a lacuna de performance entre as linguagens se torna menos expressiva com o aumento dos níveis de segurança.

- **Encapsulamento (Encaps):** O desempenho nesta operação é extremamente competitivo. A versão em Go apresenta uma ligeira vantagem para os modelos de menor segurança, sendo 0,38% mais rápida no Frodo-640 e 0,26% no Frodo-976. Para o Frodo-1344, o C retoma a liderança com uma performance 1,53% superior.
- **Desencapsulamento (Decaps):** Esta é a operação em que a implementação em Go consistentemente supera a de C em todos os modelos. O ganho de desempenho de Go varia de 0,70% (Frodo-640 e Frodo-1344) a um expressivo 4,27% (Frodo-976).
- **Prova de Conhecimento (ZKPoP Total):** Para a prova de conhecimento, os resultados são mistos. Enquanto a implementação em C é mais rápida para os modelos Frodo-640 (+7,46% de tempo em Go) e Frodo-1344 (+8,79% em Go), a versão em Go se destaca no Frodo-976, completando a operação em um tempo 1,78% menor que a de C.

Análise do Cenário com Parâmetros Elevados (N=65536) A tabela também apresenta um caso de teste para o Frodo-640 em Go com parâmetros significativamente maiores (‘N=65536‘ e ‘TAU=8‘). Enquanto as operações KEM (‘KeyPair‘, ‘Encaps‘, ‘Decaps‘) mantiveram tempos de execução da mesma ordem de magnitude, o custo computacional da prova de conhecimento (‘ZKPoP Total‘) aumentou drasticamente. O tempo médio saltou para 152,68 segundos, um valor centenas de vezes maior que os 0,24 segundos da mesma operação no cenário padrão em Go. Este resultado evidencia que, embora o KEM em si escale de forma razoável, a prova de conhecimento associada possui um custo proibitivo com esses parâmetros, tornando-a inviável para aplicações práticas que exijam agilidade.

5. Conclusões

Neste trabalho foi realizada uma análise comparativa entre as implementações em C e Go para o FrodoKEM com Prova de Posse de chave privada. Enquanto a implementação em C reafirma sua proficiência em performance bruta, sendo consistentemente mais rápida na geração de chaves, a versão em Go demonstra uma notável competitividade, em especial no desencapsulamento. O achado mais expressivo, no entanto, é a drástica degradação de desempenho da prova de conhecimento (ZKPoP) em Go ao utilizar parâmetros de maior escala, evidenciando um severo desafio de escalabilidade. Como trabalhos futuros, pretende-se integrar as versões em Go no ACME e avaliar outros algoritmos do processo de padronização do NIST.

Agradecimentos

Os autores agradecem à UTFPR pelo apoio financeiro neste trabalho.

Referências

Agence Nationale de la Sécurité des Systèmes d'Information (2023). Anssi views on the post-quantum cryptography transition (2023 follow up). https://cyber.gouv.fr/sites/default/files/document/follow_up_position_paper_on_post_quantum_cryptography.pdf. Acesso em: 20 mar. 2025.

Barnes, R., Hoffman-Andrews, J., McCarney, D., and Kasten, J. (2019). Automatic Certificate Management Environment (ACME). RFC 8555, RFC Editor.

Chair for Security Engineering (2025). Kem-nizkpop repository. https://github.com/Chair-for-Security-Engineering/KEM-NIZKPoP.

Federal Office for Information Security (2023). Cryptographic mechanisms: recommendations and key lengths. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf. Acesso em: 12 mar. 2025.

FrodoKEM (2025). Frodokem: Post-quantum cryptography based on lattices. https://frodokem.org.

Güneysu, T., Hodges, P., Land, G., Ounsworth, M., Stebila, D., and Zaverucha, G. (2022). Proof-of-Possession for KEM Certificates Using Verifiable Generation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 1337–1351, New York, NY, USA. Association for Computing Machinery.

Microsoft Corporation (2025). Pqcrypto-lweke repository. https://github.com/microsoft/PQCrypto-LWEKE. Código-fonte oficial do FrodoKEM.

NIST (2022). PQC Standardization Process: Announcing four candidates to be standardized, plus fourth round candidates. https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4. Acessed: 2022-11-02.

Ramos, G. S., Esteves, A. F., and Giron, A. A. (2025). Repositório zkpop-go. https://github.com/PQC-Group-UTFPR/zkpop-go. Acesso em: 26 maio 2025.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134, Santa Fe, NM, USA. IEEE, IEEE.