KpqC 격자 기반 알고리즘 후보의 벤치마크

권혁동* 심민주* 송경주* 이민우** 서화정***

*한성대학교 정보컴퓨터공학과 (대학원생)
**한성대학교 IT융합공학부 (대학원생)
***한성대학교 융합보안학과 (부교수)

Benchmarks of KpqC lattice-based algorithm candidates

Hyeok-Dong Kwon* Min-Joo Sim* Gyeong-Ju Song* Min-Woo Lee** Hwa-Jeong Seo***

*Dept. of Information Computer Engineering, Hansung University(Graduate student)
**Dept. of IT Convergence Engineering, Hansung University(Graduate student)
***Dept. of Convergence Security, Hansung University(Associate Professor)

요 약

한국형 양자내성암호 표준 선정을 위해 양자내성암호연구단은 국가보안기술연구소와 협력하여 KpqC 공모전을 개최하였다. 본 공모전에는 공개키 암호화 알고리즘 7종, 전자서명 알고리즘 9종이 Round 1을 통과하였다. 각 알고리즘은 Round 2에 진출하기 위해 지속적인 연구 개발이 진행 중에 있다. 하지만 알고리즘마다 성능을 측정한 환경이 다르기 때문에 직접적인 비교가 다소 어려운 점이 존재한다. 본 논문에서는 KpqC Round 1 후보 알고리즘 중에서 격자 기반 알고리즘을 통일된 환경에서 벤치마크한 결과를 제시하며, 알고리즘 간의 원활한 비교를 하고자 한다.

I. 서론

양자컴퓨터의 등장으로 현대 암호 체계가 많은 위협을 받고 있다. 미국의 국립표준기술연구소(National Institute of Standards and Technology, NIST)는 양자내성암호의 선제 도입을 위해 표준화 선정이 필요함을 강조하였고, 이는 양자내성암호 표준화 공모전으로 이어졌다. 공모전 결과, 공개키 암호화 알고리즘 Kyber[1], 전자서명 알고리즘으로 Dilithium[2], Falcon[3], SPHINCS+[4]가 선정되었다. 추가적인 공개키 암호화 알고리즘 표준 선정을 위해 Round 4가 진행 중에 있다.

한국에서도 한국형 양자내성암호 표준화 선정을 위해 KpqC 공모전이 개최되었다. 주체는 양자내성암호연구단과 국가보안기술연구소이다. 본 논문에서는 KpqC 공모전과 Round 1 알고리즘에 대한 간략한 확인과 벤치마크 결과를 제시한다. 이후 구성은 다음과 같다. 2장에서 KpqC 공모전 Round 1 알고리즘 중 격자 기반 알고리즘들을 간단히 소개한다. 3장에서 벤치마크를 위한 환경 설정과 그 결과를 제시한다. 4장에서 본 논문의 결론을 맺는다.

II. KpqC 공모전

KpqC 공모전은 한국의 양자내성암호 표준을 선정하기 위해 개최된 공모전으로, 23년 5월 현

Table. 1 Round 1 candidates of KpqC.

Type	PKE/KEM	Digital Sign.
Code -based	IPCC Layered-ROLLO PALOMA REDOG	Enhanced pqsigRM
Lattice -based	NTRU+ SMAUG TiGER	GCKSign HAETAE NCC-Sign MQ-Sign Peregrine SOLMAE
Hash -based	x	FIBS
Zero knowledge -based	x	AIMer

재 Round 1 진행 중이다. 동년 12월에 Round 2 결과를 발표할 예정이며, 24년 9월에 최종 결과를 발표할 계획이다. 표 1에서는 Round 1 알고리즘을 각 기반 별로 표시하였다. 코드 기반 암호가 5종, 격자 기반 암호가 9종, 해시 기반 암호가 1종, 영지식 기반 암호가 1종이 Round 1에 진출하였다. NIST의 공모전과는 다르게 다변수 다항식 기반 알고리즘은 1건도 진출하지 못하였다. Round 1 진출 알고리즘 중 과반수가 격자 기반 알고리즘이다. NIST에서 선정한 표준 4종 중 3종이 격자 기반 알고리즘인 것을 감안하면, 양자내성암호 기반으로는 격자 기반이 강세임을 알 수 있다.

III. 격자 기반 알고리즘 벤치마크

본 장에서는 KpqC 후보 알고리즘 중, 격자 기반 알고리즘의 벤치마크 결과를 소개한다. 공개키 암호화 알고리즘, 전자서명 알고리즘 개별로 성능 측정 결과를 나열하였다.

3.1. 벤치마크 환경 설정

각 알고리즘을 벤치마크하기 위해 몇 가지 설정을 수행한다. 첫 번째로는 외부 라이브러리 의존성 제거 과정이다. 대부분의 알고리즘은 OpenSSL 의존성을 지니고 있다. 외부 라이브러리를 사용하게 될 경우, 라이브러리 버전에 따라 성능 차이가 발생할 수 있다. 또한 라이브러리 설정에 익숙치 않은 사용자는 소스코드 가동이 어려울 수 있다. 따라서 의존성을 제거하는 대신, 필요한 암호 모듈을 직접 삽입한다. 이때 사용하는 암호 모듈은 모든 알고리즘에 동일한 것을 사용하여 모듈로 인해 성능 차이가 발생하지 않도록 한다. 소스코드 가동 환경은 표 2와 같다.

Table. 2 Testing environment.

OS	Ubuntu 22.04
CPU	Ryzen 7 4800H
IDE	Visual Studio Code
Compiler	gcc 11.3.0
Optimization level	-O2

본 실험에서는 최적화 옵션을 -O2로 적용하였다. 이는 -O3 옵션을 사용하게 될 경우, 컴파일러의 판단으로 연산 과정이 일부 생략되거나 잘못될 가능성이 있다. 따라서 이러한 오류를 줄이기 위해 -O2 옵션을 적용하였다. KpqC 알고리즘은 공식 문서에서 알고리즘 성능을 제공하고 있으나, 이는 대부분 -O3를 적용한 결과물이다. 때문에 본 논문에서 제시하는 실험 결과는 공식 결과에 비해 다소 나쁘게 나올 수 있다. 각 알고리즘은 10,000회 반복한 값의 평균 값을 적용하였다.

3.2. 공개키 암호화 알고리즘

해당되는 알고리즘으로는 NTRU+[5], SMAUG[6], TiGER가[7] 존재한다. 각 알고리즘의 공식 문서에서 제시한 공개키, 비밀키, 암호문 크기는 표 3에서 확인할 수 있다.

Table. 3 Public key, Secret key, Ciphtertext size of lattice-based PKE/KEM algorithms. (Unit: byte)

Algorithm	PK size	SK size	CT size
NTRU+-576	864	1,728	864
NTRU+-768	1,152	2,304	1,152
NTRU+-864	1,296	3,168	1,296
NTRU+-1152	1,728	3,456	1,728
SMAUG-128	672	174	768
SMAUG-192	992	185	1,024
SMAUG-256	1,632	182	1,536
TiGER-128	480	528	768
TiGER-192	800	1,056	1,024
TiGER-256	928	1,056	1,152

공개키 크기와 암호문 크기는 세 알고리즘이 비슷하거나 공개키 부분에서 TiGER가 근소하게 우세하다 평가할 수 있으나, 비밀키의 경우에는 SMAUG가 명백하게 작은 크기를 지니고 있다.

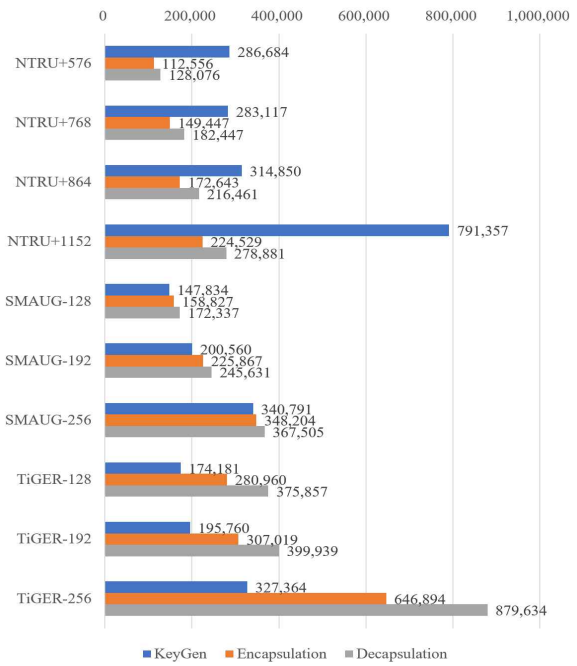

Figure. 1 Benchmark result of PKE/KEM candidates (Unit: clock cycles).

그림 1은 세 알고리즘의 성능 측정 결과를 그래프로 도식화한 것이다. 측정 결과 NTRU+의 암호화, 복호화 성능이 가장 우수한 것으로 측정되었다. 하지만 NTRU+는 키 생성 과정에서 비효율적인 연산 속도를 가졌다. 반면 SMAUG는 연산 종류에 상관없이 균일하게 우수한 지표를 제공했다. TiGER의 경우에는 중간 정도의 성능을 지닌다고 할 수 있다. 다만, TiGER-256의 복호화 과정에서 다소 느린 성능을 보여줬다.

3.3. 전자서명 알고리즘

이에 해당되는 알고리즘으로는 GCKSign[8], HAETAE[9], NCC-Sign[10], MQ-Sign[11], Peregrine[12], SOLMAE[13]가 존재한다. 각 알고리즘의 공식 문서에서 제시한 공개키, 비밀키, 암호문 크기는 표 4에서 확인할 수 있다.

Table. 4 Public key, Secret key, Ciphtertext size of lattice-based Digital signature algorithms (ori: original, con: conserparam, Unit: byte).

Algorithm	PK size	SK size	Sig. size
GCKSign-II	1,760	288	1,952
GCKSign-III	1,952	288	2,080
GCKSign-V	3,040	544	3,104
HAETAE-II	1,056	x	3,040
HAETAE-III	1,568	x	4,064
HAETAE-V	2,080	x	5,792
NCC(ori)-I	1,564	2,266	2,458
NCC(ori)-III	1,997	3,312	3,605
NCC(ori)-V	2,663	4,402	5,055
NCC(con)-I	1,984	2,800	3,186
NCC(con)-III	2,443	3,914	4,251
NCC(con)-V	3,091	4,940	5,385
MQ-72-46	328,411	15,561	134
MQ-112-72	1,238,761	37,729	200
MQ-148-96	2,892,961	66,421	260
Peregrine-512	897	1,281	666
Peregrine-1024	1,793	2,305	1,280
SOLMAE-512	896	x	666
SOLMAE-1024	1,792	x	1,375

가장 눈에 띄는 부분은 MQSign이다. MQSign은 가장 큰 공개키와 비밀키 크기를 가지고 있다. 반면 서명의 크기는 가장 작은 극단적인 설계를 가지고 있다. Peregrine은 Falcon과 완벽히 동일한 매개변수를 가지고 있으며 이는 Falcon의 설계를 추종한다고 할 수 있다. GCKSign은 비밀키가 가장 작으며 공개키 크기와 서명 크기는 중간 정도에 속한다. NCCSign은 매개변수 종류가 두 종류로 Original과 이것을 다소 변경한 Conserparam이 있다. Conserparam이 original보다 근소하게 큰 매개변수를 지닌다. HAETAE의 경우에는 비밀키 크기가 문서에 정확히 명시되지 않아서 판단할 수 없었다. 공개키 크기는 작은 편에 속한다.

표 5는 NCC의 두 버전을 포함하여 총 6종의 알고리즘의 성능 측정 결과를 나열하였다. SOLMAE의 경우에는 Makefile 작성 문제로 자미 제외하였다. 알고리즘 성능 격차가 커서 그래프 상으로는 잘 표현이 되지 않았기에 표로 명시한다. 전체적인 알고리즘 연산 속도가 공개키 암호화 알고리즘에 비해 느린 것을 확인할 수 있다. GCKSign는 연산 종류에 관계없이 대체로 안정적인 성능을 지니고 있다. 다만, II, III의 성능 격차가 뚜렷하지 않은 반면, V에서는 급격하게 성능이 떨어지는 모습을 보인다. HAETAE는 서명 생성이 매우 느린 모습을 보인다. NCCSign은 original이 conserparam보다 더 작은 매개변수를 가지고 있었으나, 연산 효율은 conserparam이 더 뛰어난 경향을 보인다. MQSign은 가장 작은 서명 크기를 가지고 있었으나, 키 생성이 매우 느리다는 단점을 보인다. 마지막으로 Peregrine은 키 생성은 다소 느린 편이나 서명 생성 속도와 매우 빠른 검증 속도를 가지고 있었다.

Table. 5 Benchmark result of Digital signature candidates (Unit: clock cycles).

Algorithm	Keygen	Sign	Verify
GCKSign-II	191,048	812,492	175,823
GCKSign-III	198,099	891,697	180,034
GCKSign-V	432,874	1,841,754	342,830
HAETAE-II	1,986,112	14,791,845	2,427,387
HAETAE-III	4,413,154	41,798,697	5,089,998
HAETAE-V	6,810,562	37,858,000	7,803,629
NCC(ori)-I	2,669,077	33,881,336	5,233,430
NCC(ori)-III	4,482,423	30,863,557	8,851,645
NCC(ori)-V	7,246,585	86,158,604	14,356,581
NCC(con)-I	1,891,678	16,626,486	3,723,436
NCC(con)-III	3,689,017	32,444,017	7,265,226
NCC(con)-V	6,279,118	43,207,193	12,437,221
MQ-72-46	95,242,213	521,368	1,470,313
MQ-112-72	490,303,519	1,501,506	5,203,085
MQ-148-96	1,492,894,919	3,172,366	12,054,509
Peregrine-512	12,502,738	330,561	37,475
Peregrine-1024	41,931,477	709,773	80,480

IV. 결론

본 논문에서는 KpqC 후보 알고리즘 중, 격자 기반 알고리즘의 벤치마크를 진행하였다. 모두 동일조건 하에서 벤치마크를 하였기에 안정적인 비교를 할 수 있었으며, 외부 의존성 제거로 누구나 손쉽게 알고리즘을 가동할 수 있는 환경을 준비하였다. 이후로 격자 기반 알고리즘 외에 모든 KpqC 알고리즘을 동일하게 벤치마크를 진행하며, 사용한 소스코드를 퍼블릭 도메인으로 공개하고자 한다.

V. Acknowledgement

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%) and this work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00627, Development of Lightweight BIoT technology for Highly Constrained Devices, 50%).

[참고문헌]

[1] R.Avanzi, J.Bos, L.Ducas, E.Kiltz, T.Lepoint, V.Lyubashevsky, J.M.Schanck, P.Schwabe, G.Seiler, and D.Stehlé, "CRYSTALS-Kyber algorithm specifications and supporting documentation," *NIST PQC Round,* 2(4), pp. 1-43, 2017.

[2] L.Ducas, E.Kiltz, T.Lepoint, VLyubashevsky, P.Schwabe, G.Seiler, and D.Stehlé, "Crystals-dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 238-268, 2018.

[3] P.A.Fouque, J.Hoffstein, P.Kirchner, V.Lyubashevsky, T.Pornin, T.Prest, T.Ricosset, G.Seiler, W.Whyte, and Z.Zhang, "Falcon: Fast-Fourier lattice-based compact signatures over NTRU," 36(5), pp 1-75, 2018.

[4] D.J.Bernstein, A.Hülsing, S.Kölbl, R.Niederhagen, J.Rijneveld, and P.Schwabe, "The SPHINCS+ signature framework," In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 2129-2146, Nov. 2019.

[5] J.Kim, and J.H.Park, "NTRU+: Compact Construction of NTRU Using Simple Encoding Method," *Cryptology ePrint archive*, 2022.

[6] J.H. Cheon, H.Choe, D.Hong, J.Hong, H.Seong, J.Shin, and M.Yi, "SMAUG: the Key Exchange Algorithm based on Module-LWE and Module-LWR," 2022.

[7] S.Park, C.G.Jung, A.Park, J.Choi, and H.Kang, "TiGER: Tiny bandwidth key encapsulation mechanism for easy miGration based on RLWE (R)," *Cryptology ePrint Archive*, 2022.

[8] J.Woo, K.Lee, and J.H.Park, "GCKSign: Simple and Efficient Signatures from Generalized Compact Knapsacks," *Cryptology ePrint archive*, 2022.

[9] J.H.Cheon, H.Choe, J.Devevey, T.Güneysu, D.Hong, M.Krausz, G.Land, J.Shin, D.Stehl´e, and M.Yi, "HAETAE: Hyperball bimodAl modulE rejecTion signAture schemE," 2022.

[10] K.A.Shim, J.Kim, and Y.An, "NCC-Sign: A New Lattice-based Signature Scheme using Non-Cyclotomic Polynomials," 2022.

[11] K.A.Shim, J.Kim, and Y.An, "MQ-Sign: A New Post-Quantum Signature Scheme based on Multivariate Quadratic Equations: Shorter and Faster," 2022.

[12] E.Y.Seo, Y.S.Kim, J.W.Lee, and J.S.No, "Peregrine: Toward Fastest FALCON Based on GPV Framework," *Cryptology ePrint Archive*, 2022.

[13] SOLMAE Algorithm Specifications, [Online]: https://kpqc.or.kr/images/pdf/SOLMAE.pdf