NIST PQC 암호 최적화 구현 동향

엄시우*, 송민호*, 김상원*, 서화정**

*한성대학교 (대학원생)
**한성대학교 (부교수)

Trends in Optimized Implementations of NIST PQC Cryptography

Si-Woo Eum*, Min-Ho Song*, Sang-Won Kim*, Hwa-Jeong Seo*

*Hansung University(Graduate student)
**Hansung University(Professor)

요 약

양자컴퓨터의 빠른 발전은 현재의 안전하다고 알려진 공개키 암호 알고리즘을 위협하고 있다. NIST에서는 양자컴퓨터로부터 안전한 양자내성암호를 개발하고자 공모전을 개최하였으며, 2022년 4개의 최종 알고리즘이 선정되었다. 양자내성암호는 기존 암호 알고리즘에 비해 많은 암호화 시간을 필요로 하며 또한 많은 메모리를 요구한다. 이러한 문제로 인해, 암호화 시간을 최소화 하거나 제한된 환경에서도 활용할 수 있도록 메모리를 최소화 하는 방법등 여러 최적화 연구가 진행되고 있다. 본 논문에서는 최근 PQC 최적화 연구에 대해서 조사하여 소개하고 이를 통해 최신 PQC 최적화 구현 기법에 대해 확인한다.

I. 서론

현재 안전하다고 알려져 사용되고 있는 공개키 암호 알고리즘(RSA, ECDSA)은 양자컴퓨터의 발전으로 안정성이 위협받고 있다. 이로 인해 미국 국립표준기술연구소(National Institute of Standards and Technology, NIST)는 양자컴퓨터로부터 안전한 양자내성암호(Post Quantum Cryptography, PQC) 표준화를 위한 공모전을 개최하였다[1]. PQC 알고리즘은 기존에 사용되던 암호 알고리즘에 비해 연산 시간이 오래 걸리고 많은 메모리를 필요로 하는 특징이 있다. 따라서 연산 시간을 줄이고 메모리 적게 사용하기 위한 최적화 구현 연구가 활발하게 진행되고 있다.

본 논문에서는 NIST PQC 공모전에서 최종 선정된 암호 알고리즘에 대해 소개하고, 최신 최적화 구현 연구 동향을 알아본다.

II. NIST PQC 알고리즘

NIST PQC는 2015년 양자컴퓨터로부터 안전한 양자내성암호 개발을 위한 공모전을 개최하였다. 2022년 Crystals-Kyber, Falcon, Crystals-Dilithium, Sphincs+ 4개의 알고리즘이 최종 선정되었다[2-5].

Crystals-Kyber는 LWE의 변형인 Module RLWE를 기반으로 하는 KEM 알고리즘이다. 안전성에 따라서 kyber512, kyber768, kyber1024 3개의 파라미터 셋을 지원한다. 가장 낮은 안정성을 제공하는 kyber512를 기준으로 보았을 때, 800바이트의 공개키, 1632바이트의 비밀키 그리고 768바이트의 암호문 크기를 갖는다.

Crystals-Dilithium는 Kyber와 마찬가지로 LWE의 변형인 Module RLWE를 기반으로 하는 전자 서명 알고리즘이다. 안전성에 따라서 Dilithium2, Dilithium3, Dilithium5를 제공하고

있으며, 가장 낮은 안정성을 제공하는 Dilithium2의 경우 1,312바이트의 공개키, 2,528바이트의 비밀키 그리고 2,420바이트의 서명크기를 갖는다.

Falcon은 NTRU 문제에 기반한 전자 서명 알고리즘이다. 안정성에 따라서 Falcon-512, Falcon-1024를 지원하며, Falcon-512의 경우 897바이트의 공개키, 1,281바이트의 비밀키 그리고 752바이트의 서명 크기를 갖는다.

Sphincs+는 해시 트리 기반의 전자 서명 알고리즘이다. Sphincs+는 내부에서 사용되는 해시 알고리즘에 따라서 Sphincs+-sha2, Sphincs+shake로 나뉘며 각각 안전성에 따라서 128, 192, 256를 지원한다. Sphincs+-sha2-128의 경우 32바이트의 공개키, 64바이트의 비밀키 그리고 17,088바이트의 서명 크기를 갖는다.

III. NIST PQC 최적화 동향

PQC 알고리즘 최적화를 위한 전략으로는 PQC 알고리즘 내부 연산중 많은 비중을 차지하고 있는 다항식 곱셈을 최적화하기 위해 NTT와 같은 다항식 곱셈을 최적화하는 연구가 진행되어 왔다[6-8]. 최근에는 NTT 최적화 뿐만 아니라 AES 그리고 Keccak 같은 PQC 알고리즘 내부에서 사용되는 기존 암호 알고리즘을 최적화하는 연구가 진행되고 있다.

3-1. Crystals-Kyber

Xinyi Ji. et al.[9]은 GPU를 활용하여 Kyber 최적화 구현인 HI-Kyber를 소개하고 있다. HI-Kyber는 Kyber 알고리즘의 핵심 연산인 NTT, INTT, SHA3, Encode, Decode 등에 대해 전역 메모리 접근 횟수를 최소화하고 전체 알고리즘의 성능을 향상 시키는 효율적인 커널 융합 스킴을 제안하고 있다. 또한 핵심 연산중 하나인 NTT 연산의 병목 현상을 해결하기 위해 SLM, SDFS-NTT, EDFS-NTT 세가지 계산 체계를 제안하고 잇다. SLM(Sliced Layer Merging) 체계는 NTT 계수의 여러 레이어를 재사용함으로써 글로벌 메모리의 메모리 접근 횟수를 줄이는 것을 목표로 하는 체계이다. SDFS-NTT(Sliced depth-first search NTT)와 EDFS-NTT(Entire depth-first search NTT)는 깊이 우선 탐색 기법을 활용하는 새로운 접근 방식은 제안하였다. 효율적인 연산을 위해 데이터를 NTT 계수의 연속적인 부분을 포함하는 슬라이스로 나누고 병렬로 처리할 수 있도록 구성하였다. 이를 통해 메모리 접근은 최소화하고 레지스터의 활용을 최대화하여 성능 향상을 보여주고 있다. 이러한 기법을 통해 결과적으로 각각의 계산 체계는 일반적인 구현에 비해 7.5%, 28.5% 41.6%의 성능 향상을 보여주고 있다. 또한 전체적인 성능은 기존의 GPU 최적화 구현 대비 3.52배 높은 성능을 보여주고 있으며, AI tensor core를 활용한 구현의 성능 대비 1.78배 높은 성능을 보여주었다.

Lipeng Wan. et al.[10]은 고성능 암호 컴퓨팅에 AI 가속기를 활용하는 것을 연구하였다. 본래 AI 가속기는 기계 학습 또는 신경망을 위해 사용되는 것이 목표이다. 하지만 해당 연구에서는 암호 작업을 AI 가속기를 활용할 수 있게끔 변환하여 정확성을 보장하면서 성능 향상을 가져오는 것을 목표로 하였다. 격자 기반 암호화에서 가장 많은 시간이 소요되는 다항식 곱셈을 가속화하였고, 이를 위해서 NTT box 모듈을 제안하고 있다. NTT box는 AI 가속기의 행렬 곱셈 기능을 활용하여 다항식 곱셈을 효율적으로 계산하는데 사용된다. 결과적으로 AI 가속기를 활용한 구현은 기존의 동일한 GPU 플랫폼에서의 최적화 구현 대비 6.47배의 성능 향상을 달성하였고, 또한 NIST에 제출된 Kyber의 AVX2 구현 대비 각 단계(키 생성, 캡슐화, 디캡슐화)에서 26배, 36배 그리고 35배의 성능 향상 결과를 보여준다. 해당 연구는 AI 가속기의 활용 가능성을 보여주고 있으며 또한 높은 성능 향상을 가져올 수 있음을 보여주고 있다.

3-2. Crystals-Dilithium

J. Huang. et al.[11]는 ARMv7-M에서 기존의 Keccak 최적화 구현을 분석하고 기존의 Keccak 구현을 개선하여 최적화 구현을 하였다. Keccak의 최적화 구현을 위해 Pipelining 메모리 접근 최적화와 Lazy rotations 최적화 기법을 제안하였다. Pipelining 메모리 접근 최적화는 Cortex-M3에서 메모리 로드 명령어(ldr)와 연산 명령어를 교대로 사용하는 것이 아니라 메모리 로드 명령어를 그룹화하여 한번에 실행하고 연산 명령어가 실행하도록 함으로써 Pipeline 지연을 최소화 하는 방법이다. Lazy rotation은 인라인 배럴 시프터를 활용하며 rotations 과정을 미루고 생략함으로써 최적화를 하는 방법이다. 결과적으로 이러한 기법을 통해 ARMv7-M에서 12.84%의 성능 향상을 보여주었으며, 이외에도 NTT 연산 최적화도 제안하였다.

3-3. Falcon

Duc Tri Nguyen and kris Gaj[12]는 ARMv8 상에서 Falcon의 최적화 구현을 연구하였다. 해당 연구에서는 서명 생성을 위해 벡터화된 확장 가능한 FFT 구현을 개발하였다. 해당 구현은 5레벨 이상의 모든 FFT에서 활용 가능하다. 함수 호출 오버헤드를 최소화하기 위해 재귀적 접근법 대신 반복적 접근번으로 구현을 하였으며, 불규칙한 버터플라이 패턴에 대한 스케줄링 오버헤드를 최소화하였다. 또한 Twiddle factor 테이블 크기를 16KB에서 4KB로 줄임으로써 저장 공간이나 메모리가 제한된 장치에서도 활용될 수 있는 점을 강조하였다. 결과적으로 이러한 최적화 기법을 적용함으로써 서명 생성 과정은 기존 최적화 구현 성능 대비 서명과 검증 과정에서 약 1.5배 정도의 성능 향상 결과를 보여주고 있다. 또한, Dilithium 대비 서명 생성 과정은 2배 느리지만, 서명 검증 과정은 3-3.9배 빠른 성능을 보여준다. 서명 생성 과정에서는 더 느린 성능을 보여주지만, 서명 검증 과정에서 더 빠른 성능을 보여주기 때문에 컴퓨팅 성능이 서버보다는 제한된 클라이언트에서는 보다 효율적으로 활용될 수 있음을 보여주고 있다.

3-4. Sphincs+

D. C. Kim. et al.[13]은 GPU의 높은 병렬 연산 능력을 활용하여 Sphincs+를 병렬 최적화 구현을 하였다. Sphincs+의 서명 과정의 분석을 통해 FORS, WOTS+ 그리고 MSS 연산 과정이 GPU 환경에서의 병렬 가능성을 발견하고 병렬화 하였다. 또한 Hypertree 연산 과정에서 MSS 구성과 WOTS+ 서명이 병렬로 연산 가능함을 발견하였고, 이를 위해서 Hypertree의 실행 순서를 변경하여 모든 MSS 구성을 먼저 계산하고 모든 WOTS+ 서명을 계산하는 것을 병렬로 실행함으로써 처리량을 최대화하였다. 이와 동시에 CUDA의 이점을 활용하여 GPU 성능을 최대로 사용하였다. 예를들어 글로벌 메모리에 비해서 낮은 Latency로 사용 가능한 공유 메모리를 사용하고, 이때 공유 메모리의 성능을 최대화 하기 위한 Bank 충돌 최소화 접근법을 사용하였다. Bank 충돌 최소화를 위해서 공유 메모리의 짝수 단위의 메모리 영역을 사용할 때는 홀수 단위의 메모리 영역에 결과를 저장하고, 반대로 홀수 단위의 메모리 영역의 값을 사용할 때는 결과값을 짝수 단위의 메모리 영역에 저장함으로써 Bank 충돌을 최소화 하였다. 결과적으로 Sphincs+의 서명 생성 과정에서 AVX2 구현 성능 대비 1,100배 이상의 놀라운 성능 향상 결과를 보여주고 있다. 또한 해당 연구에서 활용한 GPU보다 높은 사양의 GPU를 활용한 기존의 성능 결과 대비 서명 생성 과정에서 2배, 키 생성에서 1.03배 그리고 서명 검증 과정에서 9배 높은 성능 향상 결과를 보여주고 있다.

IV. 결론

본 논문에서는 2022년 NIST PQC 공모전에서 최종 선정된 4개의 PQC 알고리즘의 최신 최적화 구현에 대해서 조사하였다. 기존의 PQC 알고리즘의 최적화 구현은 다항식 곱셈을 최적화하기 위한 NTT의 최적화와 같이 특정 플랫폼에서 NTT 최적화 구현 연구에 취중되어 있었다면, 최근의 최적화 구현 연구는 NTT 최적화 구현 연구와 함께 Keccak과 같은 내부에서 활용되는 해시 함수의 최적화, AI 가속기의 활용 그리고 GPU의 높은 병렬 연산 기능 활용 등 여러 기술과 플랫폼을 활용하는 연구가 진행되고 있는 것을 확인하였다. 이러한 연구 결과는 국내에서 진행되고 있는 KPQC 공모전에 제출된 알고리즘과 Additional Digital Signature 공모전에 제출된 알고리즘에 적용함으로써 여러 PQC 알고리즘의 성능 향상에 이바지 될 수 있으며, 또다른 새로운 연구 방향을 제시할 수 있을 것으로 기대된다.

[참고문헌]

[1] The US National Institute of Standards and Technology. Post-Quantum Cryptography Standardization Project. Accessed: Oct. 19. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography

[2] Bos, Joppe, et al. “CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM,” 2018 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2018.

[3] Fouque, Pierre-Alain, et al. “Falcon: Fast-Fourier lattice-based compact signatures over NTRU,” Submission to the NIST’s post-quantum cryptography standardization process36.5 (2018): 1-75. 2018.

[4] L. Ducas, et al. “Crystals-dilithium: A lattice-based digital signature scheme,” IACR Transactions on Cryptographic Hardware and Embedded Systems(2018): 238-268. 2018.

[5] Bernstein, Daniel J., et al. “The SPHINCS+ signature framework,” Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. 2019.

[6] SEILER, Gregor. “Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography,” Cryptology ePrint Archive, 2018.

[7] D. T. Nguyen and G. Kris. “Optimized software implementations of CRYSTALS-Kyber, NTRU, and Saber using NEON-based special instructions of ARMv8,” Proceedings of the NIST 3rd PQC Standardization Conference (NIST PQC 2021). 2021.

[8] Y. B. Kim, J. G. Song and S. C. Seo. “Accelerating falcon on ARMv8,” IEEE Access 10 (2022): 44446-44460. 2022.

[9] X. Y. Ji, et al. “HI-Kyber: A novel high-performance implementation scheme of Kyber based on GPU,” IEEE Transactions on Parallel and Distributed Systems, 2024.

[10] L. Wan, et al. “A novel high-performance implementation of CRYSTALS-Kyber with AI accelerator,” European Symposium on Research in Computer Security. Cham: Springer Nature Switzerland, 2022.

[11] J. H. Huang, et al. “Revisiting Keccak and Dilithium Implementations on ARMv7-M,” IACR Transactions on Cryptographic Hardware and Embedded Systems(2024): 1-24. 2024.

[12] D. T. Nguyen and G. Kris. “Fast falcon signature generation and verification using armv8 neon instructions,” International Conference on Cryptology in Africa. Cham: Springer Nature Switzerland, pp. 417-441, 2023.

[13] D. C. Kim, H. J. Choi, and S. C. Seo. “Parallel Implementation of SPHINCS + With GPUs,” IEEE Transactions on Circuits and Systems I: Regular Papers. pp 1-4, 2024.