Open Quantum Safe 프로젝트 동향 및 벤치마크

김현준*, 서화정**

*한성대학교 (대학원생)

*한성대학교 (교수)

Open Quantum Safe project trends and benchmarks

Hyun-Jun Kim*, Hwa-Jeong Seo**

*Hansung University(Graduate student)

**Hansung University(profesor)

요 약

양자 컴퓨팅의 발전은 전통 암호 체계에 도전을 제기하고 있으며, 이에 따라 전 세계적으로 양자 내성 암호화(Post-Quantum Cryptography, PQC)에 대한 관심이 증가하고 있다. Open Quantum Safe(OQS) 프로젝트는 다양한 PQC 알고리즘을 국제 표준화 과정에 맞춰 구현하고 있다. 본 연구에서는 OQS 프로젝트의 최신 업데이트와 벤치마킹 결과를 분석하여, x86_64, M1, aarch64 플랫폼에서의 알고리즘 성능을 평가하였다. 특히, 플랫폼별 높은 성능을 보이는 알고리즘에서 차이를 보였다. 이는 플랫폼별 최적화의 차이로 보이며, PQC 기술의 지속적인 업데이트가 중요함을 강조한다.

I. 서론

최근 양자 컴퓨팅 기술의 급속한 발전이 전통적인 암호 체계에 큰 도전을 제기하고 있다[1]. 이에 따라, 양자 내성 암호화(Post-Quantum Cryptography, PQC) 기술에 대한 전 세계적인 관심이 증가하고 있다. 이러한 기술 전환을 지원하고자 Open Quantum Safe(OQS) 프로젝트가 진행되고 있으며, 이 프로젝트는 국제 표준화 과정을 거치고 있는 다양한 알고리즘을 포함하여 포괄적인 포스트 양자 암호화 알고리즘 세트를 구현하고 있다[2]. 본 연구는 Open Quantum Safe 프로젝트의 최신 업데이트와 벤치마크 결과를 분석함으로써 이 기술들의 현재 성능을 평가하고자 한다.

II. Open Quantum Safe

OQS 프로젝트는 2014년 Michele Mosca와 Douglas Stsebila에 의해 시작되었다.

2024년 1월 23일 OQS 프로젝트는 Linux Foundation에 합류하였다[3]. Linux Foundation이 주도하는 Post-Quantum Cryptography Alliance (PQCA)에 참여함으로써, OQS 프로젝트는 글로벌 기술 커뮤니티와의 협력을 강화하고, 양자 내성 암호화 기술의 연구, 개발 및 표준화 작업을 가속화 중이다.

OQS는 두 가지 주요 작업으로 양자 저항성 암호화 알고리즘을 위한 오픈 소스 C 라이브러리인 liboqs 와 널리 사용되는 OpenSSL 라이브러리를 포함하여 프로토콜에 대한 프로토타입 통합으로 구성된다.

2.1 liboqs

OQS프로젝트의 오픈 소스 C 라이브러리 liboqs는 다양한 양자 내성 암호화 알고리즘을 지원한다. 주로 NIST의 양자 내성 암호화 표준화 프로세스를 통해 선별 및 검토된 알고리즘을 기반으로 한다.
Key encapsulation mechanisms (KEMs) 알고리즘으로 NIST 선정 Kyber, Round 4 후보군 Classic McEliece, BIKE, HQC 기타 알고리즘으로 FrodoKEM, NTRU-Prime를 제공한다. 서명 알고리즘으로 NIST 선정 알고리즘 Dilithium, Falcon, SPHINCS+을 제공

한다. 현재 Kyber and Dilithium는 FIPS 버전인 ML-KEM, ML-DSA의 추가적으로 제공한다[]. 또한 다른 프로그램 언어를 위한 C++, Go, Java, .Net, Python, Rust의 다양한 래퍼 제공 하고 있다.

2.2 OQS-Provider

OQS-Provider는 Open Quantum Safe라이브러리와 OpenSSL를 통합하여 PQC 알고리즘을 지원한다. OpenSSLv3의 플러그인 방식의 알고리즘 추가를 통해 기존 OpenSSL 1.1.1보다 깔끔한 알고리즘의 확장을 제공한다. liboqs에 포함된 모든 양자안전암호화(QSC) 키 교환 메커니즘(KEM)과 서명 알고리즘, 그리고 하이브리드 고전/QSC 알고리즘 쌍을 지원한다. 표준 OpenSSL 도구 세트 사용과 통합되며, QSC 서명 알고리즘을 X.509 표준에도 적용 가능하다.

III. 벤치마크

OQS의 사이트에서는 벤치마크 결과를 제공한다. KEM, 서명 알고리즘의 런타임, 메모리사용량, 그리고 OpenSSL에서의 성능과 TLS 핸드쉐이크 성능을 보여준다. 그러나 현재 유지 관리되지 않고 있다. ML-KEM, ML-DSA 알고리즘은 포함되어있지 않으며, TLS 핸드셰이크 성능은 PQC 알고리즘중 kyber만 측정되어 있다. 이러한 부분을 고려하여 본 논문에서는 OpenSSL 상에서의 성능과 TLS 핸드쉐이크 성능을 중점적으로 살펴보았다.

3.1 환경

Intel NUC(Intel i5-8259U, x86_64), 맥북 에어(m1), 라즈베리파이5 (ARM Cortex-A76, aarch64)를 대상으로 측정하였다. OQS-Provider를 사용하여 OpenSSL에서의 성능과 TLS 핸드쉐이크 성능을 측정하고 비교하였다. 각각 openssl speed, openssl s_time을 사용하였다.

3.2 KEM 성능 비교

표 1의 세 장치에 대한 openssl speed 결과에 따르면, 사용하는 디바이스에 따라 암호화 알고리즘의 성능 차이를 보였다. 특히, x86_64 플랫폼에서 mlkem 알고리즘은 뛰어난 성능을 보여주었다. 반면, 다른 디바이스에서는 kyber 알고리즘이 일반적으로 다른 알고리즘보다 높은 성능을 보였다.

특히, kyber와 hqc 알고리즘은 세 가지 디바이스(x86_64, m1, aarch64)에서 다른 알고리즘과 비교하였을 때 일관된 성능을 보여주었다. 그러나, mlkem과 bike 알고리즘은 m1과 aarch64 디바이스에서 성능이 크게 감소하는 것을 보여주어, 해당 환경에서 최적화가 덜 되어 있을 수 있음을 보였다.

표 1. 세 장치에 대한 KEM 알고리즘의 openssl speed 측정 결과

	x86_64			m1			aarch64		
	keygen/s	encaps/s	decaps/s	keygen/s	encaps/s	decaps/s	keygen/s	encaps/s	decaps/s
kyber512	82039.5	97888.8	132064.9	89761.8	82067.4	102312.3	40697.9	41906.4	46369.6
kyber768	59243.4	65130.1	84514.2	61974.9	54381.4	64478.6	28506.3	26928.9	29293.5
kyber1024	46624.8	47630.5	59335.0	47662.3	41071.7	48103.6	19432.6	18133.3	19290.4
mlkem512	89676.4	131569.9	139916.3	32117.1	30092.0	23781.9	21562.0	19571.8	15077.5
mlkem768	61919.0	84379.6	87376.7	19653.8	18760.3	15573.6	13292.8	12559.2	9966.5
mlkem1024	48479.7	59496.4	60831.5	12689.4	12751.9	10752.4	8783.6	8457.9	6924.4
bikel1	4313.6	26004.2	919.8	93.0	1730.2	120.3	50.5	966.9	60.6
bikel3	1498.3	11062.7	334.5	29.9	523.0	38.6	16.3	314.9	19.4
bikel5	567.8	5114.4	104.3	11.9	227.6	15.5	6.5	125.3	7.7
hqc128	578.4	287.8	189.3	1322.0	653.0	428.2	447.8	223.5	147.2
hqc192	189.4	94.7	62.8	437.5	218.4	144.2	141.8	72.3	48.7
hqc256	103.7	51.6	34.2	240.0	119.4	78.8	80.5	40.1	26.6

표 2. 세 장치에 대한 서명 알고리즘의 openssl speed 측정 결과

	x86_64			m1			aarch64		
	keygens/s	sign/s	verify/s	keygens/s	sign/s	verify/s	keygens/s	sign/s	verify/s
dilithium2	33996.0	15197.0	41114.4	27679.4	9748.4	30742.6	12219.4	4172.5	12845.4
dilithium3	20756.6	9425.4	24523.2	13033.3	6315.5	19975.9	6536.5	2683.0	7797.8
dilithium5	13769.7	7583.7	15479.5	11271.2	5200.9	12333.6	4389.2	2078.5	4562.1
mldsa44	34410.1	15168.5	41197.6	12319.2	2999.9	12060.7	8784.6	1941.0	8226.6
mldsa65	21217.7	9287.0	24510.1	6456.1	1756.6	7345.9	4978.9	1195.7	5172.5
mldsa87	13682.7	7506.8	15229.1	4480.1	1467.3	4432.1	3256.5	947.4	3053.1
falcon512	147.7	4107.6	24078.9	119.7	4034.0	25981.8	95.8	3403.0	19982.5
falcon1024	45.7	2046.4	11867.8	37.5	2020.3	13143.3	32.0	1693.5	10237.6

표 3. Intel NUC(Intel i5-8259U, x86_64)에서 openssl s_time 측정 결과

	dilithium2	dilithium3	dilithium5	mldsa44	mldsa65	mldsa87	falcon512	falcon1024
kyber512	2522.59	2381.90	2090.07	2391.59	2297.30	2052.99	2245.08	1871.06
kyber768	2049.12	1868.26	1816.50	1999.57	1847.13	1728.45	1857.13	1608.70
kyber1024	1906.42	1851.87	1709.58	1919.55	1769.38	1647.99	1756.58	1579.58
mlkem512	2090.09	1957.09	1807.29	2026.66	1958.54	1775.46	1933.41	1607.03
mlkem768	1984.67	1873.73	1689.48	1968.90	1874.86	1675.01	1804.37	1527.28
mlkem1024	1934.62	1819.17	1683.97	1795.33	1804.61	1667.19	1768.80	1572.50
bikel1	513.61	523.77	501.31	518.32	513.99	492.33	518.71	488.47
bikel3	229.15	226.73	223.40	226.92	224.90	223.75	225.33	220.38
bikel5	80.26	80.20	80.41	80.01	79.88	79.81	79.76	77.60
hqc128	126.72	125.83	124.73	125.74	125.77	124.19	126.16	123.70
hqc192	44.68	44.76	44.69	44.59	44.62	44.61	44.45	44.28
hqc256	25.07	24.82	25.03	24.53	25.00	25.00	24.57	24.99

표 4. 맥북 에어(m1)에서 openssl s_time 측정 결과

	dilithium2	dilithium3	dilithium5	mldsa44	mldsa65	mldsa87	falcon512	falcon1024
kyber512	3812.63	3408.02	3062.03	3150.20	2641.45	2061.08	3693.40	3033.84
kyber768	3046.98	2816.87	2539.70	2637.64	2255.14	1841.54	2985.86	2539.35
kyber1024	2934.57	2706.17	2476.16	2561.38	2202.65	1804.90	2859.80	2453.74
mlkem512	2702.29	2516.96	2290.29	2367.98	2062.01	1714.09	2606.04	2285.00
mlkem768	2408.03	2292.72	2103.18	2143.58	1873.16	1588.30	2349.18	1994.78
mlkem1024	2110.70	1993.28	1856.25	1875.40	1678.41	1441.06	2036.05	1826.86
bikel1	51.63	51.55	51.43	51.47	51.29	51.05	51.59	51.34
bikel3	16.75	16.74	16.66	16.72	16.71	16.68	16.74	16.73
bikel5	6.70	6.70	6.70	6.70	6.70	6.69	6.70	6.70
hqc128	294.38	292.12	289.16	289.82	284.61	277.15	293.21	288.50
hqc192	104.41	104.25	103.78	103.86	103.20	102.15	104.30	103.71
hqc256	58.08	58.03	57.84	57.74	57.67	57.29	57.89	57.86

3.3 서명 성능 비교

전반적으로 동일한 보안강도에서 Dilithium은 Falcon보다 높은 성능을 보였다. 특징적으로 Dilithium과 mlkem는 x86_64에서 유사한 성능을 보였으나, m1과 aarch64에서는 Dilithium이 mlkem 보다 높은 성능을 보였다. Falcon 알고리즘은 특히 다른 플랫폼에 비해 aarch64 플랫폼에서 검증을 더 잘 처리하였다.

3.4 Handshake 성능 비교

x86_64환경에서 동일한 보안강도에서 비교하였을 때, KEM 알고리즘에서는 kyber mlkem, bike, hqc 순서, 서명 알고리즘에서는 Dilithium, mldsa, Falcon 순서로 높은 성능을 보였다.

반면에, m1과 aarch64에서 동일한 보안강도

표 5. 라즈베리파이5 (ARM Cortex-A76, aarch64)에서 openssl s_time 측정 결과

	dilithium2	dilithium3	dilithium5	mldsa44	mldsa65	mldsa87	falcon512	falcon1024
kyber512	1235.42	1106.00	945.85	831.31	820.63	721.75	1251.23	1057.29
kyber768	1018.65	894.23	800.33	784.65	716.49	621.71	963.22	892.53
kyber1024	963.34	862.80	771.07	728.75	706.79	588.10	958.39	855.02
mlkem512	950.91	872.87	758.18	753.20	696.87	599.93	960.57	864.32
mlkem768	869.41	807.81	731.64	710.73	649.03	575.83	880.12	830.04
mlkem1024	804.57	750.43	668.50	665.92	612.29	533.56	781.30	746.73
bikel1	26.21	25.90	25.95	25.80	25.84	25.85	25.99	26.01
bikel3	8.71	8.70	7.99	8.58	8.74	8.67	8.71	8.68
bikel5	3.51	3.51	3.45	3.51	3.52	3.51	3.51	3.51
hqc128	86.32	85.84	84.95	85.08	85.20	84.04	86.07	85.68
hqc192	34.40	34.18	34.28	34.22	33.87	33.49	34.16	34.36
hqc256	18.94	19.03	18.99	18.89	18.93	18.98	19.05	18.94

에서 비교 하였을 때, KEM 알고리즘에서는 kyber, mlkem, hqc, bike 순서, 서명 알고리즘에서는 Dilithium, Falcon, mldsa 순서로 높은 성능을 보였다. 이는 표 1에서의 openssl speed 결과와 같이 mlkem과 bike 알고리즘은 m1과 aarch64 디바이스에서 성능이 낮은 결과로 보인다.

결과적으로 x86_64과 M1 환경에서는 Dilithium과 kyber의 조합이 높은 성능을 보였다. aarch64에서는 근소한 차이로 Falcon과 kyber의 조합이 Dilithium과 kyber의 조합보다 높은 성능을 보였다. 가장 낮은 성능을 보인 조합으로는 x86_64에서는 Falcon과 HQC조합이 가장 낮은 성능을 보였다. M1과 aarch64에서는 mldsa과 bike의 조합에서 낮은 성능을보였다.

IV. 결론

본 논문에서는 Open Quantum Safe 프로젝트의 최신 업데이트의 확인과 x86_64, m1과 aarch64의 다양한 환경에서 벤치마크 결과를 측정하고 분석하였다. x86_64과 M1 환경에서는 Dilithium과 kyber의 조합이 높은 성능을 보였으며 aarch64에서는 Falcon과 kyber의 조합이 높은 성능을 보였다. 이러한 차이는 x86_64에서의 최적화 보다 다른 환경에서의 최적화가 덜 되어있음으로 보여진다. 새로 업데이트 된 mlkem와 mldsa의 경우 기존 3라운드의 알고리즘보다 비교적 낮은 성능을 보이나 다른 PQC알고리즘 보다 높은 성능을 보여주었다.

현재 양자 컴퓨터의 등장을 대비하여 PQC의 최신 업데이트가 활발하게 진행되고 있다. PQC알고리즘은 보안은 연구 진행됨에 따라 빠르게 변화할 수 있다. OQS 프로젝트의 결과가 적절한 환경에 따른 알고리즘 선택을 위해 큰 도움이 될 것으로 사료된다.

V. Acknowledgment

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] Shor, Peter W., Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Comput., 26 (5): 1484 - 1509, 1997.

[2] Stebila, Douglas, and Michele Mosca. Post-quantum key exchange for the internet and the open quantum safe project. International Conference on Selected Areas in Cryptography. Cham: Springer International Publishing, 2016.

[3] Post-Quantum Cryptography Alliance LatThe Linux Foundation, unches to Advance Post-Quantum Cryptography https://www.linuxfoundation.org/press/announcing-the-post-quantum-cryptography-alliance-pqca, Fab, 2024