정 보 보 호 학 회 지
제 33 권 제3호, 2023. 6

# KpqC 암호 알고리즘의 효율성 관점에서의 분석

권 혁 동*, 심 민 주*, 송 경 주*, 이 민 우**, 서 화 정***

요 약

미국 국립표준기술연구소에서는 양자 컴퓨터가 가져올 현대 암호 체계 붕괴에 대비하여 양자내성암호 표준화 공모전을 개최하였고, 공개키 암호화 알고리즘 1종, 전자서명 알고리즘 3종을 표준으로 선정하였다. 이어서 국내에서도 국내 표준 양자내성암호 도입을 위한 시도가 시작되었다. KpqC 공모전은 국산 양자내성암호 알고리즘 표준화를 선정하기 위한 공모전으로, 현재 Round 1을 진행 중에 있다. 본 고에서는 KpqC 암호 알고리즘들의 암호 구현 효율성 관점에서 비교 분석한다. 이와 더불어 NIST에서 선정한 표준 알고리즘들과 실제 활용 사례를 확인해 보며 추후 표준화가 될 KpqC 암호 알고리즘들의 운영 방안에 대해 알아보도록 한다.

## Ⅰ. 서 론

양자컴퓨터는 0과 1로 데이터를 표현하는 고전컴퓨터와는 다르게 양자역학의 원리에 따라 중첩 데이터 표현이 가능하다. 이러한 특징으로 양자컴퓨터는 특정 문제에 있어서 연산 성능이 고전컴퓨터에 비해 매우 뛰어나다. 특히 일부 검색 알고리즘 또는 소인수 분해 알고리즘의 성능이 우수하며 이는 Grover 알고리즘[1] 또는 Shor 알고리즘을 [2] 포함한다. 이러한 양자 알고리즘은 각각 대칭키, 공개키 암호화 알고리즘에 대한 해킹에 효과적이기 때문에 이를 대체하고자하는 안전하게 보완하기 위한 방안들이 제시되고 있다.

미국의 국립표준기술연구소(National Institute of Standards and Technology, NIST)에서는 양자내성암호 표준화 공모전을 개최하여, 양자내성암호의 원활한 도입을 계획하였다. 한국에서도 양자내성암호 국내 표준 선정을 위한 노력이 시작되어 진행 중에 있다.

본 고에서는 한국형 양자내성암호 표준화 공모전인 KpqC에 대해서 소개한다. KpqC에 투고된 암호 알고리즘들을 소개하며, 주로 효율성 관점에서 이를 분석한다. 본 고의 구성은 다음과 같다. 2장에서 KpqC 공모전과 투고된 일부 알고리즘에 대해서 소개한다. 3장에서 KpqC 공모전의 알고리즘의 매개변수 위주로 분석을 한다. 4장에서 본 고의 결론을 맺는다.

## Ⅱ. KpqC

### 2.1. 공모전 개요

양자컴퓨터가 개발되었을 때 현대 암호 체계가 붕괴될 것을 우려하여 양자내성암호 표준을 선정해야 한다는 목소리가 점차 커져가고 있다. 국내에서는 양자내성암호연구단을 조직되어 현재 한국형 양자내성암호 표준화 공모전 KpqC를 개최하여 운영 중에 있다. KpqC 공모전의 타임라인은 [표 1]과 같다. 2021년 말, KpqC 공모전이 개최되었고 2022년 말에 Round 1 선정 결과가 발표되었다[3].

KpqC에서는 다양한 평가 기준을 제시하고 있으며, 이는 다음과 같다.

〔표 1〕 KpqC 진행 일정

| Item | Date |
|---|---|
| Holding a KpqC | 2021. 11. |
| Submission deadline | 2022. 10. |
| Round 1 announcement | 2022. 12. |
| Round 2 announcement | 2023. 12. |
| Final selection announcement | 2024. 09. |

* 한성대학교 정보컴퓨터공학과 (대학원생, korlethean@gmail.com, minjoos9797@gmail.com, thdrudwn98@gmail.com)
** 한성대학교 IT융합공학부 (대학원생, minunejip@gmail.com)
*** 한성대학교 융합보안학과 (부교수, hwajeong84@gmail.com)

- 안전성: 제안된 알고리즘은 안전성을 확보해야 한다. 이를 위해 안전성 정의에 대한 증명이 필요하다. 이때 정성적인 평가가 아닌, 정량적인 수치를 제공하여 안전성을 증명해야 한다. 또한 안전성의 범위로는 양자내성을 지닐 뿐만 아니라, 고전컴퓨터 상에서의 공격에서도 안전함을 포함해야 한다.
- 효율성: 제안된 알고리즘은 충분히 효율적으로 구현되어야 한다. 알고리즘을 구성하는 구성 요소, 즉 매개변수의 크기가 합리적이여야 한다. 또한 일부 알고리즘은 구조상 복호화 실패 확률을 지니고 있다. 이 확률을 정확히 명시해야 하며, 복호화가 실패하였을 때 대처 방안에 대해서 제시하여야 한다. 모든 알고리즘은 레퍼런스 구현물을 제출하면서 해당 구현물의 동작 효율성 정보도 제시해야 한다. 여기에는 매개변수 크기 뿐만 아니라 알고리즘이 연산을 종료하는데 필요한 시간 정보도 포함된다.
- 활용성: 제시된 알고리즘은 특정 시스템, 플랫폼에 구애받지 않고 다양한 환경에서 가동되어야 한다. 이를 위해서는 알고리즘이 가지는 의존성을 최대한 제거하여 모든 시스템에서 동등하게 동작할 수 있도록 해야한다. 구현 과정에서는 일부 외부 라이브러리를 포함할 수 있다. 하지만 이러한 라이브러리가 다양한 환경을 지원하지 못한다면 이 역시 환경 의존성을 갖추게 되므로 최소한의 의존성을 갖출 것을 권장한다.
- 독창성: 제안된 알고리즘은 다른 알고리즘에서 볼 수 없는 창의적인 구조를 가질 것을 지향한다. 이는 기존 투고된 알고리즘이 다시 투고되는 것을 방지하며, 한국의 양자내성암호 표준으로 선정되어 대표적인 자격을 지니기 때문에 독창적인 구조를 통해 국내 암호 산업의 발전을 보여줄 수 있게 한다.
- 기타: 기반 문제가 신뢰할 수 있는 문제인지 알고리즘의 특징도 평가 기준에 포함된다.

심사 결과, Round 1에서 공개키 암호화 알고리즘 후보로 7종이 선정되었고, 전자서명 알고리즘 후보로 9종이 선정되었다. 선정된 알고리즘의 목록은 [표 2]와 같다. 선정된 알고리즘 중에서 공개키 암호화 알고리즘은 부호 기반이 3종, 격자 기반이 3종, 그리고 기타 1종이다. 전자서명 알고리즘은 부호 기반이 1종, 격자 기반이 5종, 아이소지니 기반이 1종, 다변수 다항식 기반 1종 그리고 기타 1종이다. 전체 알고리즘 중에서 격자 기반 알고리즘의 수가 가장 많다. 이는 NIST의 표준 선정에서도 비슷한 경향성을 보이는데, 실제로 NIST에서 선정한 표준 알고리즘 4종 중 3종인 Kyber[4], Dilithium[5], Falcon[6]은 모두 격자 기반 알고리즘이다. 여기서는 일부 격자기반 KpqC 후보를 확인한다.

〔표 2〕 1 라운드 선정 알고리즘

| Type | Public Key Encryption-Key Establishment Mechanism | Digital Signature |
|---|---|---|
| Code | Layered ROLLO<br>PALOMA<br>REDOG | Enhanced pqsigRM |
| Lattice | NTRU+<br>SMAUG<br>TiGER | GCKSign<br>HAETAE<br>NCC-Sign<br>Peregrine<br>SOLMAE |
| Multivariate Quaratic | - | MQ-Sign |
| Isogeny | - | FIBS |
| Others | IPCC | AIMer |

## 2.2. 격자기반 공개키 암호화 및 전자 서명

### 2.2.1. SMAUG

SMAUG는 Module-Learning with Errors (M-LWE), Module-Learning with Rounding (M-LWR)을 사용하는 격자 기반 공개키 암호화 알고리즘이다 [8]. SMAUG는 IND-CPA 보안성 확보를 위해서 M-LWE와 M-LWR의 보안강도에 의존하는데, 이를 통해서 IND-CCA2 보안성까지 만족시켰다. 일반적으로 격자 기반 알고리즘은 Ring-LWE(R-LWE)와 Ring-LWR(R-LWR)을 사용하는데, SMAUG는 M-LWE, M-LWR을 사용하였다. SMAUG 연구진은 Ring 구조보다 Module 구조가 확장에 훨씬 뛰어나며, 매개변수 조절로 보안성과 효율성의 미세한 조율이 가능하기 때문이라 밝혔다. 샘플링 과정에는 M-LWR을 사용한다. 샘플링 도중에 비밀 값이 입력되는데, 이 비밀 값 재사용을 적게 하는 것으로 M-LWR 문제를 M-LWE 문제로 간주할 수 있다. 보안성이 M-LWR에

〔표 3〕 SMAUG의 파라미터와 성능.

| Scheme | SMAUG 128 | SMAUG 192 | SMAUG 256 |
|---|---|---|---|
| n | 256 | | |
| k | 2 | 3 | 5 |
| (p, q) | (1024, 256) | | |
| Public key size | 672-byte | 992-byte | 1632-byte |
| Secret key size | 174-byte | 185-byte | 182-byte |
| Cipher size | 768-byte | 1024-byte | 1536-byte |
| Keygen time | 73584 -cycles | 106956 -cycles | 191268 -cycles |
| Encryption time | 81684 -cycles | 115128 -cycles | 200520 -cycles |
| Decryption time | 88920 -cycles | 124812 -cycles | 210240 -cycles |

의존하기 때문에 발생한 특징으로 다른 알고리즘에 비해 암호문의 길이가 짧다. 이는 M-LWE의 인스턴스 대비 log q / log p의 크기로 암호문의 크기를 줄였기 때문이다.

격자 기반 암호 알고리즘의 단점 중 하나는 복호화 실패 가능성이 있다는 것이다. SMAUG 역시 M-LWE, M-LWR을 사용하기 때문에 복호화 과정에서 오류가 발생한다. SMAUG 연구진은 매개변수를 조절하는 것으로 보안성과 효율성, 오류 발생 확률 세 가지 요소가 절충될 수 있게 하였다고 밝혔다. 이것으로 SMAUG의 오류 발생 확률은 무시 가능할 정도로 낮아졌지만, 여전히 복호화 실패 가능성은 존재한다. 따라서 SMAUG는 복호화가 실패할 경우 비밀 값과 암호문의 해시 값을 반환하여 Fujisaki-Okamoto variant 변환으로 QROM의 보안성을 제공하도록 하였다. [표 3]은 SMAUG의 매개변수와 성능을 표시한다. 성능 측정은 AMD Ryzen 7 3700X 프로세서를 사용하였으며 난수 생성, 데이터 패킹과 같은 부분도 성능 측정에 포함되었다. 성능 값은 10,000회 반복 연산하여 획득한 값들 중에서 중앙 값을 사용하였다.

### 2.2.2. GCKSign

GCKSign은 격자 기반 전자서명 알고리즘으로 Discrete Gaussian Distribution에 대한 샘플링이 필요하지 않다[9]. 이러한 특징은 NIST 양자내성암호 표준화 선정 알고리즘인 Dilithium과 Falcon보다 간단하고 효율적이다. GCKSign은 Generalized knapsacks problem을 사용한다. 하지만 이는 서명 체계의 매개변수 효율을 나쁘게 만드는 보안 요구사항이 있다 [10]. 따라서 개량한 Target-modified one-wayness(TMO) 문제를 사용하였다. 이것으로 GCKSign은 Dilithium보다 더 작은 크기의 서명 및 비밀키 크기를 가진다. 서명과 비밀키 크기가 작기 때문에 부채널 내성을 효율적으로 구현할 수 있으며, 부채널 내성을 확보하면서 동시에 속도 저하도 적게 발생한다. 또한 비밀키가 유출 될 수 있는 구현 실수도 쉽게 찾아낼 수 있다. 이를 위해서 Gaussian sampling을 사용하지 않고 Uniform sampling을 사용한다. 키와 서명 크기를 줄이기 위해서 Central Limit Theorem(CLT)와 오류 함수 분석을 사용하였다. 따라서 GCKSign은 서명 체계가 단순해졌기에 빠르고 효율적인 구현이 가능하고 키 관리가 편리하게 설계되었다. [표 4]는 GCKSign에서 사용하

〔표 4〕 GCKSign의 파라미터와 성능.

| Scheme | GCKSign II | GCKSign III | GCKSign V |
|---|---|---|---|
| n | 256 | 256 | 512 |
| q | $2^{54} - 10751$ | $2^{60} - 2559$ | $2^{44} - 7167$ |
| m | 4 | 4 | 3 |
| h | 39 | 45 | 44 |
| challenge space | 192 | 212 | 256 |
| $B$ | $2^{14} - 1$ | $2^{14} + 2^{9}$ | $2^{15} - 1$ |
| $\eta$ | 1 | 1 | 1 |
| $L_S$ | 18 | 18 | 19 |
| Public key size | 1760-byte | 1952-byte | 3040-byte |
| Secret key size | 288-byte | 288-byte | 544-byte |
| Keygen time | 184 -Kcycles | 202 -Kcycles | 265 -Kcycles |
| Sign time | 1062 -Kcycles | 1240 -Kcycles | 1421 -Kcycles |
| Verify time | 237 -Kcycles | 253 -Kcycles | 373 -Kcycles |

는 매개변수의 크기와 연산 성능을 나타낸다. 성능 측정시에는 인텔 코어 i7-8700k 3.7GHz 프로세서를 사용하였고 Ubunru 20.04 LTS OS 상에서 가동하였다.

## Ⅲ. NIST 표준 알고리즘과의 비교

NIST 표준 알고리즘에는 공개키 암호화 알고리즘 Kyber, 전자서명 알고리즘 Dilithium, Falcon, 그리고 SPHINCS+[11]가 있다. 본 장에서는 NIST 표준 알고리즘과 KpqC 알고리즘의 매개변수 위주로 비교 및 실제 성능에 대해서 확인해 보도록 한다.

### 3.1. 공개키 암호화 알고리즘

공개키 암호화 알고리즘은 NIST 후보군에 1종류 밖에 없으며, 현재 추가 선정을 위해 Round 4가 진행되고 있다. Round 4의 알고리즘으로는 BIKE[12], Classic McEliece[13], HQC[14]가 있다. 여기에는 SIKE도 있지만[15], 취약점이 보고됨으로써 탈락하여 비교 선상에서 제외하였다[16]. 본 고에서는 알고리즘들의 연산 속도는 비교하지 않는다. 그 이유는 각 알고리즘마다 성능 측정한 환경이 다르고 성능 측정 기준도 전부 다르기 때문이다. 그렇기에 객관적인 비교 분석이 어려우므로 성능 측정 결과는 나열하지 않는다.

[표 5]는 KpqC Round 1 후보, NIST 표준화 및 Round 4 후보 공개키 암호화 알고리즘들의 공개키 크기를 나열한 것이다. 공개키 크기를 확인한 결과, TiGER[17] 알고리즘이 가장 작은 공개키 크기를 가지고 있음을 알 수 있다. NIST 표준인 Kyber와 비교를 한다면, 상당수의 알고리즘이 표준과 유사하거나 더 작은 크기를 지니고 있음을 알 수 있다. Kyber를 기준으로 이상적인 공개키 크기를 정한다면 대략 1KB 까지로 볼 수 있다. 이를 기준으로 본다면, IPCC와 PALOMA[18] 알고리즘을 제외한 나머지 모든 공개키 암호화 알고리즘은 안정적인 공개키 크기를 가지고 있다고 할 수 있다. IPCC는 특이하게 스킴에 따라 공개키 크기가 달라지지 않고 모두 동일한 것을 알 수 있다. PALOMA의 경우에는 다른 알고리즘들에 비해 공개키 크기가 매우 큰 편이다.

동일하게 [표 6]에서는 공개키 암호화 알고리즘들의 비밀키 크기를 나열하였다. 가장 작은 비밀키 크기를 가지는 알고리즘은 Layered ROLLO[19]이며 특이하게도 스킴에 따라 비밀키 크기가 달라지지 않았다.

〔표 5〕 PKE/KEM의 공개키 크기. 볼드체는 KpqC 후보군.

| Rank | PKE/KEM scheme | Public key size (Byte) |
|---|---|---|
| **1** | **TiGER-128** | **480** |
| **2** | **SMAUG-128** | **672** |
| 3 | KYBER-512 | 800 |
| **4** | **NTRU+-576** | **864** |
| **5** | **SMAUG-192** | **992** |
| **6** | **TiGER-192** | **800** |
| **7** | **TiGER-256** | **928** |
| **8** | **NTRU+-768** | **1,152** |
| 9 | KYBER-768 | 1,184 |
| **10** | **Layered ROLLO-128** | **1,240** |
| **11** | **NTRU+864** | **1,296** |
| 12 | KYBER-1024 | 1,568 |
| **13** | **SMAUG-256** | **1,632** |
| **14** | **Layered ROLLO-192** | **1,699** |
| **15** | **NTRU+-1152** | **1,728** |
| 16 | HQC-128 | 2,249 |
| **17** | **Layered ROLLO-256** | **2,571** |
| **18** | **IPCC-f1** | **4,800** |
| | **IPCC-f3** | **4,800** |
| | **IPCC-f4** | **4,800** |
| 21 | HQC-192 | 4,522 |
| 22 | HQC-256 | 7,245 |
| 23 | BIKE-I | 12,323 |
| **24** | **REDOG-1** | **14,250** |
| 25 | BIKE-III | 24,659 |
| **26** | **REDOG-2** | **32,840** |
| 27 | BIKE-V | 40,973 |
| **28** | **REDOG-3** | **62980** |
| 29 | mceliece348864 | 261,120 |
| **30** | **PALOMA-128** | **319,488** |
| 31 | mceliece460896 | 524,160 |
| **32** | **PALOMA-192** | **812,032** |
| **33** | **PALOMA-256** | **1,025,024** |
| 34 | mceliece6688128 | 1,044,992 |
| 35 | mceliece6960119 | 1,047,319 |
| 36 | mceliece8192128 | 1,357,824 |

IPCC도 동일한 성질을 가지는 것을 확인할 수 있다. 대부분의 알고리즘들이 공개키 크기에 비하면 더 작은 비밀키 크기를 가지고 있다. 양자내성암호 표준으로

〔표 6〕 PKE/KEM의 비밀키 크기. 볼드체는 KpqC 후보군.

| Rank | PKE/KEM scheme | Secret key size (Byte) |
|---|---|---|
| 1 | **Layered ROLLO-128** | **120** |
| | **Layered ROLLO-192** | **120** |
| | **Layered ROLLO-256** | **120** |
| **4** | **SMAUG-128** | **174** |
| **5** | **SMAUG-256** | **182** |
| **6** | **SMAUG-192** | **185** |
| 7 | **IPCC-f1** | **400** |
| | **IPCC-f3** | **400** |
| | **IPCC-f4** | **400** |
| **10** | **TiGER-128** | **528** |
| 11 | **TiGER-192** | **1,056** |
| | **TiGER-256** | **1,056** |
| **13** | **REDOG-1** | **1,450** |
| 14 | KYBER-512 | 1,632 |
| **15** | **NTRU+-576** | **1,728** |
| 16 | BIKE-I | 2,244 |
| 17 | HQC-128 | 2,289 |
| **18** | **NTRU+-768** | **2,304** |
| 19 | KYBER-768 | 2,400 |
| **20** | **REDOG-2** | **2,520** |
| **21** | **NTRU+-864** | **3,168** |
| 22 | KYBER-1024 | 2,592 |
| 23 | BIKE-III | 3,346 |
| **24** | **NTRU+-1152** | **3,456** |
| **25** | **REDOG-3** | **3,890** |
| 26 | HQC-192 | 4,562 |
| 27 | BIKE-V | 4,640 |
| 28 | mceliece348864 | 6,492 |
| 29 | HQC-256 | 7,285 |
| 30 | mceliece460896 | 13,608 |
| 31 | mceliece6688128 | 13,932 |
| 32 | mceliece6960119 | 13,948 |
| 33 | mceliece8192128 | 14,120 |
| **34** | **PALOMA-128** | **94,496** |
| **35** | **PALOMA-192** | **355,400** |
| **36** | **PALOMA-256** | **357,064** |

선정된 Kyber를 기준으로 비교해보면 PALOMA, REDOG[20]을 제외하고 대부분의 알고리즘이 Kyber보다 작거나 이에 준하는 비밀키 크기를 가지고 있음을 알 수 있다. 따라서 대부분의 KpqC 알고리즘은 표준으로 선정되기 적합한 비밀키를 가지고 있다 할 수 있다.

[표 7]은 공개키 암호화 알고리즘들이 생성하는 암호문의 크기를 나열하였다. 암호문 크기의 경우, McEliece 암호가 가장 작은 암호문을 가지고 있었다. KpqC 후보군으로만 비교한다면 PALOMA 알고리즘이 가장 작은 크기를 가지고 있었다. 특히 PALOMA는 공개키, 비밀키 크기가 가장 큰 편에 속했는데 암호

〔표 7〕 PKE/KEM의 암호문 크기. 볼드체는 KpqC 후보군.

| Rank | PKE/KEM scheme | Ciphertext size (Byte) |
|---|---|---|
| 1 | mceliece348864 | 96 |
| **2** | **PALOMA-128** | **136** |
| 3 | mceliece460896 | 156 |
| 4 | mceliece6688128 | 194 |
| 5 | mceliece6960119 | 208 |
| | mceliece8192128 | 208 |
| **7** | **PALOMA-192** | **240** |
| | **PALOMA-256** | **240** |
| **9** | **Layered ROLLO-128** | **620** |
| **10** | **TiGER-128** | **768** |
| | **SMAUG-128** | **768** |
| | KYBER-512 | 768 |
| **13** | **REDOG-1** | **830** |
| **14** | **Layered ROLLO-192** | **850** |
| **15** | **NTRU+-576** | **864** |
| **16** | **TiGER-192** | **1,024** |
| | **SMAUG-192** | **1,024** |
| 18 | KYBER-768 | 1,088 |
| **19** | **TiGER-256** | **1,152** |
| | **NTRU+-768** | **1,152** |
| **21** | **Layered ROLLO-256** | **1,286** |
| **22** | **NTRU+-864** | **1,296** |
| **23** | **REDOG-2** | **1,440** |
| **24** | **SMAUG-256** | **1,536** |
| 25 | KYBER-1024 | 1,568 |
| **26** | **NTRU+-1152** | **1,728** |
| **27** | **REDOG-3** | **2,230** |
| 28 | HQC-128 | 4,497 |
| 29 | HQC-192 | 9,042 |
| 30 | BIKE-I | 12,579 |
| 31 | HQC-256 | 14,485 |
| 32 | BIKE-III | 24,915 |
| 33 | BIKE-V | 41,229 |
| **34** | **IPCC-f1** | **92,000** |
| | **IPCC-f3** | **92,000** |
| | **IPCC-f4** | **92,000** |

문 크기는 가장 작은 편에 속한다. NTRU+[21]의 경우, 공개키, 비밀키, 암호문 크기가 모두 중간 정도에 속했다. Kyber와 비교한다면 IPCC를 제외한 대부분의 암호 알고리즘이 합리적인 암호문 크기를 지니고 있다고 평가된다.

NIST PQC 공개키 암호화 알고리즘을 실제 사물인터넷 기기에 업로드하여 테스트한 시도가 존재한다[22]. 해당 결과에서는 TLS에 PQC 암호 알고리즘을 업로드하고 4개의 플랫폼을 준비해서 Kyber를 가동하였다. 각 플랫폼은 Raspberry Pi3 Model B+(RPi3), ESP32-PICO-KIT V4(ESP32), Fieldbus Option Card(FOC), 그리고 LPC11U68 LPCXpresso(LPC)이다. 성능 비교 대상으로는 TLS에 내장된 ECDHE(secp256r1)를 비교 대상으로 두었다. 실험 결과는 [표 8]과 같다.

전체적으로 Kyber의 성능이 ECDHE보다 뛰어난 것을 알 수 있다. 특히 플랫폼의 성능이 부족할수록 그 격차는 더욱 커진다. KpqC 알고리즘들이 Kyber와 같은 플랫폼 상에서 테스트 된 적이 없기 때문에 정확한 비교는 어려울 수 있으나, 전체적으로 훨씬 작은 매개변수를 가지고 있기 때문에 사물인터넷 기기와 같은 극한의 환경에서 가동이 유리할 것으로 판단된다. 또한 Kyber와 같은 격자 기반을 사용하는 알고리즘은 그 성능도 비슷하게 도출될 것으로 보인다.

〔표 8〕 임베디드 장비 상에서의 Kyber와 ECC (SECP256R1) 비교.

| Algorithm | RPi3 | ESP32 | FOC | LPC |
|---|---|---|---|---|
| Key Generation (Unit: ms) | | | | |
| **Kyber** | **0.79** | **12** | **51** | **220** |
| ECDHE | 14 | 290 | 1,400 | 2,900 |
| Encryption (Unit: ms) | | | | |
| **Kyber** | **1.1** | **16** | **73** | **300** |
| ECDHE | 12 | 250 | 2,800 | 990 |
| Decryption (Unit: ms) | | | | |
| **Kyber** | **1.1** | **18** | **83** | **300** |
| ECDHE | 14 | 290 | 1,400 | 3,000 |

### 3.2. 전자서명

NIST 표준으로 선정된 전자서명 알고리즘은 Dilithium, Falcon, SPHINCS+로 세 가지가 있다. 공개키 암호화 알고리즘을 비교한 것과 동일하게 공개키, 비밀키, 서명 크기를 비교한다. 성능 측정 부분은 앞에서 설명한 바와 마찬가지로 각 알고리즘마다 측정 환경이 다르므로 객관적 비교가 어려워 수행하지 않는다. [표 9]는 전자서명 알고리즘들의 공개키 크기를 나열한 것이다. 전자서명 알고리즘 중에서는 FIBS[23],

〔표 9〕 전자서명의 공개키 크기. 볼드체는 KpqC 후보군.

| Rank | Signature scheme | PK size (Byte) |
|---|---|---|
| 1 | **FIBS** | **32** |
| | **AIMer-I** | **32** |
| | SPHINCS+128 | 32 |
| 4 | **AIMer-III** | **48** |
| | SPHINCS+192 | 48 |
| 6 | SPHINCS+256 | 49 |
| **7** | **AIMer-V** | **64** |
| **8** | **SOLMAE-512** | **896** |
| 9 | FALCON-512 | 897 |
| | **Peregrine-512** | **897** |
| **11** | **HAETAE-II** | **1,056** |
| 12 | Dilithium-II | 1,312 |
| **13** | **NCCSign(original)-I** | **1,564** |
| **14** | **HAETAE-III** | **1,568** |
| **15** | **GCKSign-II** | **1,760** |
| **16** | **SOLMAE-1024** | **1,792** |
| 17 | **Peregrine-1024** | **1,793** |
| | FALCON-1024 | 1,793 |
| 19 | Dilithium-III | 1,952 |
| | **GCKSign-III** | **1,952** |
| **21** | **NCCSign(conserparam)-I** | **1,984** |
| **22** | **NCCSign(original)-III** | **1,997** |
| **23** | **HAETAE-V** | **2,080** |
| **24** | **NCCSign(conserparam)-III** | **2,443** |
| 25 | Dilithium-V | 2,592 |
| **26** | **NCCSign(original)-V** | **2,663** |
| **27** | **GCKSign-V** | **3,040** |
| **28** | **NCCSign(conserparam)-V** | **3,091** |
| **29** | **MQSign-72-46** | **328,411** |
| **30** | **Enhanced pqsigRM-613** | **474,445** |
| **31** | **MQSign-112-72** | **1,238,761** |
| **32** | **Enhanced pqsigRM-612** | **2,000,000** |
| **33** | **MQSign-148-96** | **2,892,961** |

〔표 10〕 전자서명의 비밀키 크기. 볼드체는 KpqC 후보군.

| Rank | Signature scheme | SK size (Byte) |
|---|---|---|
| **1** | **AIMer-I** | **16** |
| **2** | **AIMer-III** | **24** |
| **3** | **AIMer-V** | **32** |
| **4** | **FIBS** | **64** |
| | SPHINCS+128 | 64 |
| 6 | SPHINCS+192 | 96 |
| 7 | SPHINCS+256 | 128 |
| **8** | **GCKSign-III** | **288** |
| | **GCKSign-II** | **288** |
| **10** | **GCKSign-V** | **544** |
| **11** | **Peregrine-512** | **1,281** |
| | FALCON-512 | 1,281 |
| **13** | **NCCSign(original)-I** | **2,266** |
| **14** | FALCON-1024 | 2,305 |
| | **Peregrine-1024** | **2,305** |
| 16 | Dilithium-II | 2,528 |
| **17** | **NCCSign(conserparam)-I** | **2,800** |
| **18** | **NCCSign(original)-III** | **3,312** |
| **19** | **NCCSign(conserparam)-III** | **3,914** |
| 20 | Dilithium-III | 4,000 |
| **21** | **NCCSign(original)-V** | **4,402** |
| 22 | Dilithium-V | 4,864 |
| **23** | **NCCSign(conserparam)-V** | **4,940** |
| **24** | **Enhanced pqsigRM-613** | **10,736** |
| **25** | **MQSign-72-46** | **15,561** |
| **26** | **Enhanced pqsigRM-612** | **22,512** |
| **27** | **MQSign-112-72** | **37,729** |
| **28** | **MQSign-148-96** | **66,421** |

AIMer[24], 그리고 SPHINCS+가 가장 작은 공개키 크기를 지니고 있다.

전자서명 알고리즘은 공개키 암호화 알고리즘과 다르게, 같은 크기를 가지는 키가 여러 종류가 있는 것을 확인할 수 있다. 이는 전자서명 알고리즘에서 효율적인 매개변수 설정이 정형화 되어있다고 판단할 수 있다. NCCSign[25]의 경우에는 두 가지 버전이 존재하며, 매개변수가 다르다. Dilithium은 NIST 표준으로 선정된 알고리즘으로, 키 크기가 크다는 평가를 받는다. 이러한 점에서 KpqC 후보군들을 평가해보면 MQSign[26], Enhanced pqsigRM[27]을 제외한 나머지 모든 KpqC 후보 알고리즘들은 합리적인 공개키 크기를 가지고 있다.

[표 10]은 전자서명 알고리즘들의 비밀키 값을 나열하였다. HAETAE[28]와 SOLMAE[29]의 경우에는 제공되는 문서에서 비밀키가 명확하게 명시되지 않았기에 표에 표기하지 않는다. 가장 작은 비밀키를 가지는 알고리즘은 AIMer이며 FIBS가 그 뒤를 따라오고 있다. 마찬가지로 Dilithium을 기준으로 KpqC 후보 알고리즘을 평가한다면, MQSign과 Enhanced pqsigRM을 제외한 모든 알고리즘이 합리적인 비밀키 크기를 가지도록 설계되었다 할 수 있다. 마지막으로 [표 11]은 전자서명 알고리즘의 서명 크기를 나열한 것이다. MQSign은 가장 큰 공개키와 비밀키를 가지고 있었지만, 서명 크기는 가장 작았다. Enhanced pqsigRM도 마찬가지로 키 크기는 가장 큰 편이지만, 서명은 가장 작은 유형에 속했다. Peregrine[7]은 모든 경우에서 중간 이상을 기록하고 있다. AIMer는 공개키와 비밀키 크기가 가장 작았지만, 서명 크기는 KpqC 후보 알고리즘 중에서 가장 크다.

NIST PQC 공개키 암호화 알고리즘에 대한 테스트를 수행한 [22]에서는 전자서명 알고리즘도 TLS 상에 구현하여 사물인터넷 기기에서 가동하여 연산 성능을 도출하였다. 결과는 [표 12]와 같다. 실험 환경은 공개키 암호화 알고리즘과 동일한 환경이다. 전자서명 표준은 3종류가 있지만, 여기서는 SPHINCS+를 사용하였고, 그 중에서도 SHA, SHAKE 기반의 버전을 사용하였다. Haraka를 사용한 SPHINCS+는 실험에서 제외되었다. 비교 상대 알고리즘은 TLS에 내장된 EDCSA(SECP256R1)이다. 비교 결과 ECDSA가 SPHINCS+에 비해 좋은 성능을 지니는 것을 알 수 있다. 특히 가용 자원이 적은 FOC, LPC 플랫폼에서는 그 차이가 훨씬 커진다. 서명 생성의 경우에는 키 생성과 검증 과정에 비해 훨씬 더 느린 것을 알 수 있다. 이는 서명 생성 과정에서 해시 호출 횟수가 더 많기 때문에 발생하는 오버헤드이다.

KpqC 후보 알고리즘 중에서는 AIMer가 대칭키 기반으로 설계되어 해당 경향성을 따라갈 것으로 생각된다. 하지만 다른 알고리즘 역시 비슷한 흐름을 보일 수 있다. 따라서 공개키 암호화 알고리즘에 비해서 전자서명 알고리즘은 동작 효율을 더 고려해봐야 한다. 다만 KpqC 후보 알고리즘들은 NIST 표준 알고리즘에 비해 매개변수가 더 작은 편이므로, 연산 성능 역시 더

나을 것으로 기대할 수 있다.

〔표 11〕 전자서명의 서명 크기. 볼드체는 KpqC 후보군.

| Rank | Signature scheme | Sign. size (Byte) |
|---|---|---|
| **1** | **MQSign-72-46** | **134** |
| **2** | **MQSign-112-72** | **200** |
| **3** | **MQSign-148-96** | **260** |
| **4** | **Enhanced pqsigRM-613** | **512** |
| **5** | FALCON-512 | 666 |
| | **Peregrine-512** | **666** |
| | **SOLMAE-512** | **666** |
| **8** | **Enhanced pqsigRM-612** | **1,024** |
| **9** | **Peregrine-1024** | **1,280** |
| | FALCON-1024 | 1,280 |
| **11** | **SOLMAE-1024** | **1,375** |
| **12** | **GCKSign-II** | **1,952** |
| **13** | **GCKSign-III** | **2,080** |
| 14 | Dilithium-II | 2,420 |
| **15** | **NCCSign(original)-I** | **2,458** |
| **16** | **HAETAE-II** | **3,040** |
| **17** | **GCKSign-V** | **3,104** |
| **18** | **NCCSign(conserparam)-I** | **3,186** |
| 19 | Dilithium-III | 3,293 |
| **20** | **NCCSign(original)-III** | **3,605** |
| **21** | **HAETAE-III** | **4,064** |
| **22** | **NCCSign(conserparam)-III** | **4,251** |
| 23 | Dilithium-V | 4,595 |
| **24** | **NCCSign(original)-V** | **5,055** |
| **25** | **NCCSign(conserparam)-V** | **5,385** |
| **26** | **HAETAE-V** | **5,792** |
| **27** | **AIMer-I** | **5,904** |
| 28 | SPHINCS+128s | 7,856 |
| **29** | **AIMer-III** | **13,080** |
| 30 | SPHINCS+192s | 16,224 |
| **31** | **FIBS** | **17,088** |
| 32 | SPHINCS+128f | 17,088 |
| **33** | **AIMer-V** | **25,152** |
| 34 | SPHINCS+256s | 29,792 |
| 35 | SPHINCS+192f | 35,664 |
| 36 | SPHINCS+256f | 49,856 |

〔표 12〕 임베디드 시스템 상에서의 SPHINCS+와 ECC (SECP256R1) 비교.

| Algorithm | RPi3 | ESP32 | FOC | LPC |
|---|---|---|---|---|
| Key Generation (Unit: ms) | | | | |
| **SPI+SHA-256** | **26** | **710** | **6,000** | **12,000** |
| **SPI+SHAKE-256** | **200** | **2,100** | **17,000** | **45,000** |
| ECDSA | 28 | 260 | 3,700 | 2,800 |
| Sign (Unit: ms) | | | | |
| **SPI+SHA-256** | **840** | **22,000** | **51,000** | **380,000** |
| **SPI+SHAKE-256** | **5,100** | **6,400** | **200,000** | **1,300,000** |
| ECDSA | 15 | 290 | 1,500 | 1,100 |
| Verification (Unit: ms) | | | | |
| **SPI+SHA-256** | **66** | **950** | **2,200** | **16,000** |
| **SPI+SHAKE-256** | **240** | **2,800** | **8,900** | **60,000** |
| ECDSA | 25 | 580 | 2,900 | 4,100 |

## Ⅳ. 결 론

본 논문에서는 KpqC 후보 알고리즘들에 대한 효율성 관점에서 분석을 진행하였다. 이는 NIST 표준 선정 알고리즘 및 Round 4에 진출한 알고리즘과의 비교를 통해 이루어졌다. 대부분의 KpqC 알고리즘들은 NIST 양자내성암호 공모전의 알고리즘보다 더 작은 매개변수를 갖고 있었다. 이는 효율적인 키 관리, 암호문/서명 전송에 유리함을 의미한다. 연산 성능을 직접적으로 비교할 수는 없었지만, NIST 양자내성암호 알고리즘의 활용도를 살펴보고 경향성을 확인하였다. 그 결과 KpqC 알고리즘은 NIST 양자내성암호 알고리즘에 비해서 유리한 매개변수를 가지고 있었기에 실제 활용 시에도 비슷한 결과 또는 더 좋은 결과를 도출할 것으로 기대된다. 하지만 실제 실험으로는 이어지지 않았기에, 정밀하게 통제된 환경에서 알고리즘을 가동하고 그 결과를 정리할 필요가 있다. 따라서 본 연구진에서는 실제 연산 성능에 대한 객관적 평가 결과를 도출하여 효율성 관점에서 더 자세히 확인해 볼 계획이다.

## 참 고 문 헌


[1] L. K. Grover, “A fast quantum mechanical algorithm for database search,” *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212-219, 1996.

[2] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” *Proceedings 35th annual symposium on foundations of computer science*, pp. 124-134, Ieee, 1994.

[3] 양자내성암호연구단, [Online] https://kpqc.or.kr/

[4] 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.

[5] L. Ducas, E. Kiltz, T. Lepoint, V Lyubashevsky, 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] V. Lyubashevsky, “Fiat-Shamir with aborts: Applications to lattice and factoringbased signatures,” In *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 598-616, Dec. 2009.

[11] 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.

[12] N. Aragon, P. S. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J. C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J. P. Tillich, V. Vasseur, and G. Zémor, “BIKE: bit flipping key encapsulation,” 2017.

[13] D. J. Bernstein, T. Chou, T. Lange, I. von Maurich, R. Misoczki, R. Niederhagen, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, and W. Wang, “Classic McEliece: conservative code-based cryptography,” *NIST submissions*, 2017.

[14] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J. C. Deneuville, P. Gaborit, E. Persichetti, G. Zémor, and I.C. Bourges, “Hamming Quasi-Cyclic (HQC),” *NIST PQC Round*, 2(4), 2018.

[15] D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B Hess, A. Jalali, B Koziel, B, LaMacchia, P. Longa, M. Naehrig, J. Renes, and V. Soukharev, “Supersingular Isogeny Key Encapsulation,” *Submission to the NIST Post-Quantum Standardization Project*, 2017.

[16] SIKE Team, “SIKE and SIDH are insecure and should not be used,” 2022.

[17] 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.

[18] D. C. Kim, C. Y. Jeon, Y. Kim, and M. Kim, “PALOMA: Binary Separable Goppa-based

KEM," 2022.

[19] C. Kim, Y. S. Kim, and J. S. No, "Layered ROLLO-I: Faster rank-metric code-based KEM using ideal LRPC codes," *Cryptology ePrint Archive*, 2022.

[20] J. L. Kim, J. Hong, T. S. C. Lau, Y. Lim, C. H. Tan, T. F. Prabowo, B. S. Won, "REDOG and Its Performance Analysis," *Cryptology ePrint archive*, 2022.

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

[22] K. Bürstinghaus-Steinbach, C. Krauß, R. Niederhagen, and M. Schneider, "Post-Quantum TLS on Embedded Systems: Integrating and Evaluating Kyber and SPHINCS+ with mbed TLS," In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pp. 841-852, 2020.

[23] S. Kim, Y. Lee, and K. Yoon, "FIBS : Fast Isogeny Based Digital Signature," 2022.

[24] S. Kim, J. Ha, M. Son, B. Lee, D. Moon, J. Lee, S. Lee, J. Kwon, J. Cho, H. Yoon, and J. Lee, "The AIMer Signature Scheme," 2022.

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

[26] 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.

[27] J. Cho, J. S. No, Y. Lee, Z. Koo, and Y. S. Kim, "Enhanced pqsigRM: Code-Based Digital Signature Scheme with Short Signature and Fast Verification for Post-Quantum Cryptography," *Cryptology ePrint Archive*, 2022.

[28] 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.

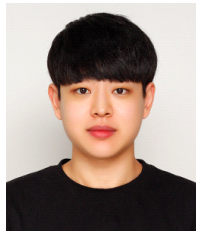
[29] SOLMAE Algorithm Specifications, [Online]: https://kpqc.or.kr/images/pdf/SOLMAE.pdf

## 〈저자소개〉

**권혁동 (Hyeok-Dong Kwon)**
학생회원
2018년 2월 : 한성대학교 IT융합공학부 졸업
2020년 2월 : 한성대학교 IT융합공학부 석사
2020년 3월~현재 : 한성대학교 정보컴퓨터공학과 박사과정
<관심분야> 정보보안, 암호구현

**심민주 (Min-Joo Sim)**
2021년 2월 : 한성대학교 IT융합공학부 졸업
2023년 2월 : 한성대학교 IT융합공학부 석사
2023년 3월~현재 : 한성대학교 정보컴퓨터공학과 박사과정
<관심분야> 정보보안, 암호구현

**송경주 (Gyeong-Ju Song)**
학생회원
2021년 2월 : 한성대학교 IT융합공학부 졸업
2023년 2월 : 한성대학교 IT융합공학부 석사
2023년 3월~현재 : 한성대학교 정보컴퓨터공학과 박사과정
<관심분야> 양자컴퓨팅, 암호구현, 정보보안



**이민우 (Min-Woo Lee)**
학생회원
2023년 2월 : 한성대학교 IT융합공학부 졸업
2023년 3월~현재 : 한성대학교 IT융합공학부 석사과정
<관심분야> 정보보안, 암호구현

**서 화 정 (Hwa-Jeong Seo)**

종신회원

2010년 2월 : 부산대학교 컴퓨터공학과 졸업

2012년 2월 : 부산대학교 컴퓨터공학과 석사

2016년 2월 : 부산대학교 컴퓨터공학과 박사

2017년 4월~2023년 2월 : 한성대학교 IT융합공학부 조교수

2023년 3월~현재 : 한성대학교 융합보안학과 부교수

<관심분야> 정보보안, 암호구현