ARMv8 상에서의 격자 기반 암호 최적 구현 연구 동향

이민우* 심민주* 엄시우* 송경주* 윤세영** 서화정***

*한성대학교 정보컴퓨터공학과(대학원생)

**한성대학교 융합보안학과(대학원생)

***한성대학교 융합보안학과(부교수)

Implementation Trends of Lattice-Based Cryptography on ARMv8 Architectures

Min-Woo Lee* Min-Joo Sim* Si-Woo Eum Gyeong-Ju Song* Sae-Young Yoon** Hwa-Jeong Seo***

*Hansung University(Graduate student)

**Hansung University(Graduate student)

***Hansung University(Associate Professor)

요 약

양자 컴퓨팅의 발전으로 기존 공개키 암호체계의 안전성에 대한 우려가 커지면서, 양자 내성 암호의 중요성이 부각되고 있다. 특히 격자 기반 암호는 안정성과 연산 효율성 측면에서 유력한 대안으로 평가받고 있으며, 고속 다항식 연산을 위한 NTT가 핵심 기술로 활용된다. 본 논문에서는 ARMv8 아키텍처 환경에서 격자 기반 암호 알고리즘의 최적 구현 사례를 정리하고, 주요 연산인 NTT의 병목 해소와 병렬화 기법을 중심으로 최신 연구 동향을 분석하였다. 이를 위해 Crystals-Dilithium, HAETAE, Falcon, HAWK 알고리즘의 ARMv8 기반 구현 사례를 중심으로, 각 알고리즘에서 적용된 최적화 기법과 성능 개선 결과를 비교·분석하였다.

I. 서론

양자 컴퓨터의 실현 가능성이 점차 높아지면서, 기존의 공개키 암호체계는 근본적인 보안 위협에 직면하고 있다. 특히 쇼어 알고리즘은 큰 소수의 곱셈 문제나 이산로그 문제를 효율적으로 해결할 수 있어, RSA나 타원곡선 암호 등 현행 암호체계를 무력화시킬 수 있다. 이러한 배경에서, 양자 컴퓨터 환경에서도 안전한 암호 기술을 확보하기 위한 새로운 암호 체계로 양자 내성 암호가 등장하였다.

양자 내성 암호 중에서도 격자 기반 방식은 수학적 구조의 안전성과 고속 연산이 가능한 특성 덕분에 가장 효율적인 후보군으로 평가받고 있다. 격자 기반 암호는 다항식 기반의 연산 구조를 가지고 있으며, 특히 정수론적 푸리에 변환(NTT)을 활용한 고속 다항식 곱셈 연산이 핵심 기술로 사용된다. 하지만 이러한 연산은 계산량이 많아, 연산 자원이 제한적인 임베디드 시스템이나 모바일 기기 환경에서는 효율적인 구현이 필수적이다. 이에 따라 낮은 전력 소비, 병렬 연산 명령어를 지원하는 ARMv8 아키텍처는 격자 기반 암호 구현의 주요 대상 플랫폼으로 주목받고 있다.

현재 ARMv8 환경에서의 격자 기반 암호 알고리즘 구현을 최적화하기 위한 다양한 연구가 이루어지고 있으며, 특히 NTT 연산과 모듈러 곱셈 등 성능 병목이 일어나는 부분을 중심으로 최적화 기법이 적용되고 있다. 본 논문에서는 이러한 구현 최적화 동향을 중심으로, 국내외 주요 알고리즘의 사례들을 정리하고

ARMv8 상에서의 효율적인 격자 기반 암호 구현 방안을 고찰한다.

본 논문의 구성은 다음과 같다. 2장에서는 ARMv8 아키텍처와 NTT에 대한 개요를 다룬다. 3장에서는 격자 기반 암호 알고리즘의 구현 사례 및 최적화 기법을 살펴보고, 4장에서는 결론을 제시한다.

II. 관련 기술

2.1 ARMv8 아키텍처

ARMv8은 고성능 임베디드 환경을 위한 64비트 명령어를 지원하는 아키텍처로, 2011년에 처음 발표되었다. 해당 아키텍처는 기존의 32비트 기반인 ARMv7과의 호환성을 유지하면서도, 확장된 레지스터 구성과 향상된 연산 성능을 제공하는 것이 특징이다. 특히 ARMv8은 64비트 실행 모드인 AArch64와 기존 32비트 모드인 AArch32를 모두 지원하여 다양한 응용 환경에 대응할 수 있다.

AArch64 모드에서는 총 31개의 범용 레지스터를 제공하며, 64비트 정수 연산을 지원한다. 또한 고속 병렬 처리를 위해 128비트 벡터 레지스터를 제공한다. 이러한 벡터 레지스터는 패킹 방식을 통해 8비트(B), 16비트(H), 32비트(S), 64비트(D) 단위로 데이터를 저장하고 병렬 연산할 수 있다. 이러한 구조는 정수형 덧셈, 곱셈, 비트 연산, 전치 연산 등을 벡터 단위로 처리하는 SIMD 연산에 최적화되어 있다[1].

ARMv8 아키텍처는 낮은 전력 소비 대비 높은 연산 성능을 제공하기 때문에, 스마트폰 및 태블릿은 물론, 최근에는 노트북과 경량 서버 기기에서도 널리 사용되고 있다. 특히 격자 기반 암호 알고리즘과 같이 연산량이 많은 구조를 효율적으로 구현하는 데 있어, ARMv8의 SIMD 명령어 집합은 연산 병목을 줄이는 데 효과적으로 활용된다.

2.2 NTT

NTT는 이산 푸리에 변환의 정수 연산 버전으로, 격자 기반 암호 알고리즘에서 다항식 곱셈의 고속화를 위한 핵심 연산으로 널리 사용된다[2]. 일반적인 다항식 곱셈은 $O(n^2)$의 시간 복잡도를 가지지만, NTT를 적용하면 이를 $O(nlogn)$으로 줄일 수 있다.

NTT는 정수 계수 다항식의 곱셈을 빠르게 수행하기 위한 알고리즘으로, 복소수 대신 정수만을 사용하는 환경에서 이산 푸리에 변환과 유사한 방식으로 작동한다. 이러한 연산이 가능하기 위해서는 특정 조건을 만족하는 소수 계수와 변환 크기 n이 필요하며, 특히 n차 거듭제곱 시 1이 되는 특수한 정수인 거듭제곱근의 존재가 필수적이다. 이 값은 변환 과정에서 다항식의 계수를 정수형 주파수 영역에 대응하는 값으로 바꾸는 역할을 수행한다. 이와 같은 구조를 이용하면, 복잡한 다항식 곱셈을 변환된 계수 간의 단순한 점별 곱셈(pointwise multiplication)으로 대체할 수 있으며, 이후 역변환을 통해 다시 시간 영역의 계수로 복원할 수 있다.

NTT의 연산 과정은 여러 단계의 버터플라이 연산으로 이루어지며, 각 단계는 두 계수의 합과 차를 계산하고, 그중 하나에 사전 계산된 거듭제곱근 값을 곱하는 방식으로 구성된다. 이러한 연산은 덧셈, 뺄셈, 모듈러 곱셈이 반복되기 때문에 계산량이 많고, 연산 병목이 발생하기 쉬운 구조를 갖는다.

이를 보완하기 위해 NTT 구현에는 Montgomery 곱셈이나 Barrett 리덕션과 같은 고속 모듈러 곱셈 기법이 적용되며, 이들 연산은 반복 구조를 갖기 때문에 SIMD 기반 병렬화에 적합하다. 실제 하드웨어 환경에서는 벡터 레지스터를 활용해 병렬 연산을 수행함으로써 루프 반복 횟수를 줄이고, 명령어 병렬도를 높여 처리 속도를 향상시킬 수 있다.

III. 연구 동향

3.1 Crystals-Dilithium on ARMv8

해당 연구는 Crystals-Dilithium의 NTT, INTT, 점별 곱셈 등 NTT 기반 다항식 연산의 병목을 제거하는 데 집중하였으며, ARM NEON 명령어를 활용한 SIMD 병렬화를 통해 계산 속도를 향상시켰다[3]. 성능 향상을 위해

적용된 핵심 전략 중 하나는 register holding 기법이다. 자주 사용되는 중간 결과값을 메모리에 저장하지 않고 레지스터 내에 유지함으로써, 불필요한 메모리 접근을 줄이고 데이터 이동 오버헤드를 감소시켰다. 또한 NTT 연산 내에서의 덧셈, 곱셈, 모듈러 연산 등 연속되는 연산 흐름을 병합하여, 이를 하나의 NEON 명령어 시퀀스로 구성하는 merging 기법을 적용하였다. 여기에 더해, ARM 기본 명령어와 NEON 명령어의 교차 사용을 통해 레지스터 자원을 효율적으로 분배하고, 벡터 유닛 사용률을 극대화하는 최적화를 수행하였다.

NTT 연산의 병렬화는 특히 점별 곱셈 단계에서 뚜렷한 효과를 보였다. 연구진은 벡터 레지스터에 다항식 계수를 16비트 단위로 할당하고, 동일한 twiddle factor를 공유하는 계수 그룹을 정렬함으로써, 버터플라이 연산을 병렬로 수행하였다. 이 과정에서 NEON의 128비트 레지스터를 적극 활용하여, 한 번의 명령어로 여러 항의 곱셈을 동시에 처리할 수 있도록 구성하였다.

성능 측정 결과, NTT 연산 시간은 최대 260%까지 향상되었으며, 전체 알고리즘의 주요 단계에서도 의미 있는 개선이 나타났다. 키 생성(KeyGen)은 약 43.8%, 서명(Sign)은 113.2%, 검증(Verify)은 41.9% 성능 향상을 기록하였다. 코드 크기 측면에서는 소폭 증가가 있었으나, 메모리 접근 횟수와 전체 연산 비용은 줄어들었으며, 전반적인 자원 효율은 개선되었다.

3.2 HAETAE on ARMv8

HAETAE는 KpqC 공모전에서 최종 표준 알고리즘으로 선정된 격자 기반 전자 서명 알고리즘이다. 해당 연구는 Apple M1 프로세서를 포함한 ARMv8 기반 플랫폼을 대상으로 하며, NTT 및 역변환, Montgomery 곱셈 등 HAETAE 내부의 핵심 연산에 대해 고급 최적화 기법을 적용하였다. 구현에는 ARM NEON SIMD 명령어와 128비트 벡터 레지스터를 활용하였으며, 기존 참조 구현 대비 상당한 성능 향상을 달성하였다[4].

성능 개선의 핵심은 NEON 기반의 병렬 처리 구조에 있다. 다항식 곱셈에 필요한 계수들을 128비트 벡터 레지스터에 32비트 단위로 할당하여 병렬 연산을 수행하고, 이를 통해 단일 명령어로 4개의 연산을 동시에 처리하였다. 특히 SQDMULH, SHSUB, SMULL, ZIP1, TRN1 등 NEON 고속 명령어를 조합하여 NTT의 버터플라이 연산을 병렬화하였다. 여기에 데이터를 미리 재배치하는 데이터 재정렬 기법을 적용하여 동일한 루트 값을 공유하는 항목들을 인접하게 배치함으로써 병렬 처리 효율을 더욱 높였다.

또한, INTT 연산에 필요한 루트 테이블을 사전 계산하여 저장하는 방식으로 계산 시간을 단축하였으며, 실제 구현에서는 계수의 순서를 재배치하는 전처리 과정을 통해 마스킹의 효율성을 높였다.

성능 측정 결과는 다음과 같다. 논문에 따르면, NTT는 최대 3.07배, INTT는 3.63배, Montgomery 곱셈은 9.15배의 성능 개선이 이루어졌다. 이러한 개선은 최종적으로 전체 알고리즘의 주요 단계인 키 생성, 서명, 검증 단계의 처리 속도 향상으로 이어졌다. HAETAE-120 기준으로는 키 생성 단계에서 1.16배, 서명 생성 단계에서 1.12배, 서명 검증 단계에서 1.27배의 성능 개선이 이루어졌으며, 보다 높은 보안 수준을 갖는 HAETAE-260의 경우에도 전체적으로 최대 1.25배의 향상이 확인되었다.

3.3 Fast Falcon Signature Generation and Verification Using Armv8 NEON Instructions

Falcon은 CRYSTALS-Dilithium과 함께 NIST PQC 디지털 서명 알고리즘의 최종 후보로 선정된 격자 기반 알고리즘 중 하나이다. Falcon의 서명 연산은 FFT를 활용한 다항식 곱셈이 핵심 연산으로 작용하며, 해당 구현에서는 레벨 5 이상의 FFT에 적용 가능한 범용 벡터화 구조를 설계하였다. 또한 twiddle factor 테이블을 16KB에서 4KB로 압축하는 방식으로 메모리 사용량을 절감하였다. 이 압축은 켤레근(conjugate root) 기반의 대칭 구조를 활용한 것이며, 플랫폼에 종속되지 않도록 설계되었다.

검증 연산에서는 NTT를 활용한 다항식 곱셈이 중심이며, Cooley-Tukey 구조를 Forward NTT에, Gentleman-Sande 구조를 Inverse NTT에 적용하였다. 모듈러 곱셈 과정에서는 Barrett 리덕션과 Montgomery 곱셈이 모두 사용되었으며, 특히 Montgomery 연산에서는 계수 n^{-1}을 역변환 이전 단계에 미리 곱하는 방식으로 불필요한 연산을 제거하는 최적화가 적용되었다. 이는 계산 단계에서의 모듈러 리덕션 횟수를 줄이고, 부호 있는 정수 연산 명령어를 활용하여 전체적인 실행 효율을 높이는 효과를 보였다[5].

성능 측면에서 Falcon의 서명 생성은 기존 구현 대비 최대 1.42배(NEON 기준) 향상되었으며, 검증 속도는 Dilithium 대비 평균 3.0~3.9배 빠른 결과를 기록하였다. 다만 Falcon은 64비트 부동소수점 연산과 직렬 샘플링, 직렬 해싱 구조를 포함하고 있어, 32비트 정수 연산과 병렬 해싱이 가능한 Dilithium에 비해 서명 생성에서의 개선 폭은 제한적이다. 반면, 검증 연산에서는 구조적으로 하나의 다항식 곱셈만 수행하므로 NEON 기반 최적화가 더 효과적으로 작동하였다.

3.4 Optimizing HAWK Signature Scheme Performance on ARMv8

HAWK는 NIST 추가 디지털 서명 알고리즘 표준화 공모전에서 현재 라운드 2에 진출한 격자 기반 전자 서명 알고리즘이다. 본 절에서는 HAWK 알고리즘을 ARMv8 아키텍처 기반에서 최적 구현한 사례를 소개한다.

해당 연구는 Apple M1 프로세서를 대상으로 하며, 전반적인 연산 흐름에 대한 프로파일링 분석을 통해 병목 구간을 식별한 뒤, 최적 구현을 진행하였다. 전반적인 연산 흐름에 대한 프로파일링 분석을 수행하여 병목 지점을 식별한 결과 결과, 전체 연산의 98%가 키 생성 단계에서 발생함이 확인되었다. 특히 NTRUSolve(66.9%)와 Rebuild_CRT(27.9%) 함수가 주요 병목으로 확인되었다. 해당 병목 함수 내부의 핵심 연산인 mod_small_unsigned()와 add_mul_small() 함수에 대해 ARMv8의 sbfx와 csel, madd 등의 명령어를 활용한 최적화를 수행하였다. 해당 명령어는 조건 분기 없이 조건 선택이나 비트 필드를 효율적으로 처리할 수 있다. 따라서 분기 예측 실패에 의한 성능 저하를 방지할 수 있고, 명령어 수 또한 간소화할 수 있다.

NTT 연산에 대해서는 병렬화 최적화 기법이 적용되었다. 구체적으로는 다항식 차수 n의 로그 값인 $\log n$에 따라 병렬 처리의 성능 영향이 달라진다는 점에 주목하였다. $\log n$ 값이 작을 경우 오히려 병렬화 시 성능 저하를 유발함을 확인하였으나, $\log n \geq 8$ 수준부터는 병렬화가 유의미한 속도 향상을 보였다. 따라서 $\log n$의 값에 따라 병렬화 여부를 조건적으로 선택하는 방식으로 구현을 진행하였다[6].

성능 측정 결과, HAWK-512 기준으로 전체 처리 속도에서 약 2.5%, HAWK-1024에서는 약 4.0%의 성능 향상이 관측되었다. 특히 HAWK-1024의 키 생성 단계에서는 약 194만 사이클이 감소했으며, 이는 키 생성 전체 시간의 약 4%에 해당하는 절감 효과임을 확인하였다.

IV. 결론

본 논문에서는 ARMv8 아키텍처 상에서 격자 기반 암호 알고리즘을 효율적으로 구현하기 위한 최적화 동향을 정리하였다. 특히 NTT를 중심으로 병목이 발생하는 연산에 대해, SIMD 명령어를 활용한 병렬화 기법과 연산 구조 최적화 방식을 살펴보았다. 사례별 최적화 방식은 다르지만, 공통적으로 벡터 연산 기능을 적극 활용하고, 연산 흐름을 구조적으로 단순화하여 병목을 해소한 점이 특징적이다. 향후에는 다양한 알고리즘에 대해 플랫폼별 특성을 고려한 맞춤형 최적화가 지속적으로 요구될 것으로 보인다.

V. Acknowledgment

This work was partly supported by Institute of Information & communications

Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2025-02306395, Development and Demonstration of PQC-Based Joint Certificate PKI Technology, 50%) and this work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services, 50%).

[참고문헌]

[1] Arm® A64 Instruction Set Architecture:Armv8, for Armv8-A Architecture Profile.Available online: https://developer.arm.com/Architectures/A64%20Instruction%20Set%20Architecture (accessed on 1 May 2025)

[2] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," Proceedings of the International Conference on Cryptology and Network Security, pp.124 - 139, 2016.

[3] Y. Kim, J. Song, T.-Y. Youn, and S. C. Seo, "Crystals-Dilithium on ARMv8," Security and Communication Networks, vol. 2022, Article ID 5226390, pp. 1 - 16, 2022.

[4] M. Sim, M. Lee, and H. Seo, "HAETAE on ARMv8," Electronics, Vol.13, No.19, pp.3863, 2024.

[5] D. T. Nguyen and K. Gaj, "Fast Falcon Signature Generation and Verification Using ARMv8 NEON Instructions," Proceedings of the International Conference on Cryptology in Africa (AFRICACRYPT), pp.417 - 441, 2023.

[6] S. Eum, M. Lee, and H. Seo, "Optimizing HAWK Signature Scheme Performance on ARMv8," Applied Sciences, Vol.14, No.19, pp.8647, 2024.