논문 20-45-12-24 The Journal of Korean Institute of Communications and Information Sciences '20-12 Vol.45 No.12
https://doi.org/10.7840/kics.2020.45.12.2183

GPU를 이용한 LWE 기반 양자 내성 암호의 병렬화 및 성능 분석

김 예 원◆°, 염 용 진*, 강 주 성*

Parallel Processing and Performance Analysis of LWE-Based Post-Quantum Cryptography Using GPU

Yewon Kim◆°, Yongjin Yeom*, Ju-Sung Kang*

요 약

양자 컴퓨팅 기술을 이용한 공격에서도 안전성을 보장받을 수 있는 양자 내성 암호에 관한 연구가 활발하게 진행되고 있다. 미국 NIST는 양자 내성 암호 알고리즘의 표준화에 대한 프로젝트를 진행하고 있으며, 2020년 7월에 15개의 3라운드 양자 내성 암호 알고리즘을 발표하였다. 이 중 7개의 알고리즘은 최종 알고리즘이며, 3라운드 종료 후 진행될 1차 표준화 대상이다. 또한, 8개의 알고리즘은 대체 후보 알고리즘이며, 안전성과 성능 분석이 추가적으로 필요한 2차 표준화 대상이다. 따라서 본 논문은 다양한 환경에서의 성능 분석이 필요한 3라운드 대체 후보 알고리즘인 LWE 문제 기반인 FrodoKEM의 성능을 분석한다. FrodoKEM의 알고리즘에 대한 분석을 통해 FrodoKEM의 알고리즘을 구성하는 6가지의 주요 단계를 선정한다. 그리고 PC와 NIST가 제시한 Cortex-M4보다 성능이 좋은 Cortex-A53 코어 보드에서 주요 단계별 소요시간을 측정한 결과를 제시한다. 또한, 소요 시간이 가장 긴 2가지의 단계를 GPU를 이용하여 병렬화하였을 때의 성능을 제시하고, 병렬화된 단계가 적용된 FrodoKEM의 성능을 제시한다.

키워드 : 양자 내성 암호, 격자 기반 양자 내성 암호, 키 설정 메커니즘, FrodoKEM, 병렬 구현
Key Words : Post-Quantum Cryptography, Lattice based Post-Quantum Cryptography, Key Establishment Mechanism(KEM), FrodoKEM, Parallel Implementation

ABSTRACT

Research on post-quantum cryptography(PQC) that can guarantee the security even in attacks using quantum computing technology has been actively conducted. NIST has been working on a project on the standardization of post-quantum cryptographic algorithms, and announced 15 third round candidate PQC algorithms in July 2020. Seven of these algorithms are the finalists and subject to primary standardization after the end of the 3 round. In addition, eight algorithms are alternate candidates and subject to secondary standardization requiring additional security and performance analysis. This paper analyzes the performance of FrodoKEM, which is based on learning with errors(LWE) problem and included in eight alternate candidates that require performance analysis in various environments. We analyze FrodoKEM algorithms and select six main steps that

◆° First and Corresponding Author : Kookmin University Department of Financial Information Security, fdt150@kookmin.ac.kr, 학생회원
* Kookmin University Department of Information Security, Cryptology and Mathematics, salt@kookmin.ac.kr, 종신회원; jskang@kookmin.ac.kr, 정회원
논문번호 : 202008-180-D-RE, Received July 31, 2020; Revised September 10, 2020; Accepted September 11, 2020

compose FrodoKEM algorithms. And we measure the execution time required for each major step in PC as well as Cortex-A53 board, which has better performance than the Cortex-M4 used in NIST evaluations. In addition, we investigate the performance of parallelizing essential two stages, which require the longest execution time, using GPU and the performance of FrodoKEM with the parallelized stages applied.

Ⅰ. 서 론

구글, IBM 등에서 양자 컴퓨팅 기술에 관한 연구를 진행하고 있고[1,2], 향후 양자 컴퓨터가 상용화 가능한 기술임을 예상할 수 있다. 양자 컴퓨터 상에서 공개키 암호(RSA, ECC, DSA 등)의 기반인 소인수 분해와 이산로그 문제는 1994년에 제안된 양자 알고리즘인 Shor의 알고리즘에 의해 다항 시간 내에 해결될 수 있다[3]. 따라서 양자 컴퓨터 상에서도 안전성을 보장받을 수 있는 암호 기술인 양자 내성 암호(Post-Quantum Cryptography, PQC)에 대한 연구가 필요하고, 현재 활발하게 진행되고 있다.

미국 국립표준기술연구소(NIST)는 양자 컴퓨팅 시대에 대응하기 위해 2016년부터 양자 내성 암호 공모전을 진행하고 있으며, 2022년에 표준 초안을 완성하는 것을 목표로 하고 있다. 대상 알고리즘은 서명(signature), 공개키 암호화(Public-Key Encryption, PKE)와 키 설정 메커니즘(Key Establishment Mechanism, KEM)이다. 2017년 12월에 공개된 64 개의 1 라운드 후보 알고리즘은 안전성에 대한 평가를 받았다. 2019년 1월, 1 라운드의 평가 결과가 반영되어 선정된 26 개의 2 라운드 후보 알고리즘이 공개되었다. 표 1.에 작성된 바와 같이, 12 개의 격자 기반 양자 내성 암호, 7 개의 코드 기반 양자 내성 암호, 2 개의 해시 기반 양자 내성 암호, 4 개의 다변수 기반 양자 내성 암호 그리고 1 개의 아이소제니 기반 양자 내성 암호가 2 라운드 후보 알고리즘으로 선정되었다. 2 라운드 후보 알고리즘은 2019년 8월, NIST 2nd PQC Standardization Conference에서 발표된 바와 같이 성능(하드웨어 및 소프트웨어)이 중점적으로 평가되었다[4]. 또한, 성능을 측정할 하드웨어로 Cortex-M4와 Artix-7이 권고되었다[4]. 2020년 7월, NIST는 안전성, 데이터 전송 비용과 계산 효율성을 모두 고려하여 15 개의 알고리즘을 3 라운드의 후보 알고리즘으로 선정하였다[5]. 표 2.에 작성된 바와 같이, 15 개의 알고리즘 중 7 개의 알고리즘이 최종 후보이고 8 개의 알고리즘이 대체 후보이다. 최종 후보 알고리즘은 12 ~ 18 개월 동안 진행될 예정인 3 라운드가 종료되면 표준화될 대상으로 검토되는 알고리즘이다[5]. 대체 후보 알고리즘은 또 다른 평가 라운드 후에 표준화될 가능성이 큰 알고리즘이다[5].

표 1. 2 라운드 후보 알고리즘
Table 1. Round 2 candidate algorithms

Type	2 round PQC algorithm	
	Signature	PKE/KEM
Lattice	CRYSTALS -DILIGHIUM, FALCON, qTESLA	CRYSTALS-KYBER, FrodoKEM, LAC, NewHope, NTRU, NTRU Prime, Round5, SABER, Three Bears
Code	-	BIKE, HQC, Classic McEliece, LEDAcrypt, ROLLO, NTS-KEM, RQC
Symmetric	SPHINCS+, Picnic	-
Multivariate	GeMSS, LUOV, MQDSS, Rainbow	-
Isogeny	-	SIKE

표 2. 3 라운드 최종 및 대체 후보 알고리즘
Table 2. Round 3 finalists / alternate candidates

Type	3 round finalists	
	Signature	PKE/KEM
Lattice	CRYSTALS -DILIGHIUM, FALCON	CRYSTALS-KYBER, NTRU, SABER
Code	-	Classic McEliece
Multivariate	Rainbow	-

Type	3 round alternate candidates	
	Signature	PKE/KEM
Lattice	-	FrodoKEM, NTRU Prime
Code	-	BIKE, HQC
Symmetric	SPHINCS+, Picnic	-
Multivariate	GeMSS	-
Isogeny	-	SIKE

2차 표준화 대상인 대체 후보 알고리즘은 다양한 환경에서의 성능에 대한 분석이 필요하다[5]. 따라서 본 논문은 PC와 NIST가 제시한 Cortex-M4보다 성능이 좋은 Cortex-A53 코어 보드 상에서 대체 후보 알고리즘인 FrodoKEM에 대한 성능을 분석하고자 한다. 또한, GPU를 이용한 고속 병렬화에 관한 연구가 NTRU 등의 양자 내성 암호에 대해서는 진행되고 있으나[6-8], FrodoKEM에 대해서는 연구가 미비하다. 따라서 본 논문에서는 FrodoKEM에 대해 GPU를 이용한 고속 병렬화의 가능성을 분석하고자 한다. 이때, 가장 대표적인 GPGPU(General Purpose computing on GPU) 프로그래밍 라이브러리인 CUDA(Compute Unified Device Architecture)[9]를 이용한다.

2장에서는 FrodoKEM의 기반 문제인 LWE(Learning With Errors) 문제와 FrodoKEM의 구조를 소개한다. 3장에서는 2 종의 플랫폼에서 FrodoKEM의 주요 단계별 성능을 측정한 결과를 제시한다. 4장에서는 소요 시간이 가장 긴 2 가지의 각 주요 단계를 GPU를 이용하여 병렬화한 성능을 제시하고 분석한다. 또한, FrodoKEM에 제시된 행렬 파라미터값과 함께 더 큰 행렬을 사용하는 파라미터값의 병렬 구현결과를 비교하여 병렬화 성능 향상의 한계를 분석한다. 병렬화된 단계가 적용된 FrodoKEM의 성능을 제시하여 5장에서는 결론을 맺는다.

Ⅱ. FrodoKEM

FrodoKEM[10]은 LWE 문제를 기반으로 하는 키 설정 메커니즘으로, IND-CCA(Indistinguishability under Chosen Ciphertext Attack) 안전성을 세 가지의 레벨(level)로 제공하도록 설계되었다. 각 레벨 1/3/5는 NIST가 정의한 보안 요구사항으로, 각 128/192/256 비트 크기의 키를 가진 블록 암호(AES 등)에 대한 키 전수조사 공격에 필요한 계산량 이상의 보안 강도를 가짐을 의미한다. FrodoKEM-640은 레벨 1, FrodoKEM-976은 레벨 3, 그리고 FrodoKEM-1344는 레벨 5를 제공하는 것을 목표로 한다. 각각의 알고리즘은 두 가지의 변형을 제공한다. 각 변형은 최대 (1344 × 1344) 크기의 공개 행렬 A를 의사난수(pseudo-random number)로 생성하기 위해 AES-128 또는 shake-128을 사용하는 것으로 구분된다. Intel 플랫폼의 AES-NI 등과 같은 AES 하드웨어 가속기가 있는 경우 AES-128을 사용하는 FrodoKEM-640/976/1344-AES이 적합하다. 반면, 하드웨어 가속기가 없는 경우, FrodoKEM-640/976/1344-SHAKE가 AES-128을 사용하는 것과 유사하거나 더 나은 성능을 제공한다.

2.1 LWE 문제

본 절에서는 LWE 확률 분포를 정의하고, FrodoKEM의 기반이 되는 LWE 문제를 소개한다.

정의> LWE 확률 분포[10]

n, q는 양의 정수이고, χ는 Z 위에서의 분포라 하자. 벡터 $\vec{s} \in Z_q^n$에 대해, LWE 분포 $L_{s,\chi}$는 랜덤한 벡터 $\vec{a} \in Z_q^n$와 분포 χ로부터의 에러 $e \in Z_p$를 선택하여 얻은 $Z_q^n \times Z_q$ 위에서의 분포로, $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e \bmod q) \in Z_q^n \times Z_q$를 출력한다.

LWE 문제는 2005년에 Regev가 처음 제안한 문제이고, LWE 문제를 푸는 것은 최악인 경우의 격자 문제들(worst-case lattice problems)을 푸는 것을 의미한다[11]. LWE 문제에는 탐색 문제(search problem)와 결정 문제(decision problem)가 있다. 탐색 문제는 랜덤한 벡터 $\vec{s} \in Z_q^n$에 대한 LWE 분포 $L_{s,\chi}$로부터 m 개의 샘플 $(\vec{a_i}, \langle \vec{a_i}, \vec{s} \rangle + e_i \bmod q)$, $i = 1, ..., m$이 주어졌을 때, 벡터 $\vec{s}$를 찾는 것이다[10]. 결정 문제는 랜덤한 벡터 $\vec{s} \in Z_q^n$에 대한 LWE 분포 $L_{s,\chi}$로부터 추출된 m 개의 샘플 $(\vec{a_i}, \langle \vec{a_i}, \vec{s} \rangle + e_i \bmod q)$, $i = 1, ..., m$과 균등분포 $U(Z_q^n \times Z_q)$로부터 추출된 m 개 $(\vec{a_i}, b_i \in Z_q)$, $i = 1, ..., m$을 구별하는 것이다[10].

2.2 FrodoKEM 구조

FrodoKEM의 키 생성(key generation) 과정에서는 개인키 sk와 공개키 pk를 생성한다. 캡슐화(encapsulation) 과정에서는 공개키 pk를 입력받아 임의의 메시지에 대한 암호문 ct와 비밀키 ss를 생성한다. 그리고 역캡슐화(decapsulation) 과정에서는 개인키 sk와 암호문 ct를 입력받아 비밀키 ss를 생성한다. 이처럼 비밀키 ss가 공유되는 메커니즘인 FrodoKEM의 전체적인 동작 과정을 그림으로 도식화하면 그림 1.과 같다. 또한, FrodoKEM이 사용하는 주요 파라미터의 값은 표 3.에 작성되어 있다.

그림 2.에서 보이는 바와 같이, FrodoKEM의 각 과정은 운영체제 난수발생기, shake-128 또는 shake-256, AES-128, 행렬 곱 연산, 에러 샘플링

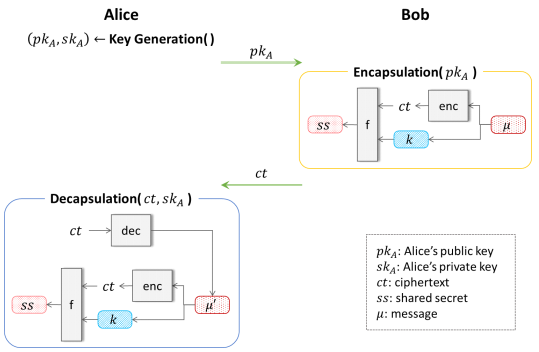

그림 1. FrodoKEM의 전체적인 동작 과정
Fig. 1. Overall process of FrodoKEM

표 3. FrodoKEM-640/976/1344의 주요 파라미터[10]
Table 3. Parameters for FrodoKEM-640/976/1344[10]

Parameter	FrodoKEM -640	FrodoKEM -976	FrodoKEM -1344
n	640	976	1344
$\bar{n}$	8	8	8

(error sampling), 인코딩(encoding), 패킹(pack) 등과 같은 함수로 구성되어 있다. 이를 통해 FrodoKEM의 각 과정을 6 가지의 주요 단계로 나누었다[12]. 6 가지의 단계와 그 사용처는 다음과 같다.

① 운영체제 난수발생기(Linux OS의 /dev/urandom 또는 Windows OS의 BcryptGenRandom)
: 난수(random number)인 시드(seed), 메시지 등 생성

② shake-128 또는 shake-256
: 의사난수인 시드 등 생성.
FrodoKEM-640/976/1344-SHAKE일 경우, 공개 행렬 A 생성

③ AES-128
: FrodoKEM-640/976/1344-AES일 경우, 공개 행렬 A 생성

④ 행렬 곱 연산
: 암호문 생성. 평문 복호화

⑤ 에러 샘플링
: 이산 가우시안 분포로부터 샘플링한 에러로 구성된 행렬 생성

⑥ 기타(인코딩, 패킹 등)

Ⅲ. 2 종의 플랫폼에서의 FrodoKEM 성능

본 장은 2 종의 플랫폼에서 FrodoKEM의 성능을 분석하고자 한다. 또한, 2.2절에서 살펴본 6 가지의 주요 단계별로 성능을 측정한 결과를 통해, 소요 시간이 긴 주요 단계를 GPU를 이용하여 병렬화할 대상으로 선정하고자 한다. 이때, 공개 행렬 A를 생성하기 위해 AES-128을 사용하는 FrodoKEM-640/976/1344-AES을 성능 측정 대상으로 선정하였다.

3.1 실험환경

실험 플랫폼은 PC와 Khadas VIM2 Pro 보드[13]로, 이에 대한 설명은 표 4.에 작성되어 있다. 대체 후보 알고리즘인 FrodoKEM은 다양한 환경에서의 성능에 대한 추가 분석이 필요하므로[5], NIST가 성능 측정을

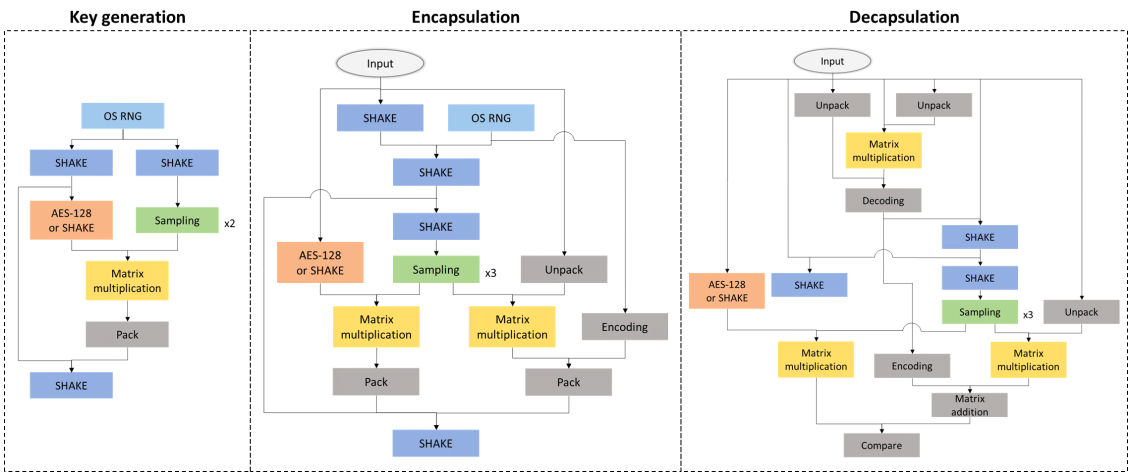

그림 2. 함수 단위로 도식화한 키 생성 과정, 캡슐화 과정과 역캡슐화 과정
Fig. 2. Key generation, encapsulation and decapsulation displayed by functions

표 4. 실험환경
Table 4. Experiment environments

	PC	Khadas VIM2 Pro board
CPU model	Intel Core i7-8086K	64-bit Octa Core ARM Cortex-A53
OS	Windows 10	Ubuntu 16.04.4
GPU	NVIDIA GeForce GTX TITAN Xp	ARM Mali-T82.MP3

위한 하드웨어로 제시하였던 Cortex-M4보다 성능이 좋은 Cortex-A53 코어를 내장하고 있는 Khadas VIM2 보드를 실험 플랫폼으로 선정하였다.

FrodoKEM의 저자가 제출한 최적화 코드(optimized code)[14]를 활용하였고, 32 비트 단위로 최적화한 AES-128을 사용하여 주요 단계별 성능을 측정하는 실험을 진행하였다. 사용한 컴파일 옵션은 다음과 같다.

- ○ CC: gcc
- ○ ARCH: x64
- ○ GENERATIN_A: AES128
- ○ USE_OPENSSL: TRUE

3.2 주요 단계별 소요 시간

PC와 Khadas VIM2 보드에서 최대 파라미터값(표 3.)을 가지는 FrodoKEM-1344-AES의 성능을 마이크로초(microseconds, μs) 단위로 측정한 결과는 표 5.에 작성되어 있다. 표 5에서 보이는 바와 같이, PC와 Khadas VIM2 보드에서 키 생성 과정, 캡슐화 과정과 역캡슐화 과정의 각 소요 시간이 유사함을 확인할 수 있다. 또한, 2 종의 플랫폼에서 모두 AES-128의 소요 시간이 각 과정의 전체 소요 시간에서 가장 큰 비중을 차지하고, 행렬 곱 연산의 소요 시간이 그다음 비중을 차지함을 확인할 수 있다.

2 종의 플랫폼에서 성능을 측정한 각각의 FrodoKEM-640/976/1344-AES의 각 과정의 전체 소요 시간 대비 주요 단계별 소요 시간의 비율을 그래프로 나타내면 그림 3.과 같다. 그림 3.의 그래프에서 x 축은 6 가지의 주요 단계를 나타내고, y 축은 전체 소요 시간 대비 각 단계의 소요 시간의 비율을 나타낸다. 또한, 범례에 작성된 PC-640, PC-976, PC-1344는 PC에서 각 FrodoKEM-640/976/1344-AES의 성능을 측정한 결과를 나타내고, VIM2-640, VIM2-976, VIM2-1344는 Khadas VIM2 보드에서 성능을 측정한 결과를 나타낸다. 그림 3.의 그래프에서 보이는 바와 같이, PC에서 AES-128은 각

표 5. 2 종의 플랫폼에서의 FrodoKEM-1344의 주요 단계별 소요 시간 측정 결과 (단위: μs)
Table 5. Execution time required for each step of FrodoKEM-1344 on two platforms (μs)

FrodoKEM-1344		PC	Khadas VIM2 board
Key generation	①	3	10
	②	245	739
	③	10,947	8,768
	④	3,584	5,349
	⑤	45	305
	⑥	194	908
	Total	15,018	16,079
Encapsu-lation	①	2	10
	②	337	1,002
	③	10,951	8,832
	④	3,870	5,880
	⑤	46	309
	⑥	729	2,693
	Total	15,935	18,726
Decapsu-lation	①	-	-
	②	245	749
	③	10,832	8,822
	④	3,870	5,903
	⑤	45	308
	⑥	725	2,711
	Total	15,717	18,493

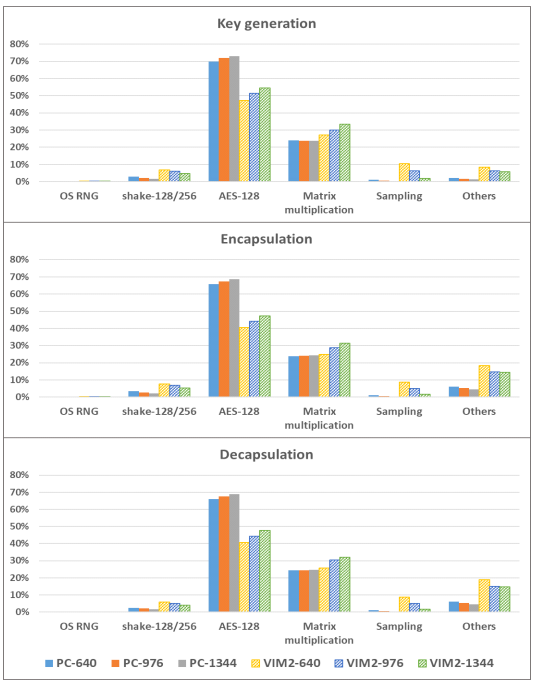

그림 3. 각 과정의 전체 소요 시간 대비 주요 단계별 소요 시간의 비율
Fig. 3. Ratio of the execution time of each step to the total time executed in each process

과정의 전체 소요 시간의 66 ~ 73 %를 차지하였고, 행렬 곱 연산은 약 24 %를 차지하였다. 이는 각 과정의 90 % 이상의 소요 시간이 AES-128과 행렬 곱 연산을 수행하는 데 사용됨을 의미한다. 또한, Khadas VIM2 보드에서 AES-128은 전체 소요 시간의 40 ~ 55 %를 차지하였고, 행렬 곱 연산은 25 ~ 33 %를 차지하였다. 이는 각 과정의 65 ~ 88 % 이상의 소요 시간이 AES-128과 행렬 곱 연산을 수행하는 데 사용됨을 의미한다.

Ⅳ. GPU를 이용하여 최적화된 FrodoKEM 성능 분석

본 장에서는 3.2절에서 살펴본 FrodoKEM의 소요 시간의 대부분을 차지하는 AES-128과 행렬 곱 연산을 GPU를 이용하여 병렬화하였을 때의 성능을 제시한다. 표 3.에 작성된 FrodoKEM이 제시한 파라미터값뿐만 아니라 다른 파라미터값도 사용하여 성능을 측정한 결과를 제시하고, FrodoKEM이 제시한 파라미터값에 대한 성능 향상의 한계치를 제시한다. 또한, GPU를 이용하여 최적화된 AES-128과 행렬 곱 연산이 모두 적용된 FrodoKEM의 성능을 측정한 결과를 제시한다. 성능은 10^3 또는 10^6 사이클 카운트(cycle count) 단위로 측정한다.

4.1 GPU를 이용하여 병렬화한 AES-128의 성능 및 분석

4.1.1 FrodoKEM의 AES-128이 사용되는 과정

AES-128은 FrodoKEM의 키 생성 과정, 캡슐화 과정과 역캡슐화 과정에서 128 비트 크기의 시드 $seed_A$로부터 공개 행렬 $A \in Z^{n \times n_q}$를 생성할 때 사용된다. 각 과정에서 AES-128은 $n^2/8$ 번 사용된다. 최대 파라미터값을 갖는 FrodoKEM-1344의 경우, 각 과정에서 AES-128은 225,792 번 동작한다.

FrodoKEM의 문서[10]와 소스 코드[14]에 대한 분석을 통해 확인한 AES-128을 이용하여 $n \times n$ 크기의 행렬 A가 생성되는 과정은 다음과 같다. 128 비트 크기의 데이터 C를 $(<i> \| <8j> \| 0 \| ... \| 0)$로 구성한다. 이때, i는 행렬 A의 열 인덱스(index)로, $0 \le i < n$이다. j는 행렬 A의 일부 행 인덱스로, $0 \le i < n/8$이다. 또한, $<\bullet>$는 크기가 16 비트인 데이터를 의미한다. 데이터 C를 시드 $seed_A$가 암호화 키인 AES-128의 입력 데이터로 사용한다.

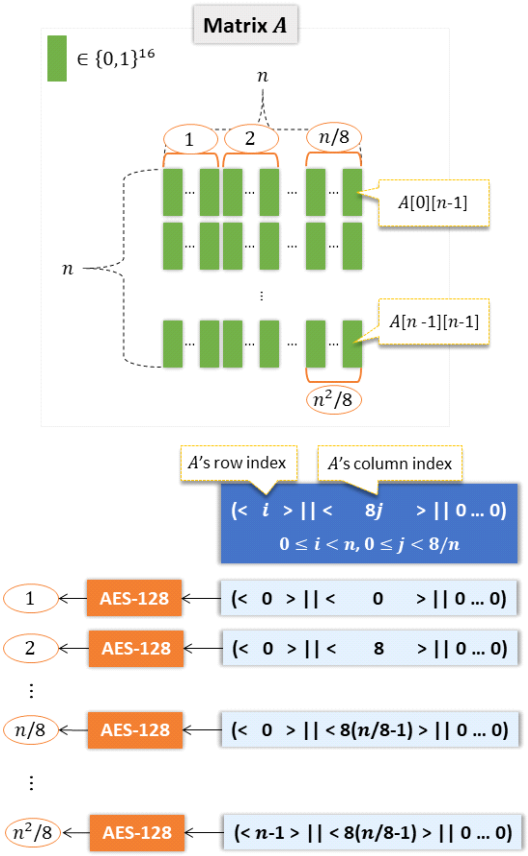

그림 4. AES-128이 사용되는 행렬 A 생성 과정
Fig. 4. The process of generating a matrix A using AES-128

AES-128의 출력 데이터 $(o_0, ..., o_7)$은 $A[i][8j] = o_0$, $A[i][8j+1] = o_1$, ..., $A[i][8j+7] = o_7$이 된다. 이때, o_i와 $A[i][j]$의 크기는 16 비트이다. 이 과정을 그림으로 도식화하면 그림 4.와 같다.

4.1.2 병렬화한 AES-128의 성능 및 분석

$n^2/8$ 번의 AES-128을 GPU를 이용하여 병렬화하기 위해, 열 인덱스 i를 GPU 블록(block)의 인덱스(blockIdx)로 대체하고 일부 행 인덱스 j를 스레드(thread)의 인덱스(threadIdx)로 대체하였다. 따라서 $n/8$개의 스레드로 구성된 n 개의 GPU 블록을 사용하였다. 또한, 블록당 5,296 바이트 크기의 공유 메모리(shared memory)가 사용되었고, 점유(occupancy)는 100 %로 계산되었다.

행렬 A의 행/열 크기인 n에 대해, GPU를 이용하여 병렬 구현한 행렬 A의 생성 과정의 성능은 CPU에서의 성능과 함께 표 6.에 작성되어 있다. 그림 5.는

파라미터 n에 따른 행렬 A의 생성 과정의 CPU 대비 GPU 성능을 보여준다. 표 6.과 그림 5.에서 보이는 바와 같이, n의 값이 커질수록 병렬 구현한 행렬 A의 생성 과정의 성능이 향상됨을 확인할 수 있다. 또한, 표 3.에 작성된 FrodoKEM이 제시하는 파라미터값으로 병렬 구현하면 CPU에서의 성능보다 최대 4.4 배의 성능 향상을 얻을 수 있음을 확인할 수 있다.

표 6. 파라미터 n의 값에 따른 행렬 A 생성 과정에 대한 CPU 상에서의 성능 및 GPU 상에서의 성능 (단위: 10^6 사이클 카운트)
Table 6. Performances on CPU and on GPU for the process of generating matrix A according to the value of parameter n (10^6 cycle counts)

n		CPU	GPU
6,144		887	116
4,096		339	57
2,048		99	17
FrodoKEM	1,344	44	10
	976	23	6
	640	9	4

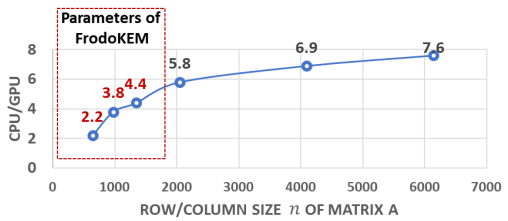

그림 5. 파라미터 n의 값에 따른 행렬 A 생성 과정에 대한 CPU에서의 성능 대비 GPU에서의 성능증가율
Fig. 5. Ratio of the performance improvement on GPU to the performance on CPU for the process of generating matrix A according to the value of parameter n

4.2 GPU를 이용하여 병렬화한 행렬 곱 연산의 성능 및 분석

4.2.1 FrodoKEM의 행렬 곱 연산이 사용되는 과정

FrodoKEM의 키 생성 과정, 캡슐화 과정과 역캡슐화 과정에서 $n\times n$ 크기의 행렬 A와 $n\times\bar{n}$ 크기의 행렬 S를 곱하여 $n\times\bar{n}$ 크기의 행렬 B를 계산한다. 이때, FrodoKEM 저자의 코드[14]에 따르면 $O(n^2\bar{n}/4)$의 계산량이 필요함을 확인하였다. 최대 파라미터값을 갖는 FrodoKEM-1344의 경우, 각 과정에서 행렬 곱 연산에 필요한 계산량은 3,612,672이다.

4.2.2 병렬화한 행렬 곱 연산의 성능 및 분석

$n\times n$ 크기의 행렬 A와 $n\times\bar{n}$ 크기의 행렬 S에 대한 행렬 곱 연산을 GPU를 이용하여 병렬화하기 위해, 행렬 B에 해당하는 $n\times\bar{n}$ 크기의 데이터 중에서 $\bar{n}\times\bar{n}$크기의 데이터를 하나의 GPU 블록이 처리하도록 구현하였다[9]. 하나의 블록이 $\bar{n}\times\bar{n}$ 크기의 데이터를 계산하는 데 필요한 행렬 A에서의 $n\times\bar{n}$ 크기의 데이터와 행렬 S는 GPU의 전역 메모리(global memory)에 저장된다. 하지만, GPU의 메모리 중 전역 메모리를 읽고 쓰는데 걸리는 시간이 가장 오래 걸린다[9]. 따라서, 전역 메모리에 대한 접근 지연 시간을 줄이기 위해 행렬 A에서의 $n\times\bar{n}$ 크기의 데이터와 행렬 S 각각에서 계산에 필요한 $\bar{n}\times\bar{n}$ 크기의 데이터를 공유 메모리에 저장하여 사용하였다. 표 3.에 작성된 FrodoKEM이 제시하는 파라미터 $\bar{n}$의 값은 8이므로, 256 개의 스레드로 구성된 $n/8$ 개의 GPU 블록을 사용하였다. 또한, 블록당 512 바이트 크기의 공유 메모리가 사용되었고, 점유는 100 %로 계산되었다. 이 과정을 그림으로 도식화하면 그림 6.과 같다.

행렬 B의 열/행 크기인 n과 $\bar{n}$에 따라, GPU를 이용하여 병렬 구현한 행렬 곱 연산의 성능은 CPU에서의 성능과 함께 표 7.에 작성되어 있다. 그림 7.은 파라미터 n과 $\bar{n}$에 따라 행렬 곱 연산의 CPU 대비 GPU 성능을 보여준다. 표 7.과 그림 7.에서 보이는 바와 같이, n과 $\bar{n}$의 값이 커질수록 병렬 구현한 행렬 곱 연산의 성능이 향상됨을 확인할 수 있다. 또한, 표 3.에 작성된 FrodoKEM이 제시하는 파라미터값으로 병렬 구현하면 CPU에서의 성능보다 최대 2.3 배의 성

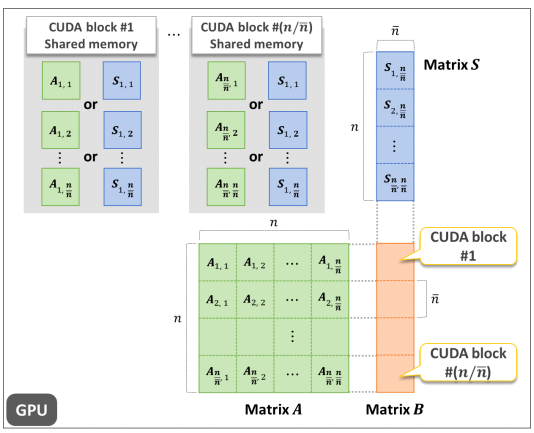

그림 6. 행렬 곱 연산에 대한 병렬화 방법
Fig. 6. Parallel method for matrix multiplication

표 7. 파라미터 n과 $\overline{n}$의 값에 따른 행렬 곱 연산에 대한 CPU 상에서의 성능 및 GPU 상에서의 성능 (단위: 10^6 사이클 카운트)
Table 7. Performances on CPU and on GPU for the process of matrix multiplication according to the value of parameters n and $\overline{n}$ (10^6 cycle counts)

n		$\overline{n}$	CPU	GPU
3,200		3,200	36,925	496
1,344		1,344	2,659	48
		320	624	17
		64	125	9
		16	32	7
FrodoKEM	1,344	8	16	7
	976	8	9	4
	640	8	4	3

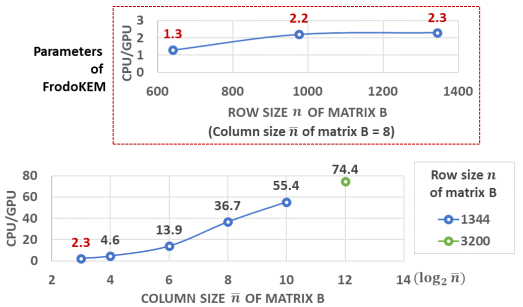

그림 7. 파라미터 n과 $\overline{n}$의 값에 따른 행렬 곱 연산에 대한 CPU에서의 성능 대비 GPU에서의 성능증가율
Fig. 7. Ratio of the performance improvement on GPU to the performance on CPU for matrix multiplication according to the value of parameters n and $\overline{n}$

능 향상을 얻을 수 있음을 확인할 수 있다.

4.3 GPU를 이용하여 최적화된 FrodoKEM의 성능 및 분석

본 절에서는 4.1절과 4.2절에서 GPU를 이용하여 병렬화한 AES-128과 행렬 곱 연산이 적용된 FrodoKEM의 성능을 측정한 결과를 제시한다. 성능은 10^3 사이클 카운트 단위로 측정한다.

표 8.은 CPU 상에서의 FrodoKEM-640-AES의 주요 단계별 성능과 GPU 상에서의 최적화 기법이 적용된 주요 단계별 성능을 보여준다. 표 9.와 표 10.은 각각 FrodoKEM-976/1344-AES에 대한 CPU와 GPU 상에서의 성능을 보여준다. 표 8.에서 보이는 바와 같이, FrodoKEM-640-AES의 경우, 각 과정에 대해 약 1.8 배의 성능 향상이 이루어졌다. 이는 FrodoKEM이 제시하는 파라미터에 대한 성능 향상의 한계치인 그림 5.와 그림 7.에서 보이는 바와 유사하게, AES-128을 이용한 행렬 생성 과정의 성능이 약 2.5 배 향상되

표 8. CPU와 GPU 상에서의 FrodoKEM-640의 주요 단계별 성능 (단위: 10^3 cycle counts)
Table 8. Execution time of each step of FrodoKEM-640 on CPU and on GPU (10^3 cycle counts)

FrodoKEM-640		CPU	GPU
Key generation	①	28	4
	②	366	368
	③	9,621	3,851
	④	3,170	3,289
	⑤	146	145
	⑥	248	262
	Total	13,579	7,919
Encapsu-lation	①	7	3
	②	512	486
	③	10,076	3,677
	④	3,679	3,225
	⑤	159	145
	⑥	900	429
	Total	15,333	7,965
Decapsu-lation	①	-	-
	②	378	371
	③	9,990	3,707
	④	3,634	3,333
	⑤	152	149
	⑥	898	465
	Total	15,052	8,025

표 9. CPU와 GPU 상에서의 FrodoKEM-976의 주요 단계별 성능 (단위: 10^3 cycle counts)
Table 9. Execution time of each step of FrodoKEM-976 on CPU and on GPU (10^3 cycle counts)

FrodoKEM-976		CPU	GPU
Key generation	①	26	5
	②	692	684
	③	22,345	6,139
	④	7,307	4,162
	⑤	189	186
	⑥	439	419
	Total	30,998	11,595
Encapsu-lation	①	3	6
	②	907	924
	③	22,170	6,424
	④	7,893	4,382
	⑤	190	189
	⑥	1,730	606
	Total	32,893	12,531
Decapsu-lation	①	-	-
	②	696	695
	③	22,183	6,312
	④	7,986	4,480
	⑤	191	190
	⑥	1,760	633
	Total	32,816	12,310

표 10. CPU와 GPU 상에서의 FrodoKEM-1344의 주요 단계별 성능 (단위: 10^3 cycle counts)
Table 10. Execution time of each step of FrodoKEM-1344 on CPU and on GPU (10^3 cycle counts)

FrodoKEM-1344		CPU	GPU
Key generation	①	27	8
	②	960	944
	③	42,828	10,254
	④	14,042	7,619
	⑤	177	171
	⑥	690	683
	Total	58,724	19,679
Encapsu-lation	①	5	4
	②	1,256	1,251
	③	42,341	9,602
	④	14,923	7,402
	⑤	174	172
	⑥	2,774	906
	Total	61,473	19,338
Decapsu-lation	①	-	-
	②	943	942
	③	42,192	9,752
	④	15,046	7,591
	⑤	173	173
	⑥	2,781	921
	Total	61,136	19,379

었고 행렬 곱 연산의 성능이 약 1.3 배 향상되었기 때문으로 분석된다. 표 9.에서 보이는 바와 같이, FrodoKEM-976-AES의 각 과정은 성능이 약 2.3 배 향상되었다. 또한, 표 10.에서 보이는 바와 같이, FrodoKEM-1344-AES의 각 과정은 성능이 약 3 배 향상되었다.

표 8. ~ 표 10.의 결과를 통해 FrodoKEM의 각 과정이 가질 수 있는 최대 성능 향상치를 다음과 같이 계산할 수 있다. 만약 AES-128을 이용한 행렬 생성 과정에 대해 완전한 최적화가 이루어진다면, 이에 필요한 사이클 카운트 값이 0에 가까워질 것이다. 이로 인해, 표 10.에 작성된 GPU를 이용하여 최적화된 FrodoKEM-1344-AES의 각 과정에 필요한 사이클 카운트 값이 약 10,000이 된다. 따라서 6 배의 성능 향상이 이루어진다. 이는 AES-128을 이용한 행렬 생성 과정의 최적화로 인해 FrodoKEM의 각 과정이 가질 수 있는 최대 성능 향상치다.

V. 결 론

본 논문은 NIST의 양자 내성 암호 공모사업의 3 라운드 대체 후보 알고리즘인 FrodoKEM의 성능을 주요 단계별로 나누어 PC와 Khadas VIM2 보드에서 측정하였다. 2 종의 각 플랫폼에서 AES-128을 이용한 행렬 생성 과정과 행렬 곱 연산이 키 생성 과정, 캡슐화 과정과 역 캡슐화 과정의 각 소요 시간의 90 %, 65 % 이상을 차지함을 확인하였다. 본 논문은 각각의 GPU를 이용하여 병렬화한 AES-128과 행렬 곱 연산의 성능이 파라미터값이 커질수록 향상됨을 확인하였고, FrodoKEM이 제시한 파라미터값에 대해서는 각각 최대 4배, 3배의 성능이 향상됨을 확인하였다. 또한, GPU를 이용하여 병렬화한 AES-128과 행렬 곱 연산이 적용된 FrodoKEM의 성능이 최대 3 배 향상됨을 확인하였다.

본 논문의 결과는 FrodoKEM과 유사한 구조를 가지는 암호 알고리즘의 성능 분석에 활용되며 본 논문의 연구결과와 유사한 성능 향상이 이루어질 것으로 기대한다.

References

[1] F. Arute, et al., “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, no. 7779, pp. 505-510, Oct. 2019.

[2] P. Jurcevic, et al., “Demonstration of quantum volume 64 on a superconducting quantum computing system,” *arXiv preprint* arXiv:2008.08571, 2020.

[3] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Proc. IEEE 35th Annu. Symp. Foundations of Computer Sci.*, pp. 124-134, Santa Fe, USA, Nov. 1994.

[4] M. Dustin, “The 2nd round of the NIST PQC standardization process,” in *NIST Second PQC Standardization Conf.*, Santa Barbara, USA, Aug. 2019.

[5] G. Alagic, et al., “*Status report on the second round of the NIST Post-Quantum cryptography standardization process*,” NIST, Tech. Rep., Jul. 2020.

[6] T. Bai, et al., “Accelerating NTRU encryption

with graphics processing units," *Int. J. Netw. and Distrib. Comput.*, vol. 2, no. 4, pp. 250-258, Oct. 2014.

[7] W. Dai, et al., "NTRU modular lattice signature scheme on CUDA GPUs," in *Proc. IEEE 2016 HPCS*, pp. 501-508, Innsbruck, Austria, Jul. 2016.

[8] S. Akleylek, D. Özgur, and Y. T. Zaliha, "On the efficiency of polynomial multiplication for lattice-based cryptography on GPUs using CUDA," in *Proc. Int. Conf. Cryptography and Info. Secur. in the Balkans*, Springer, Cham, pp. 155-168, Koper, Slovenia, Sep. 2015.

[9] NVIDIA, "*CUDA C++ programming guide*," PG-02829-001_v11.0, Jul. 2020.

[10] E. Alkim, et al., "*FrodoKEM: Learning With Errors Key Encapsulation*," NIST submissions, Updated Mar. 2020.

[11] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *JACM*, vol. 56, no. 6, pp. 1-40, Sep. 2009.

[12] Y. Kim and Y. Yeom, "A study on the GPU-based accelerated implementation of LWE-based post-quantum Cryptography in NIST submissions," in *Proc. KICS 2020*, pp. 628-629, Gangwon-do, Korea, Feb. 2020.

[13] *Khadas VIM2 Pro board*, Retrieved Jul. 27, 2020, from https://www.khadas.com/vim2.

[14] *PQCrypto-LWEKE*, Retrieved Jul. 27, 2020, from https://github.com/Microsoft/PQCrypto-LWEKE.

김 예 원 (Yewon Kim)

2015년 8월 : 숙명여자대학교 수학과 졸업

2017년 8월 : 국민대학교 금융정보보안학과 석사

2017년 9월~현재 : 국민대학교 금융정보보안학과 박사과정

<관심분야> 암호구현, 난수성 분석 및 평가, 병렬 프로그래밍

[ORCID:0000-0003-0334-6093]

염 용 진 (Yongjin Yeom)

1991년 2월 : 서울대학교 수학과 졸업

1994년 2월 : 서울대학교 수학과 석사

1999년 2월 : 서울대학교 수학과 박사

2000년 4월~2012년 2월 : ETRI 부설연구소 책임연구원/팀장

2012년 3월~현재 : 국민대학교 과학기술대학 정보보안암호수학과 정교수

2013년~현재 : 국민대학교 BK21+ 미래 금융정보보안 인력양성사업단 교수

<관심분야> 암호구현 및 분석, 보안시스템 평가

[ORCID:0000-0002-8240-8661]

강 주 성 (Ju-Sung Kang)

1989년 2월 : 고려대학교 수학과 졸업

1991년 2월 : 고려대학교 일반대학원 수학과 석사

1996년 2월 : 고려대학교 일반대학원 수학과 박사

1997년~2004년 : 한국전자통신연구원 선임연구원/팀장

2004년 3월~현재 : 국민대학교 과학기술대학 정보보안암호수학과 정교수

2013년~현재 : 국민대학교 BK21+ 미래 금융정보보안 인력양성사업단 교수

<관심분야> 암호이론, 정보보안 프로토콜, 안전성 분석 및 평가

[ORCID:0000-0002-0846-389X]