고차원 Radix Winograd BFU 를 이용한 격자 기반 암호 연산 가속 가능성 연구

조 상 우, 김 영 식
대구경북과학기술원 전기전자컴퓨터공학과
acxz6469@dgist.ac.kr, ysk@dgist.ac.kr

A Study on the Acceleration Potential of Lattice-Based Cryptographic Operations Using High-Radix Winograd BFU

Sang-Woo Cho, Young-Sik Kim
Department of Electrical Engineering and Computer Science, DGIST

요 약

격자 기반 암호에서 핵심 연산인 다항식 곱셈은 Number Theoretic Transform(NTT)을 통해 $O(nlogn)$복잡도로 감소된다. 본 논문에서는 전통적 Radix-2 Cooley-Tukey Butterfly Unit 대신 Winograd 기법을 적용한 Radix-8/16 BFU 구조를 제안하고, 이론적 연산량 분석 및 소프트웨어 환경에서의 정량적 성능 평가를 통해 각 기법의 장단점을 비교하였다. Intel Core i9+Python+gmpy2(single-thread) 환경에서 100,000 회 반복 벤치마크한 결과, 차수 N=8 인 경우 Radix-8 Winograd BFU 는 기존 Radix-2 대비 약 26 %(6.61 μs→4.88 μs)의 처리 시간 단축을 보였으나, N=16 인 경우 Radix-16 은 약 14 % 오버헤드(18.29 μs→20.83 μs)로 성능이 저하되었다. 이는 Radix 차수가 증가할수록 모듈러 곱셈 절감 효과보다 덧셈·뺄셈 오버헤드가 커지기 때문임을 시사한다. 향후 FPGA 상 구현을 기반으로 성능 향상을 분석하고, Radix-32 이상 고차 Winograd BFU 확장 등을 통해 실용적 유효성을 더욱 심층 평가할 예정이다.

Ⅰ. 서 론

양자 계산의 위협에 대비하기 위한 포스트-양자 암호(Post-Quantum Cryptography, PQC) 표준화 과정에서, 미국 국가표준기술연구소(NIST)는 코드 기반(code-based) 암호와 격자 기반(lattice-based) 암호를 최종 후보로 선정하였다. PQC 는 양자 공격에도 안전한 차세대 암호 기술로, 향후 국가 차원의 암호 인프라 전면 교체를 이끄는 핵심 동력이 될 중요한 요소이다. 이 중 격자 기반 암호에서는

$$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$

상의 다항식 곱셈이 핵심 연산으로 자리 잡고 있으며, 이 연산은 Number Theoretic Transform(NTT)을 통해 $O(nlogn)$의 복잡도로 구현된다. 전통적으로 NTT 하드웨어 구현은 Radix-2 Cooley-Tukey(CT) 와 Gentleman-Sande(GS) butterfly 구조를 채택하여 모듈러 덧셈·곱셈 연산을 반복 수행해 왔으나, 더 빠른 처리 속도에 대한 논의가 필요하다. 따라서 본 논문에서는 하드웨어 중심으로 설계된 Radix-8/16 Winograd BFU 구조를 중심으로, 고차 Radix NTT 가 갖는 하드웨어 설계 관점의 가속 잠재력을 이론적 연산량을 기반으로 분석한다. 또한 이 과정을 소프트웨어에 접목시켜 소프트웨어상 연산 가속에 대해 분석한다.

Ⅱ. 본론

Winograd[1]은 DFT 에서 곱셈 회로 수를 획기적으로 줄이는 기법을 제시하였다. Mandal and Roy[2]은 이 기법을 NTT 에 확장 적용하여 FPGA 상에서 Radix-8 및 Radix-16 Winograd BFU 를 설계·구현하고, 이를 CRYSTALS-Kyber, CRYSTALS-Dilithium, FALCON 등에 통합하여 뛰어난 성능을 입증하였다.

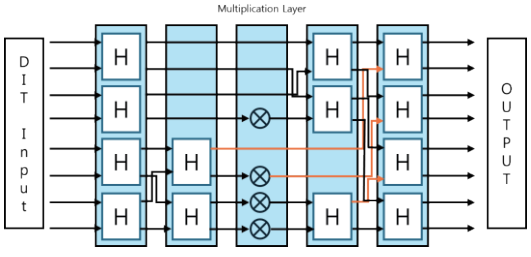

그림 1 Radix-8 Winograd BFU

그림 1 은 Winograd 기법을 적용한 Radix-8 BFU 구조를 나타낸다. 'H'로 표기된 Hadamard Unit 은 두 입력 a, b 에 대해 $(a+b) mod\ q, (a-b)\ mod\ q$ 를 동시에 출력하는 연산 유닛이다. 전통적인 Radix-2 CT Butterfly Unit 은 길이 N NTT 에서 총 $\frac{N}{2}\log_2 N$회의 모듈러 곱셈을 요구하지만, Winograd BFU 는 단일 Multiplication Layer 에서 4 회의 모듈러 곱셈만 수행하도록 설계되어 곱셈 횟수를 12 회에서 4 회로 대폭 감소시킨다. 하드웨어 관점에서 모듈러 곱셈은 덧셈 대비 훨씬 많은 연산 시간을 소요하므로, 이 Multiplication Layer 를 병렬화하면 전체

처리 속도를 크게 향상시킬 수 있다.

아래 표는 소프트웨어 환경에서 차수 N 다항식에 대해 Radix-2(CT BFU), Radix-8, Radix-16 (Winograd BFU) 기반 NTT 의 BFU 실행 시간을 비교한 결과이다. 성능 평가는 다음과 같은 환경에서 수행되었다.

- **하드웨어:** Intel Core i9-14900K (24 코어/32 스레드, 3.20 GHz 기본·5.80 GHz 최대), ASUS PRIME Z790-P, 32 GB DDR5
- **소프트웨어:** Microsoft Windows 11 Home (10.0.22621), Python 3.10.0rc2 (CPython 64-bit), gmpy2 (모듈러 연산 가속)
- **구현:** 단일 프로세스(single-threaded) 방식

	Radix-2	Radix-8	Radix-16
N=8	6.61	4.88	
N=16	18.29		20.83

시행횟수: 100,000, 단위: μs

Ⅲ. 결론

본 연구에서는 Winograd BFU 를 이용하여 Radix-8, Radix-16 NTT 를 기존 Radix-2 CT BFU 와 비교 평가하였다. 100,000 회 반복 측정한 결과, N=8 일 때 Radix-2 가 6.61 μs 의 처리 시간을 기록한 반면, Winograd BFU 를 적용한 Radix-8 은 4.88 μs 로 약 26 %의 성능 향상을 보였다. 그러나 N=16 환경에서는 Radix-2 가 18.29 μs 였던 데 비해 Radix-16 은 20.83 μs 로 오히려 약 14 % 더 느려졌다.

이러한 결과는 Winograd BFU 가 Radix 크기가 커질수록 모듈러 곱셈 횟수는 줄어들지만, 덧셈 · 뺄셈 연산이 급격히 증가해 단일 스레드 소프트웨어 구현에서는 오히려 오버헤드를 유발함을 시사한다. 반면, FPGA 와 같은 환경에서는 데이터 전송 시 통신 비용이 거의 없으므로, Radix-16 이상의 고차 NTT 에도 Winograd BFU 가 실질적인 성능 이점을 제공할 수 있을 것으로 기대된다.

본 연구의 주요 기여는 소프트웨어와 하드웨어 구현 관점에서 Winograd 기법의 장단점을 종합적으로 분석했다는 점이다. 이 과정에서 하드웨어 중심적으로 설계된 Winograd BFU 가 특정 조건 하에서 소프트웨어상에서의 성능 향상도 관찰할 수 있었던 점은 의의가 크다. 향후 연구에서는 GPU, SIMD 레지스터, 멀티코어 등의 기법을 활용하여 소프트웨어 상 성능 향상을 논의한다. 또한 Radix-32 이상의 Winograd BFU 확장 적용 가능성에 대해 연구하고, FPGA 상의 성능 향상 가능성을 검증할 필요가 있다.

ACKNOWLEDGMENT

이 논문은 2024 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(RS-2024-00399401, 양자안전 보안인프라 전환 및 대양자 복합 안전성 검증기술 개발).

참 고 문 헌

[1] S. Winograd, “On computing the discrete Fourier transform,” Mathematics of Computation, vol. 32, no. 141, pp. 175-199, Jan. 1978.

[2] S. Mandal and D. B. Roy, “Winograd for NTT: A case study on higher-radix and low-latency implementation of NTT for post quantum cryptography on FPGA,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 71, no. 12, pp. 6396-6409, Dec. 2024, doi:10.1109/TCSI.2024.3470335.