NTRU PQC 알고리즘 가속을 위한 성능 분석

김지환*, 조명현*, 이용석*, 백윤흥*
*서울대학교 전기정보공학부, 반도체공동연구소
{jhkim, mhcho, yslee}@sor.snu.ac.kr, ypaek@snu.ac.kr

Performance Analysis for Accelerating NTRU PQC Algorithm

Jeehwan Kim*, Myunghyun Cho*, Yongseok Lee*, and Yunheung Paek*
*Dept. of Electrical and Computer Engineering and Inter-University Semiconductor Research Center (ISRC), Seoul National University

요 약

양자 컴퓨터 기술의 발전에 따라 현재 사용되고 있는 암호 알고리즘과 시스템들이 위협받고 있다. 이러한 시대적 흐름에 따라 양자 컴퓨터로도 쉽게 해결할 수 없는 양자내성암호의 개발이 요구되고 있으며, 미국 NIST 에서는 양자내성암호의 표준화를 위한 공모전을 진행하고 있다. 본 논문에서는 공모전 최종 후보 중 하나인 NTRU 알고리즘을 가속화하기 위한 성능 분석을 진행하였다.

1. 서론

오늘날 IBM, Microsoft, Google 과 같은 기업들을 중심으로 양자 컴퓨터 기술은 빠르게 발전하고 있는데, 2015 년에는 일반적인 컴퓨터 대비 1 억 배 이상 빠른 성능의 양자 컴퓨터 D-Wave2 가 공개되었다. 양자 컴퓨터를 이용하면 현재 널리 사용되고 있는 암호 알고리즘인 RSA, ECC 와 같은 공개키 암호 방식을 쉽게 해결할 수 있어 새로운 암호 알고리즘이 필요하다.[1] 이러한 흐름에 따라 미국 국립표준기술연구소(NIST)에서는 양자내성암호(Post Quantum Cryptography, PQC) 알고리즘 표준화를 위한 공모전을 진행하고 있다.

현재 PQC 표준화 공모전은 세 번째 라운드까지 진행되면서 7 개의 최종 후보가 남아있으며, NTRU 알고리즘도 그 중 하나이다. NTRU 알고리즘은 1996 년에 처음 소개된 암호 알고리즘으로,[2] 발표된 이후 20 여 년 간 계속된 연구로 개선되면서 지금과 같은 형태의 격자 기반 암호 방식이 되었다.[3] NTRU 알고리즘은 속도, 암호키 크기, 암호문 크기 등에서 다른 격자 기반 암호 알고리즘보다 경쟁력을 갖고 있다.[4] 하지만 공모전 내 경쟁 알고리즘과 비교했을 때의 평범한 성능은 NTRU 알고리즘의 실용화를 위해 개선을 필요로 한다.

본 논문에서는 NIST PQC 표준화 공모전에 제출된 NTRU 알고리즘의 성능을 분석해보았다. 이를 통해서 NTRU 알고리즘에서 성능 개선이 필요한 부분을 찾을 수 있다.

2. 배경 지식

NTRU 알고리즘은 PQC 표준화 공모전의 첫 번째 라운드에서는 따로 제출된 NTRU-HRSS-KEM, NTRU-Encrypt 두 알고리즘이 통합된 공개키 암호(Public Key Encryption, PKE) 및 키 캡슐화 매커니즘(Key Encapsulation Mechanism, KEM)이다.[5]

NTRU 알고리즘은 환(Ring) 형태의 N-1 차 다항식을 두 서로소 p, q 로 모듈러하여 연산에 이용한다. 이러한 다항식 연산을 통해 PKE/KEM 암호 알고리즘이 수행된다. 아래 <표 1>은 통합된 NTRU 알고리즘의 네 가지 파라미터 세트를 나타낸 것이다.

<표 1> NTRU 알고리즘 Parameter Sets

	NTRUHPS 2048509	NTRUHPS 2048677	NTRUHPS 4096821	NTRU HRSS701
N	509	677	821	701
p	3	3	3	3
q	2048	2048	4096	8192

PKE/KEM 암호 방식은 키 생성(Key Generation), 암호화(Encapsulation), 복호화(Decapsulation)의 세 가지 단계로 이루어진다. 키 생성 단계에서는 암호화, 복호화에 필요한 공개키와 개인키를 생성한다. 암호화 단계에서는 공개키를 이용해 전달하고자 하는 메세지를 암호화한다. 복호화 단계에서는 개인키를 이용해 전달받은 메시지를 복호화한다.

<표 2> NTRU 알고리즘 주요함수 소요 시간 및 비중

Function	NTRUHPS2048509		NTRUHPS2048677		NTRUHPS4096821		NTRUHRSS701	
Key Generation	1744.66 (ms)	100 (%)	3047.28 (ms)	100 (%)	4582.23 (ms)	100 (%)	3351.03 (ms)	100 (%)
poly_Rq_inv	814.58	46.69	1442.87	47.35	2158.69	47.11	1589.06	47.42
poly_Rq_mul x8	306.01	17.54	482.08	15.82	707.04	15.43	493.61	14.73
poly_S3_inv	759.80	43.55	1314.29	43.13	1980.90	43.23	1445.97	43.15
poly_Rq_mul x5	173.59	9.59	289.49	9.50	426.14	9.30	313.32	9.35
Encapsulation	46.05 (ms)	100 (%)	78.96 (ms)	100 (%)	110.58 (ms)	100 (%)	80.31 (ms)	100 (%)
poly_Rq_mul	39.51	85.80	68.16	86.33	99.14	89.66	76.47	95.22
sample_rm	4.72	10.26	6.94	8.79	8.81	7.97	0.55	0.69
Decapsulation	116.60 (ms)	100 (%)	209.47 (ms)	100 (%)	301.95 (ms)	100 (%)	226.37 (ms)	100 (%)
poly_S3_mul	39.27	33.68	69.71	33.28	99.67	33.01	74.90	33.09
poly_Rq_mul	38.46	32.98	67.76	32.35	99.34	32.90	72.80	32.16
poly_Sq_mul	39.14	33.57	69.71	33.28	99.43	32.93	74.90	33.09
poly_Rq_mul	38.29	32.84	67.55	32.25	99.34	32.90	72.57	32.06
poly_Rq_mul	38.26	32.70	67.49	32.22	99.34	32.90	72.51	32.08

3. 성능 분석 실험 및 결과

NTRU 알고리즘을 각 파라미터 세트와 단계별로 나눠서 성능 분석을 진행했다. 성능 분석을 위해 336MHz ARM Cortex-M4 가 장착된 STM32F407VG 마이크로 컨트롤러를 이용했고, GCC 컴파일러의 최적화 정도는 -O0 으로 설정했다.

위 <표 2>는 NTRU 알고리즘의 각 파라미터 세트에 따른 성능 분석 결과를 단계별 주요 함수가 실행되는데 걸리는 시간과 비중으로 나타냈다. 표에서 들여쓰기된 함수는 상위 함수에 속한 하위 함수를 나타내고, 여러 번 호출된 함수는 x(호출된 횟수) 로 나타냈다.

실험 결과, 모든 알고리즘의 동일한 단계에서 특정 함수가 대부분의 시간을 소모하는 것을 확인할 수 있었다. 먼저 키 생성 단계에서는 다항식의 모듈러 역원을 연산하는 poly_Rq_inv, poly_S3_inv 두 함수가 수행되는 동안 90% 정도의 시간이 소모되었고, 두 다항식의 곱연산인 poly_Rq_mul 함수가 실행되는 시간을 더하면 99% 정도의 시간을 소요했다.

암호화 단계에서는 NTRUHPS 알고리즘과 NTRU-HRSS 알고리즘 간의 차이가 나타났는데, NTRUHPS 알고리즘에서는 poly_Rq_mul 함수가 85~89%의 비중을 차지하고 메시지의 다항식을 생성하는 sample_rm 함수가 7.9~10.2%의 비중을 차지했다. 하지만 NTRU-HRSS 알고리즘에서는 poly_Rq_mul 함수가 95.2%의 비중을 차지하고 sample_rm 함수의 비중은 0.69%에 그쳤다.

복호화 단계에서는 poly_S3_mul, poly_Sq_mul, poly_Rq_mul 함수가 각각 32, 33% 정도의 비중을 차지했다. 그런데 poly_S3_mul 함수와 poly_Sq_mul 함수가 실행되는 시간의 대부분이 하위 함수인 poly_Rq_mul 함수가 실행되는데 소요되어, 복호화 단계에서는 poly_Rq_mul 함수의 실행이 96~98%의 비중을 차지한다는 것을 알 수 있다.

4. 결론

NTRU 알고리즘을 파라미터 세트와 단계별로 성능을 분석한 결과, 어떤 함수가 실행되면서 시간을 많이 소요하는지 확인할 수 있었다. 키 생성 과정에서는 poly_Rq_inv 함수와 poly_S3_inv 함수가 수행 시간의 90% 이상을 차지했고, 암호화, 복호화 과정에서는 poly_Rq_mul 함수가 sample_rm 함수와 함께 95% 이상의 비중을 차지했다. 특히 poly_Rq_mul 함수는 키 생성 과정에서도 24~27% 정도의 시간을 소요했다. 따라서 poly_Rq_inv, poly_S3_inv, poly_Rq_mul, sample_rm 함수를 가속하면 NTRU 알고리즘의 성능을 향상시킬 수 있을 것으로 기대된다.

향후 이 네 함수의 알고리즘을 단순화하여 가속하거나, 하드웨어 가속기를 설계하여 가속화하는 연구 등 다양한 방법으로 NTRU 알고리즘을 가속하는 연구가 필요할 것이다.

ACKNOWLEDGEMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구이며 (No.2018-0-00230, (IoT 총괄/1 세부) IoT 디바이스 자율 신뢰 보장 기술 및 글로벌 표준기반 IoT 통합보안 오픈 플랫폼 기술개발 [TrusThingz 프로젝트]), (No.2020-0-00325, 클라우드 엣지 전주기 데이터 안정성을 위한 추적성 보장 기술 개발), 2021 년도 BK21 FOUR 정보기술 미래인재 교육연구단의 지원을 받았으며, 2021 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2020R1A2B5B03095204).

참고문헌

[1] Peter W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring", 35th Annual Symposium on Foundations of Computer Science, 1994

[2] J. Hoffstein, J. Pipher, J.H. Silverman, “NTRU: A new high speed public key cryptosystem”, Crypto 96, 1996

[3] J. Hoffstein, J. Pipher, J.H. Silverman, "NTRU: A ring-based public key cryptosystem", International Algorithmic Number Theory Symposium (ANTS 1998), 1998

[4] Andreas Hulsing, Joost Rijneveld, John Schanck, and Peter Schwabe, “High-Speed Key Encapsulation from NTRU”, Cryptographic Hardware and Embedded Systems (CHES 2017), 2017

[5] Cong Chen, et al, "NTRU-Algorithm Specifications And Supporting Documentation", NIST Post-Quantum Cryptography Standardization Project, 2020