정 보 보 호 학 회 지
제 30 권 제1호, 2020. 2

저성능 사물인터넷 상에서의 양자 내성 암호 구현

서 화 정*

요 약

기존 공개키 암호의 보안성을 저하시킬 수 있는 양자 컴퓨터와 양자 알고리즘의 급속한 발전으로 인해 미국의 NIST에서는 양자 내성 암호 표준화를 수행하고 있다. 전 세계에서 다양한 양자 내성 암호 표준안을 제시하였으며 현재 2 단계로 넘어가 차세대 양자 내성 암호를 다방면에서 평가하고 있다. 양자 내성 암호를 평가하는 다양한 항목 중에는 암호 보안 강도와 암호 구현 효율성이 있다. 본 기고에서는 자원적인 제한사항이 많은 저성능 사물인터넷 상에서의 양자 내성 암호 구현에 대해 확인해 보도록 한다.

Ⅰ. NIST 양자 내성 암호 표준화

미국의 NIST에서는 양자 컴퓨터 상에서 양자 알고리즘 활용한 공격에도 안전한 양자 내성 암호 표준화 작업을 2017년부터 진행해 오고 있다 [1]. 2017년 11월에는 양자 내성 암호 알고리즘 제안서를 전 세계 암호학자들로부터 제출받았으며 그 중에서 2018년 4월에는 보안상 문제가 없거나 제출자가 유보시키지 않은 양자 내성 암호 알고리즘을 공식적으로 공지하였다.

NIST에서 진행 중인 양자 내성 암호 공모전에 제안된 양자 내성 암호 알고리즘이 수행 가능한 연산으로는 전자서명 그리고 키 교환을 포함한다. 보안 등급은 AES-128 (level 1), AES-192 (level 3), AES-256 (level 5), SHA-256 (level 2), 혹은 SHA-384 (level 4)와 동일한 등급을 만족하기를 권장하고 있다. 초기에 NIST 공모전의 전체 일정은 총 10년으로 계획되었다. 이중 전반기 5년의 경우에는 표준 권고 알고리즘을 선정하는 작업을 하게 되며 후반기 5년 동안에는 선정된 알고리즘 구현 검증 및 산업계 확산적용을 수행하기로 계획하였다.

하지만 양자 컴퓨터의 발전이 유수의 IT 기업들 (IBM 그리고 Google)의 공격적인 투자와 함께 기존 예상과는 달리 매우 빠르게 진행되고 있다. 이는 대표적인 공개키 암호 시스템인 RSA와 ECC의 붕괴를 더 빠르게 가지고 올 수 있다는 불안감을 야기 시키고 있다. 이와 같은 사회적인 변화에 NIST도 빠르게 반응하여 애초 계획과는 달리 해당 공모전의 일정을 앞당기고 있다. 예를 들어 1차 선정 결과 발표가 초기에는 2019년도 8월로 계획되어 있었지만 실제로는 2019년도 1월에 기습 발표를 함으로써 7개월의 시간을 단축하였다. 이와 더불어 3차 그리고 최종 후보 선정 및 표준 문서 초안 작성도 일정이 앞당겨지고 있다. 이로인해 10년 계획은 빠르면 5년 늦어도 7년으로 단축되어 진행될 것으로 예상되고 있다. [표 1]에서는 현재 NIST 양자 내성 암호 표준화 로드맵을 나타내고 있다.

〔표 1〕 NIST 양자 내성 암호 표준화 로드맵

일정	내용
2017. 11	양자 내성 암호 알고리즘 제안 마감
2018. 04	제안 양자 내성 암호 알고리즘 소개
2018~2019	1차 평가분석 진행 (1차 후보 선정)
2019. 01	1차 선정 결과 발표
2019~2020	2차 평가분석 진행 (2차 후보 선정)
2020~	2차 선정결과 발표, 3차/최종 후보 선정 및 표준 문서 초안 작성

NIST 양자 내성 암호 2단계 표준안 후보로 채택된 알고리즘의 수는 총 26개이다. 이 중에서 17개는 키 교환 알고리즘 그리고 9개는 전자 서명 알고리즘에 각각

본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.
* 한성대학교 IT융합공학부(교수, hwajeong84@gmail.com)

포함된다. 키 교환 알고리즘에 채택된 알고리즘으로는 BIKE, Classic McEliece, CRYSTALS-KYBER, FrodoKEM, HQC, LAC, LEDAcrypt, NewHope, NTRU, NTRU Prime, NTS-KEM, ROLLO, Round5, RQC, SABER, SIKE, 그리고 Three Bears이다. 전자서명 알고리즘에 채택된 알고리즘은 CRYSTALS-DILITHIUM, FALCON, GeMSS, LUOV, MQDSS, Picnic, qTESLA, Rainbow, 그리고 SPHINCS+이다. 이와 더불어 산업체에서는 GPDR (General Data Protection Regulation)과 같은 개인정보 보호 규정을 만족시키기 위해 지금 당장 활용 가능한 양자 내성 암호를 NIST에 요구하였다. 이에 대한 대응으로 NIST에서는 XMSS와 같이 오랜 시간동안 검증되어 온 stateful 해시 기반 양자 내성 암호를 우선적으로 사용하길 공고하고 있다. NIST에서는 주기적으로 제안된 양자 내성 암호에 대한 이해를 돕고 새로운 암호 취약점을 공유하기 위해 워크샵을 개최하고 있으며 2019년에는 8월에 Crypto 와 동일한 장소인 산타바바라에서 그 결과물을 발표하였다. 해당 워크샵을 통해 현재 연구 중인 다양한 양자 내성 암호를 상호 평가하였으며 2단계 평가를 통과하게 될 양자 내성 암호의 윤곽이 어디 정도 표면 상에 나타나는 계기가 되었다.

현재 NIST 양자 내성 암호 표준화에서 2단계로 진행중인 양자 내성 암호 후보군들을 살펴보면 [표 2]와 같다. 다양한 양자 내성 암호들은 다양한 암호학적 문제(격자, 코드, 다항식, 해시, 그리고 아이소지니)에 기반하고 있으며 이는 NIST에서 다양한 양자 내성 암호에 모두 관심을 가지고 진행 중에 있음을 확인할 수 있다.

〔표 2〕 다양한 양자 내성 암호 장점 및 단점 비교

종류	알고리즘	장점	단점
격자 기반 양자 내성 암호	NTRU, LAC	빠른 연산 속도	파라미터 설정 어려움
다변 수 기반 양자 내성 암호	Rainbow, LUOV	작은 서명 크기와 빠른 연산 속도	큰 키 사이즈
해시 기반 양자 내성 암호	XMSS, SPHINCS+	안전성 증명 가능	큰 서명 사이즈
Isogeny 기반 양자 내성 암호	SIKE	작은 키 사이즈	느린 연산속도
코드 기반 양자 내성 암호	Classic McEliece, BIKE	빠른 암호화 및 복호화 속도	큰 키 사이즈

지금까지 NIST에서는 블록암호 (AES) 그리고 해시함수 (SHA-3)에 대한 국제적인 표준안을 공개적으로 진행해 온 이력이 있다. 지금까지 블록암호와 해시함수 표준화의 경우 보안성과 효율성을 명확히 비교 분석할 수 있는 토대가 마련되어 있었다. 하지만 양자 내성 암호의 경우 동일한 잣대로 다양한 암호학적 문제에 기반한 표준안들을 비교분석하는 것은 불가능하다. 양자 내성 암호들은 블록암호와 해시함수와는 달리 절대적으로 모든 방면에서 우수성을 제공하고 있지 못하다. 효율성을 평가하는 기준인 연산속도, 키 크기, 서명 크기, 그리고 메시지 크기는 모든 양자 내성 암호가 상이하다. 따라서 사용자의 서비스 적용 분야에 따라 선호되는 암호화 기법이 달라질 것으로 예상된다. 현재 제안되고 있는 양자 내성 암호 군들과 해당 알고리즘들의 장점 그리고 단점을 나타내면 [표 2]와 같다.

이와 더불어 고려해야 할 사항으로는 양자 내성 암호가 실제로 양자 내성의 특성을 가지는 지에 대한 확증이다. 현재 제안되는 양자 내성 암호들의 경우 실제 양자 컴퓨터 상에서 보안 취약점이 있는지 실증을 해볼 수 없는 상황이다. 이와 더불어 쇼어 알고리즘에는 내성을 가지지만 이후에 발견되는 다양한 공격 기법에 의해 양자 내성 암호의 보안성이 훼손될 수도 있는 실정이다. 현재는 그루버 알고리즘을 이용하여 기존에 양자 내성 암호 공격에 활용되던 알고리즘의 복잡도를 줄이는 형식으로 연구들이 진행되고 있다.

Ⅱ. 양자 내성 암호를 위한 저성능 사물인터넷 플랫폼 및 벤치마크 프레임워크

양자 내성 암호의 보안취약점에 대한 관심은 NIST 양자 내성 암호 표준화 1단계에서 가장 중점적으로 살펴보았던 내용이다. 그 이유는 제대로된 양자 내성 암호 설계가 선행되어야 그 이후에 따라오는 암호 구현 그리고 부채널 공격 방어가 가능하기 때문이다. 실제로 양자 내성 암호에서 사용되는 파라미터의 값에 따라 이를 컴퓨터 상에서 구현하는 암호 구현 기법 전체가 변경될 수 있다. 하지만 이와 더불어 어떤 파라미터가 실제 구현 시에 높은 성능을 가지는지를 확인하는 용도로 구현과 파라미터 설정을 병행하는 방법론도 같이 고려되고

있다.

암호 구현은 사용 목적에 따라 소프트웨어 혹은 하드웨어의 형태로 제작 가능하다. 소프트웨어는 쉽게 배포가 가능하다는 특징이 있지만 단독적으로 높은 보안성을 확보하기에는 어려움이 있다. 하드웨어의 경우 높은 성능과 더불어 보안성을 확보할 수 있다. 다만 내부 로직 혹은 알고리즘의 문제점이 발견된 경우 이를 수정하는 것이 어렵다는 특징을 가진다.

현재 NIST에서는 소프트웨어 그리고 하드웨어 두 환경 모두에 대한 성능 평가를 요구하고 있다. 특히 하드웨어의 경우 Artix-7 FPGA 보드 그리고 소프트웨어의 경우 ARM Cortex-M4 보드를 주요 타겟으로 설정하였다. 본 고에서는 저성능 프로세서인 ARM Cortex-M4 상에서 양자 내성 암호를 구현한 결과물들에 대해 확인해 보도록 한다.

지금까지의 양자 내성 암호 구현의 경우 성능 상에서는 큰 어려움이 없는 데스크탑 레벨에서 수행되었다. 따라서 Intel 명령어 혹은 SIMD 명령어인 AVX2를 활용한 구현기법들이 주로 소개되었다. 하지만 대부분의 암호화 디바이스의 경우 데스크탑에 비해 떨어지는 성능을 가지고 있다. 해당 디바이스들은 제한적인 RAM, ROM 그리고 느린 연산 속도를 가진다. 따라서 이러한 저성능 환경을 대표하는 ARM Cortex-M4 보드 상에서의 구현이 성공적으로 된다면 데스크탑에서는 큰 무리가 없을 것으로 보인다. 대표적인 ARM Cortex-M4 개발 보드로는 STM32F4DISCOVERY 보드가 있다. 이는 32-비트 ARMv7E-M을 지원하며 192 KB RAM과 168 MHz의 연산속도를 가진다. 기존의 저성능 프로세서인 8-비트 AVR에 비해서는 월등한 성능을 가지지만 Intel 프로세서에 비해서는 매우 제한적인 성능을 지니고 있다. 현재 ARM Cortex-M4를 벤치마크해 볼 수 있는 플랫폼으로는 PQM4 프로젝트가 있다. 여기서는 자동화된 툴을 통해 양자 내성 암호의 연산 속도와 RAM 그리고 ROM 사용량을 확인해 볼 수 있도록 하고 있다 [2]. PQM4 프로젝트는 현재 4가지 형식의 구현 결과물을 포함하고 있다. 해당 구현물은 레퍼런스 코드부터 ARM Cortex-M4에 특화된 코드까지 다양한 종류를 포함하며 이에 대한 상세한 내용은 [표 3]과 같다.

PQM4에서는 객관적인 벤치마크 지표 도출을 위해 많은 외부적인 요인들을 통일시키는 작업을 진행 중에 있다. 먼저 cycle count이다. 임베디드 프로세서 상에서

〔표 3〕 PQM4의 구현 결과물 종류 상세

종류	설명
ref	NIST 표준 제안에 제출된 C 레퍼런스 코드
clean	제출된 코드를 변경하여 기본적인 코드 요구사항을 만족 시킨 코드
opt	portable C 구현을 최적화한 코드
m4	ARMv7E-M 어셈블리 언어를 활용하여 최적화한 코드

의 속도 측정 시 가장 정확한 성능 지표는 clock cycle이다. 이는 컴퓨터가 항상 일정한 주기로 clock을 생성하고 있다는 것에 기반을 두고 있다. 현재 ARM Cortex-M4 보드의 최대 주파수는 168MHz이다. 하지만 이는 암호구현이 아닌 메모리 컨트롤러의 딜레이에서 오는 연산 부하가 전체 clock cycle에 높은 영향을 미치게 되는 문제점을 가지고 있다. 따라서 PQM4 상에서는 정확한 벤치마크를 위해 24MHz 상에서 clock cycle을 측정하도록 하고 있다. 또한 해당 보드 상에서는 하드웨어 기반 난수 생성기를 지니고 있다. 난수 생성기는 암호의 핵심인 키를 안전하고 신속히 생성해 내는 역할을 수행하게 된다. 이와 더불어 양자 내성 암호 알고리즘들은 SHA-2, SHA-3 그리고 AES를 기반으로 하여 연산 설계가 되어 있다. 해당 암호 알고리즘은 양자 내성 암호에 꼭 필요한 부분이지만 해당 부분에서 오는 연산 속도가 전체 성능에 영향을 미치는 것은 객관적인 벤치마크를 불가능하게 하는 문제점을 가지고 있다. 따라서 PQM4 상에서는 SHA-2, SHA-3 그리고 AES에 대한 최적화된 단일 결과를 활용하도록 하고 있다. SHA-2의 경우에는 SUPERCOP 상의 C 기반 고속 구현을 활용하고 있다[1]. SHA-3의 경우 SHA-3 개발팀에서 개발한 XKCP 라이브러리 상의 ARMv7-M 어셈블리 구현을 활용하고 있다[2]. AES의 경우 SAC'16에서 제안된 ARMv7-M 상에서의 어셈블리 구현에 기반을 두고 있다 [3]. 해당 라이브러리에서는 AES-128, AES-192, 그리고 AES-256을 ARM Cortex-M3와 M4 상에서의 최적화된 구현 결과를 제공하고 있다.

현재 다양한 양자 내성암호가 PQM4 라이브러리 상에서 벤치마크되고 있으며 이에 대한 상세 비교는 [표 4]와 같다. 기본적으로 reference 코드는 NIST에서 제공하는 포맷에 맞추어 제출하였기 때문에 PQM4에 대

1) https://bench.cr.yp.to/supercop.html
2) https://github.com/XKCP/XKCP

〔표 4〕 PQM4에 포함된 양자내성암호 비교

양자내성암호	reference	optimized
CRYSTALS-Kyber	○	○
Frodo-KEM	○	○
LAC	○	-
NewHope	○	○
NTRU	○	○
NTRU Prime	○	-
Round5	○	○
SABER	○	○
SIKE	○	-
ThreeBears	○	○
CRYSTALS-Dilithium	○	○
LUOV	○	-
qTESLA	○	-
SPHINCS+	○	-

한 포팅이 손쉽게 이루어지고 있다. 이와 더불어 각 양자 암호 제안팀에서는 자신들의 암호의 효율성을 입증하기 위해 최적화된 양자내성암호 구현을 제공하고 있다.

하지만 모든 양자 내성 암호가 ARM Cortex-M4 상에서 구현 및 동작이 가능한 것은 아니다. 여기에는 2가지 이유를 생각해 볼 수 있다. 첫 번째는 큰 키 크기이다. 암호 수행 시 키와 데이터는 RAM에 필요한 영역만큼 할당된 이후에 수행되게 된다. 비록 ARM Cortex-M4가 192 KB RAM을 제공하고 있지만 양자 내성 암호는 매우 큰 키 크기로 인해 192 KB를 넘어서는 경우가 존재하게 된다. 이러한 경우 구현 및 운용이 불가능하다. 큰 키 크기로 PQM4 상에서의 벤치마크가 어려운 양자 내성 암호에는 Classic McEliece, LEDAcrypt, NTS-KEM, GeMSS, MQDSS, Picnic, 그리고 Rainbow가 있다.

두 번째로는 외부 라이브러리에 의존해서 레퍼런스 코드가 작성된 경우이다. 암호 구현 시 모든 암호화 연산을 프리미티브 연산부터 구현하는 방법과 더불어 기존 라이브러리를 호출하여 사용하는 방법도 함께 활용되고 있다. 현재 많은 레퍼런스 코드들이 OpenSSL, NTL 그리고 GMP 라이브러리에 대한 강한 의존성을 가지고 있다. 이로 인해 PQM4에 바로 적용하는 것이 힘들다. 그 이유는 해당 외부 라이브러리는 주로 데스크탑을 목적으로 설계되었으며 많은 경우에는 기계어를 이용하여 최적화되어 있다. 따라서 내부 기계어가 다르고 RAM의 크기가 상이한 저성능 프로세서 상에서는 제대로된 동작을 기대하기 어렵다. 외부 라이브러리에 대한 의존성으로 인해 PQM4 상에서의 벤치마크가 어려운 양자 내성 암호에는 BIKE, HQC, ROLLO, 그리고 RQC가 있다.

Ⅲ. 양자 내성 암호 구현

본 장에서는 ARM Cortex-M4 상에서 어셈블리 언어를 활용하여 최적화 구현한 결과물들에 대해서 종합하여 나타내도록 한다.

3.1. FrodoKEM

FrodoKEM은 Learning With Errors (LWE) 기반 문제에 기반하고 있다. 높은 보안성을 달성하기 위해서는 매우 큰 행렬 곱셈을 수행해야 한다. 해당 행렬 곱셈의 인자들은 Z_q $(q=2^{15})$ 상에서 정의되어 있다. 현재 Cortex-M4 상에서는 DSP 모듈을 이용하여 16-비트 인자를 동시에 2 개를 수행할 수 있는 SIMD 연산자를 제공하고 있다. 이를 이용하게 될 경우 메모리에 대한 접근과 연산 횟수를 절반이하로 줄일 수 있음을 나타낸다. 또한 smlad 연산자를 이용할 경우 두 개의 하프 워드를 곱한 이후에 특정한 값을 더하고 결과값을 저장하는 것이 가능하다. 만약 smlad d,a,b,c와 같이 연산을 수행하는 경우 이는 아래와 같은 연산을 수행함을 의미한다 [4].

$$d=(a \bmod 2^{16}) \cdot (b \bmod 2^{16}) + \lfloor \frac{a}{2^{16}} \rfloor \cdot \lfloor \frac{b}{2^{16}} \rfloor + c \bmod 2^{32}$$

그리고 FrodoKEM의 경우 cSHAKE128 혹은 AES128을 기본 연산으로 활용하고 있다. 하지만 기본적인 구현은 메모리 레이아웃을 고려하지 않아 연산 속도가 느려지는 단점이 있다. 해당 프로세서의 메모리 레이아웃을 고려한 구현을 제공함으로써 연산 성능을 효과적으로 높일 수 있다. 하지만 여전히 AES128와 cSHAKE128는 매우 높은 부하를 나타내게 된다. 이를

〔표 5〕 FrodoKEM640 연산 속도 비교 (연산 단위: 10^6 clock cycles)

PRNG	Function	PQM4 〔2〕	Howe et al. 〔5〕	Bos et al. 〔4〕
cSHAKE128	KeyGen	94	85	81
	Encaps	106	112	86
	Decaps	107	112	87
AES128	KeyGen	-	44	41
	Encaps	-	47	45
	Decaps	-	48	46
xoshiro128	KeyGen	-	-	14
	Encaps	-	-	14
	Decaps	-	-	15

해결하기 위해 xoshiro128과 같은 경량 암호를 활용하여 구현함으로써 연산 성능을 높였다. FrodoKEM640의 ARM Cortex-M4 상에서의 구현 결과는 [표 5]와 같다.

3.2. CRYSTALS-Kyber

Kyber는 CRYSTALS의 일부 알고리즘으로써 Module-LWE 문제에 기반하고 있다. Ring-LWE 기반 알고리즘과 달리 R_q 상에서 연산이 구현되게 된다. 여기서 $q=7681=2^{13}-2^9+1$ 이고 $R_q=Z_{7681}/(X^{256}+1)$ 을 모든 보안 레벨에서 사용하고 있다. 해당 구현에서는

〔표 6〕 Kyber 연산 속도 비교 (연산 단위: 10^3 clock cycles)

Scheme	Function	Avanzi et al.[3]	Botros et al. 〔6〕
Kyber-512 (v1)	KeyGen	666	575
	Encaps	904	763
	Decaps	934	730
Kyber-512 (v2)	KeyGen	-	499
	Encaps	-	634
	Decaps	-	597
Kyber-768 (v1)	KeyGen	1,098	946
	Encaps	1,384	1,167
	Decaps	1,417	1,117
Kyber-768 (v2)	KeyGen	-	947
	Encaps	-	1,113
	Decaps	-	1,059
Kyber-1024 (v1)	KeyGen	1,730	1,483
	Encaps	2,083	1,753
	Decaps	1,059	1,698
Kyber-1024 (v2)	KeyGen	-	1,525
	Encaps	-	1,732
	Decaps	-	1,653

〔표 7〕 Kyber 메모리 사용량 비교 (연산 단위: bytes)

Scheme	Function	Avanzi et al. [4]	Botros et al. 〔6〕
Kyber-512 (v1)	KeyGen	6,448	-
	Encaps	9,112	-
	Decaps	9,920	-
Kyber-512 (v2)	KeyGen	-	3,136
	Encaps	-	2,720
	Decaps	-	2,744
Kyber-768 (v1)	KeyGen	10,544	-
	Encaps	13,720	-
	Decaps	14,880	-
Kyber-768 (v2)	KeyGen	-	3,648
	Encaps	-	3,232
	Decaps	-	3,248
Kyber-1024 (v1)	KeyGen	15,664	3,520
	Encaps	19,352	3,568
	Decaps	20,864	3,624
Kyber-1024 (v2)	KeyGen	-	4,160
	Encaps	-	3,752
	Decaps	-	3,776

핵심연산인 NTT 연산을 효율적으로 계산하기 위해 signed Montgomery multiplication을 구현하였으며 이는 기존의 unsigned 구현에 비해 높은 성능을 나타냄을 확인하였다 [6]. NTT의 성능이 개선됨에 따라 Kyber의 성능 역시 개선됨을 확인할 수 있었다. 이와 더불어 기존 C 코드 상에서 메모리 할당을 무분별하게 해놓은 부분을 최적화함으로써 전체 소모되는 메모리의 양을 줄일 수 있었다. 연산 속도에 대한 비교는 [표 6]와 같으며 연산에 사용되는 메모리 사용량은 [표 7]과 같다.

3.3. CRYSTALS-Dilithium

Dilithium은 CRYSTALS 상의 양자 내성 서명 알고리즘으로써 보수적인 module lattice 기반으로 설계되어 있다. 따라서 파라미터의 크기가 크고 느린 속도를 가지고 있다. 속도 향상을 위해 해당 구현에서도 NTT

〔표 8〕 Dilithium III 연산 속도 비교 (연산 단위: 10^3 clock cycles)

Scheme	Function	Güneysu et al. 〔7〕
Dilithium III	KeyGen	2,320
	Encaps	8,348
	Decaps	2,342

3) https://github.com/pq-crystals/kyber/tree/cm4/cm4
4) https://github.com/pq-crystals/kyber/tree/cm4/cm4

〔표 9〕 Dilithium III 메모리 사용량 비교 (연산 단위: bytes)

Scheme	Function	Güneysu et al. [7]
Dilithium III	KeyGen	50,488
	Encaps	86,568
	Decaps	54,800

성능 향상을 위해 Montgomery reduction을 적용하였다. Dilithium III의 연산 속도와 메모리 사용량은 [표 8]과 [표 9]에 각각 나타나 있다 [7].

3.4. NTRU 그리고 Saber

NTRU-HRSS-KEM은 classic NTRU 시스템에 기반을 하고 있다. 파라미터 공간을 기존 NTRU에 비해 제한함으로써 스킴을 간략화하였다. NTRUEncrypt 역시 standard NTRU에 기반하고 있다. NTRU-HRSS-KEM이 FO transform을 사용했다면 NTRUEncrypt는 NAEP transform을 통해 시스템을 설계하였다. Saber의 경우 module-Learning With Rounding (LWR) 문제에 기반하고 있다. ACNS'19에서 발표된 논문에서는 NTRU와 Saber에서 자주 사용되는 polynomial 곱셈을 Karatsuba와 Toom-Cook 알고리즘을 이용하여 개선하였다 [8]. polynomial 차수에 따라 다양한 알고리즘들의 성능 비교를 함과 동시에 ARM 어셈블리 명령어인 smlad와 smladx를 이용하여 16-비트 단위의 곱셈을 한 번에 두 개씩 병렬 수행하였다. 세 알고리즘에 대한 연산속도와 메모리 사용량은 [표 10]과 [표 11]에 각각 비교되어 있다.

〔표 10〕 Saber, NTRU-HRSS, NTRU-KEM 743 연산 속도 비교 (연산 단위: 10^3 clock cycles)

Scheme	Function	Karmakar et al. [9]	Kannwischer et al. [8]
Saber ($n=256$, $q=2^{13}$)	KeyGen	1,147	895
	Encaps	1,444	1,161
	Decaps	1,543	1,204
NTRU-KEM-743 ($n=743$, $q=2^{11}$)	KeyGen	-	5,198
	Encaps	-	1,601
	Decaps	-	1,881
NTRU-HRSS ($n=701$, $q=2^{13}$)	KeyGen	-	145,963
	Encaps	-	404
	Decaps	-	819

〔표 11〕 Saber, NTRU-HRSS, NTRU-KEM 743 메모리 사용량 비교 (연산 단위: bytes)

Scheme	Function	Karmakar et al. [9]	Kannwischer et al. [8]
Saber ($n=256$, $q=2^{13}$)	KeyGen	13,883	13,248
	Encaps	16,667	15,528
	Decaps	17,763	16,624
NTRU-KEM-743 ($n=743$, $q=2^{11}$)	KeyGen	-	25,320
	Encaps	-	23,808
	Decaps	-	28,472
NTRU-HRSS ($n=701$, $q=2^{13}$)	KeyGen	-	23,396
	Encaps	-	19,492
	Decaps	-	22,140

3.5. Falcon

Falcon은 격자 기반 수학적 난제에 기반한 양자 내성 서명 스킴으로써 마이크로콘트롤러 상에서 구현에 적합하다. Falcon의 보안 증명은 QROM 상에서 되었으며 격자 기반 서명 스킴 중에서 가작 작은 공개키와 서명 크기를 가지고 있다. Falcon 상에서 많은 메모리를 소모하는 부분은 fast Fourier sampling에서 사용되는 tree를 구성하는 부분이다. 이를 효과적으로 대체하기 위해 tree를 항상 유지하는 대신 on-the-fly 방식으로 sampling을 취하는 방법론을 적용하였다. 만약 고정된 키를 사용하는 경우에는 tree를 구성하지 않아도 되기 때문에 연산 성능이 좋아지게 된다. Falcon의 연산 속도와 메모리 사용량은 [표 12]와 [표 13]에 각각 나타나

〔표 12〕 Falcon 연산 속도 비교 (연산 단위: 10^3 clock cycles)

Scheme	Function	Oder et al. [10]	Oder et al. (Fixed Keys) [10]
Falcon ($n=512$)	KeyGen	114,546	-
	Encaps	80,503	72,261
	Decaps	530	529
Falcon ($n=1024$)	KeyGen	365,950	-
	Encaps	165,800	147,330
	Decaps	1,046	1,083

〔표 13〕 Falcon 메모리 사용량 비교 (연산 단위: bytes)

Scheme	Function	Oder et al. [10]	Oder et al. (Fixed Keys) [10]
Falcon ($n=512$)	KeyGen	40,560	-
	Encaps	63,652	50,508
	Decaps	6,261	5,364
Falcon ($n=1024$)	KeyGen	51,704	-
	Encaps	120,596	94,260
	Decaps	11,893	10,100

있다.

3.6. ROLLO-I

ROLLO-I는 rank 기반 코드 (LRPC codes)로써 KEM을 제공하고 있다. ROLLO-I의 경우에는 기존에 외부 라이브러리 의존성으로 인해 PQM4 상에서 활용이 불가능하였다. 하지만 해당 논문에서는 이를 직접 C로 구현하여 동작시킨 결과를 나타내고 있다. 메모리 최적화를 위해서는 Schoolbook 기법을 활용하여 곱셈을 구현한다. 속도 최적화를 위해서는 Karatsuba 알고리즘을 활용하여 연산 복잡도를 낮추었다. 연산 속도와 메모리 사용량의 비교는 [표 14] 그리고 [표 15]에 각각 나타나 있다.

〔표 14〕 ROLLO-I 연산 속도 비교 (연산 단위: 10^6 clock cycles)

Scheme	Function	Lablanche et al. (memory) [11]	Lablanche et al. (speed) [11]
ROLLO-I-128	KeyGen	9.7	8.68
	Encaps	1.51	0.6
	Decaps	10.17	3.97
ROLLO-I-192	KeyGen	12.7	11.11
	Encaps	2.39	0.8
	Decaps	15.29	6.63

〔표 15〕 ROLLO-I 메모리 사용량 비교 (연산 단위: bytes)

Scheme	Function	Lablanche et al. (memory) [11]	Lablanche et al. (speed) [11]
ROLLO-I-128	KeyGen	2,640	3,148
	Encaps	1,928	3,376
	Decaps	2,168	3,660
ROLLO-I-192	KeyGen	2,972	3,520
	Encaps	2,156	3,508
	Decaps	2,748	5,076

3.7. SIKE

SIKE는 아이소지니 기반 키교환 프로토콜을 KEM으로 확장한 프로토콜로써 ECC의 프리미티브 연산에 기반을 두고 있는 양자 내성 암호이다. SIKE는 매우 작은 키와 암호문의 크기를 가지지만 연산 속도가 느린 단점을 가지고 있다. 따라서 ARM Cortex-M4 상에서 만족할 만한 성능을 도출하는 것이 매우 중요하다. CANS'19에서 발표된 연구 결과에서는 Montgomery multiplication을 최적화하고 연산자들의 파이프라이닝을 만족하는 형식으로 명령어셋을 조정함으로써 높은 연산 성능을 달성하는 것이 가능함을 보였다 [12]. 연산 속도와 메모리 사용량의 비교는 [표 16] 그리고 [표 17]

〔표 16〕 SIKE 연산 속도 비교 (연산 단위: 10^6 clock cycles)

Scheme	Function	PQM4 [2]	Seo et al. [12]
SIKEp434	KeyGen	-	74
	Encaps	-	122
	Decaps	-	130
SIKEp503	KeyGen	1,086	104
	Encaps	1,799	172
	Decaps	1,912	183
SIKEp751	KeyGen	3,651	282
	Encaps	5,918	455
	Decaps	6,359	491

〔표 17〕 SIKE 메모리 사용량 비교 (연산 단위: bytes)

Scheme	Function	PQM4 [2]	Seo et al. [12]
SIKEp434	KeyGen	-	6,580
	Encaps	-	6,916
	Decaps	-	7,260
SIKEp503	KeyGen	-	6,204
	Encaps	-	6,588
	Decaps	-	6,974
SIKEp751	KeyGen	-	11,116
	Encaps	-	11,260
	Decaps	-	11,852

에 각각 나타나 있다.

Ⅳ. 결 론

본 고에서는 양자 내성 암호를 저성능 사물인터넷 프로세서인 32-비트 ARM Cortex-M4 상에서 구현한 최신 결과물들을 종합하여 나타내고 있다. 현재 다양한 양자 내성 암호 알고리즘들이 해당 프로세서 상에서 구현되고 있다. 해당 결과들을 살펴보면 키 크기와 연산 속도가 구현에 있어 가장 어려운 문제로 작용하고 있음을 확인할 수 있다. 아직도 많은 양자 내성 암호 후보군들이 ARM Cortex-M4 상에서 구현이 되지 못한 상황이다. 여기에는 외부 라이브러리 의존성과 키 크기의 문제점이 있다. 이러한 문제점을 해결하기 위해서는 키 크기를 줄이기 위한 알고리즘 혹은 구현 아키텍처 관점에서의 접근이 필요할 것으로 보이며 외부 라이브러리를 제외한 구현도 필요하다. 또한 기본 C언어 구현에서는 최적화된 성능을 얻을 수 없기 때문에 이를 어셈블리 언어로 변환하고 Karatsuba와 같은 알고리즘을 적용하여 연산 성능을 실용적인 단계까지 향상 시켜 나가야 할 것으로 보인다.

참 고 문 헌

[1] NIST. "Post-Quantum Cryptography," (https://csrc.nist.gov/Projects/Post-Quantum-Cryptography).

[2] M. Kannwischer, J. Rijneveld, P. Schwabe, K. Stoffelen, "PQM4: Post-quantum crypto library for the ARM Cortex-M4," 2018.

[3] P. Schwabe, K. Stoffelen, "All the AES you need on Cortex-M3 and M4," In International Conference on Selected Areas in Cryptography, pp. 180-194, 2016.

[4] J. Bos, S. Friedberger, M. Martinoli, E. Oswald, M. Stam, "Fly, you fool! Faster Frodo for the ARM Cortex-M4," IACR Cryptology ePrint Archive, 2018.

[5] J. Howe, T. Oder, M. Krausz, T. Güneysu, "Standard lattice-based key encapsulation on embedded devices," IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 372-393, 2018.

[6] L. Botros, M. Kannwischer, P. Schwabe, "Memory-efficient high-speed implementation of Kyber on Cortex-M4," In International Conference on Cryptology in Africa pp. 209-228, 2019.

[7] T. Güneysu, M. Krausz, T. Oder, J. Speith, "Evaluation of lattice-based signature schemes in embedded systems," In 2018 25th IEEE International Conference on Electronics, Circuits and Systems, pp. 385-388, 2018.

[8] M. Kannwischer, J. Rijneveld, P. Schwabe, "Faster multiplication in Z2m [x] on Cortex-M4 to speed up NIST PQC candidates," Cryptology ePrint Archive, Report 2018/1018, 2018.

[9] A. Karmakar, I. Verbauwhede, "Saber on ARM. CCA-secure module lattice-based key encapsulation on ARM," IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 243-266, 2018.

[10] T. Oder, J. Speith, K. Höltgen, T. Güneysu, "Towards Practical Microcontroller Implementation of the Signature Scheme Falcon," In International Conference on Post Quantum Cryptography, pp. 1-17, 2019.

[11] J. Lablanche, L. Mortajine, O. Benchaalal, P. Cayrel, & N. El Mrabet, "Optimized implementation of the NIST PQC submission ROLLO on microcontroller," Cryptology ePrint Archive, Report 2019/787, 2019.

[12] H. Seo, A. Jalali, R. Azarderakhsh, "SIKE round 2 speed record on ARM Cortex-M4," In International Conference on Cryptology and Network Security, pp. 39-60, 2019.

〈저 자 소 개〉

서 화 정 (Hwa-jeong Seo)

종신회원

2010년 2월 : 부산대학교 컴퓨터공학과 학사 졸업
2012년 2월 : 부산대학교 컴퓨터공학과 석사 졸업
2016년 2월 : 부산대학교 컴퓨터공학과 박사 졸업
2015년 4월~5월 : 싱가포르 난양공대 인턴쉽
2016년 1월~2017년 3월 : 싱가포르 과학기술청 연구원
2017년 4월~현재 : 한성대학교 조교수
<관심분야> 암호 구현