Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.5, Oct. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.5.781

μ-Hope : 오류 정정 부호를 사용한 RLWE 기반의 경량 KEM*

이 주 엽,[1†] 김 수 리,[2] 김 창 한,[3] 홍 석 희[4‡]
[1,2,4]고려대학교 (대학원생, 박사후 연구원, 교수), [3]세명대학교 (교수)

μ-Hope : Compact Size RLWE Based KEM Using Error Correcting Code*

Juyeop Lee,[1†] Suhri Kim,[2] Chang Han Kim,[3] Seokhie Hong[4‡]
[1,2,4]Korea University (Graduate student, Post doctor, Professor),
[3]Semyung University (Professor)

요 약

본 논문에서는 RLWE 기반 암호 알고리즘인 NewHope에 Error Correcting Code(ECC)를 적용한 RLWE 기반의 암호 알고리즘 μ-Hope를 제안한다. 기존의 NewHope는 소수로 12289를 사용하여, 공개키, 개인키, 암호문 사이즈가 각각 928-byte, 1888-byte, 1120-byte로 다른 RLWE 기반 알고리즘에 비하여 그 사이즈가 크다고 할 수 있다. 본 논문에서는 공개키, 개인키, 암호문 크기를 줄이기 위하여 소수 12289를 769로 변경한 μ-Hope를 제안하며 소수의 변경으로부터 발생하는 복호화 실패율을 줄이기 위해 ECC로 XE1을 채택하였다. 그 결과 NewHope 대비 공개키, 개인키, 암호문의 사이즈가 각각 38%, 37%, 37% 감소했다. 또한, 키 사이즈가 줄 뿐만 아니라, ECC의 사용으로 인한 성능 저하보다 작은 소수를 사용하면서 발생하는 연산 효율성이 더 커서 한 번의 키를 교환하는 과정에서 총 25%의 성능 향상도 이룰 수 있었다.

ABSTRACT

In this paper, we propose a new RLWE-based scheme named μ-Hope that exploits Error Correcting Code(ECC) on NewHope. The previous parameters of NewHope uses 12289 as a prime modulus, and the size of the public key, private key, and ciphertext is 928-byte, 1888-byte, and 1120-byte respectively, which can be said to be larger than other RLWE based algorithms. In this paper, we propose μ-Hope, which changes modulus 12289 to 769 to reduce the size of the public key, private key, and ciphertext. Also, we adopts XE1 as an Error Correcting Code(ECC) to solve the increased decryption failure rate caused by using a small prime modulus. As a result, the size of the public key, private key, and ciphertext decreased by 38%, 37%, and 37% respectively. As the computational efficiency caused by using a small prime modulus exceeds the performance degradation by exploiting ECC, this result in 25% performance improvement for a single key exchange.

Keywords: Post-quantum cryptography, RLWE, Lattice-based cryptography, Error Correcting Code

Received(08. 03. 2020), Accepted(08. 18. 2020)
* 이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2019-0-00033, 미래컴퓨팅 환경에 대비한 계산 복잡도 기반 암호 안전성 검증 기술개발)
† 주저자, yup1228@naver.com
‡ 교신저자, shhong@korea.ac.kr(Corresponding author)

I. 서 론

현재 사용되는 RSA와 타원곡선 암호 등의 공개키 기반 암호는 인수분해와 이산대수 문제의 어려움에 기반하고 있다. 하지만 이러한 문제들은 양자컴퓨터가 개발되어 Shor 알고리즘이 구현될 경우 다항시간 안에 해결할 수 있어 안전하지 않게 된다[1]. 최근 양자 컴퓨터의 개발이 가시화되었고 이에 대응하기 위해 고전 컴퓨터에서도 사용할 수 있으면서 양자컴퓨팅 환경에서 안전한 양자 내성 암호에 관한 연구가 활발하게 진행되고 있다. NIST는 이러한 변화에 따라 양자 내성 암호 표준화 공모전을 진행 중이며, 현재는 round 3 후보까지 결정되었다. 이 중 격자(Lattice) 기반 암호는 round 2 기준 26개의 후보 중 12개를 차지하였고 round 3에서는 대체 후보를 포함하여 15개의 후보 중 7개를 차지할 만큼 유망한 분야로 주목받고 있다.

격자 기반 암호의 한 갈래인 RLWE는 LWE의 큰 키 사이즈와 느린 연산속도를 개선한 방식으로 LWE 기반 암호를 $R_q=\mathbb{Z}_q[x]/(x^n+1)$ 위에서 정의한 것이다[2][3]. RLWE 기반 암호에서는 기본 연산으로 다항식 곱셈을 사용하는데, R_q상에서 정의된 다항식 곱셈은 특정 조건을 만족하는 n,q를 사용할 때 Number Theoretic Transform(NTT)을 사용하여 빠르게 처리할 수 있다. 일반적인 방식의 다항식 곱셈은 $O(n^2)$의 복잡도를 가지지만 NTT를 사용할 경우 $O(n\log n)$으로 감소한다.

NTT의 사용은 RLWE 기반 암호 알고리즘의 연산 효율성을 증가시켰지만, 사용 조건을 맞추기 위해서는 제한적인 파라미터를 설정할 수밖에 없었으며, 이로 인해 기존 공개키 암호보다 키 사이즈가 크다는 단점을 가져왔다. 이에 대하여, 최근에는 격자 기반 암호의 키 사이즈를 줄이기 위해 파라미터 조건을 완화하는 연구가 진행되었다. [4],[5]에서는 NTT의 변환 과정을 간소화함으로써 조건을 완화하여 키 사이즈를 줄일 수 있게 하였다.

키 사이즈를 줄이기 위한 다른 방법으로 LAC에서는 NTT를 사용하지 않고 1-byte 크기의 작은 modulus인 251을 사용하는 방법을 채택했다[6]. 이 과정에서 복호화 실패율이 높아지게 되었고 이를 해결하기 위하여 ECC를 사용했다. 이처럼 RLWE 기반 알고리즘의 키 사이즈를 줄이기 위한 많은 연구가 진행되고 있다.

한편, NewHope는 Alkim 등에 의하여 개발된 최초의 Diffie-Hellman 구조의 RLWE 기반 키 교환 알고리즘이다[7]. NIST 표준화 공모전 round 2의 후보인 NewHope는 Google의 인터넷 브라우저인 Chrome에 적용하여 실험됐다. 하지만 128-bit 보안 강도에 대해서 R_q에서 q를 12289로 사용하여 개인키 사이즈가 1888-byte에 달한다. 이는 고전 컴퓨팅 환경에서 같은 강도를 가진 현재 사용하는 RSA3072(384-byte), ECC256(32-byte)에 비해 키 사이즈가 크다는 단점이 존재한다.

본 논문에서는 NewHope를 변형한 μ-Hope를 제안한다. μ-Hope는 NewHope에서 사용되는 파라미터를 변형하고 ECC를 적용한 것이다. 일반적으로 ECC의 사용은 암호 알고리즘의 성능에 부하를 준다. 이에 대응하여 μ-Hope에서는 소수의 크기와 발생할 수 있는 복호화 실패율의 최적점을 계산하여 연산 부하를 최소화하고 [4]에서 제안한 최신의 연산 기법을 적용했다. 그 결과 μ-Hope는 같은 128-bit 보안 강도를 가진 NewHope-CCA-KEM512 대비 공개키, 개인키 그리고 암호문 사이즈가 약 37% 감소했으며, 속도는 25% 빠르다.

본 논문의 2장에서는 논문의 전개에 필요한 배경 지식에 대하여 설명한다. 3장에서는 제안하는 암호 알고리즘인 μ-Hope의 전체적인 내용을 기술한다. 4장에서는 안전성 분석을 진행하고 5장에서는 μ-Hope와 NewHope의 성능을 비교한다. 마지막으로 6장에서는 결론을 맺는다.

II. 배경 지식

2.1 표기법

본 논문에서 사용되는 몇 가지 표기법을 정리하면 다음과 같다.

- B : $\{0,1,\cdots,255\}$를 나타내는 집합.
- q : 다항식의 각 계수를 정의하는 소수로 $q=k\cdot 2^m+1$를 만족한다.
- $R_q=\mathbb{Z}_q[x]/(x^n+1)$: x^n+1을 법으로 하는 다항식 환. R_q상의 다항식의 각 계수는 $\mathbb{Z}_q$로 정의된다. 이때, 정수 k에 대하여 $n=2^k$을 만족한다.
- $a\xleftarrow{\$}\chi$: 분포 χ를 따라 a를 샘플링.

- ψ_k : 표준편차가 $\sigma = \sqrt{\frac{k}{2}}$ 인 중심이항분포. 이 분포에서의 샘플링 과정은 유니폼하고 독립인 $b_i, b_i' \in \{0,1\}$ 에 대하여 $\sum_{i=0}^{k-1}(b_i - b_i')$ 를 통해 이뤄진다.
- $\hat{a}$: 다항식 a가 NTT domain에 있음을 의미한다. 즉, $\hat{a} = NTT(a)$임을 의미한다.

2.2 RLWE

두 양의 정수 n, q와 이들을 이용하여 구성한 R_q가 주어졌다고 하자. 또한, χ_s, χ_e를 각각 s, e를 샘플링하기 위한 R_q에서의 분포라 하고 a를 R_q에서 유니폼 랜덤하게 샘플링 한다면 RLWE는 두 문제로 나누어 생각할 수 있다. 하나는 Decision RLWE로 두 분포 $(a, b = as + e)$와 (a, u)를 구분하는 문제이다. 이때 u는 R_q에서 유니폼하게 샘플링한 것을 의미한다. 즉, 유니폼 분포인지 만들어진 값인지를 구분하는 문제라 말할 수 있다. 다음으로는 Search RLWE로 $(a, b = as + e)$가 주어졌을 때 s를 복원하는 문제로 정의된다. 이는 에러로 사용되는 e를 알지 못하면 비밀정보 s를 알아내기 어려움을 의미한다. 즉, RLWE는 에러 e를 추가하여 안전성을 확보하는 방식이며, 일반적으로 에러는 가우시안 분포를 따른다.

RLWE의 안전성은 정의된 다항식 환의 차수 n과 식(1)의 error rate α로부터 결정되고 소수 자체의 크기와 관계가 없음이 알려져 있으며([6],[8]) α에 대한 식은 다음과 같다.

$$\alpha = \frac{\sigma}{q}\sqrt{2\pi}\ ,\ \sigma : \text{오류 분포의 표준편차} \qquad (1)$$

위 식(1)에서 볼 수 있듯이 소수 자체의 크기보다는 q와 σ의 비율인 $\frac{\sigma}{q}$가 안전성에 영향을 미친다. 그러므로 선택된 소수에 따라 에러를 샘플링하는 분포의 표준편차를 적절히 선택하는 것이 중요함을 알 수 있다.

한편, error rate α가 클수록 안전성은 높아지지만, 실패율에 영향을 준다. 실패율이 높아질 경우 [9]에서 제안한 방식의 공격에 노출될 수 있으므로 적절한 error rate를 달성해야 안전성을 확보할 수 있다.

2.3 NewHope

2016년 Alkim 등에 의하여 제안된 NewHope는 Diffie-Hellman 스타일의 RLWE 기반 키 교환 프로토콜(Fig. 1.) 이다[7]. NIST 표준화 공모전 round 2의 후보 중 하나이며 Public Key Encryption(PKE), Key Encapsulation Mechanism(KEM)으로 이용되고 있다. NewHope의 KEM은 키 교환 프로토콜을 기반으로 만들어졌으므로 본 논문에서는 키 교환 프로토콜을 기준으로 설명하고 자세한 내용은 [10]을 참고한다. 다음의 Fig. 1.은 NewHope의 키 교환 프로토콜을 나타내며 이어지는 소절에서는 프로토콜에서 사용되는 함수에 대하여 간략히 설명한다.

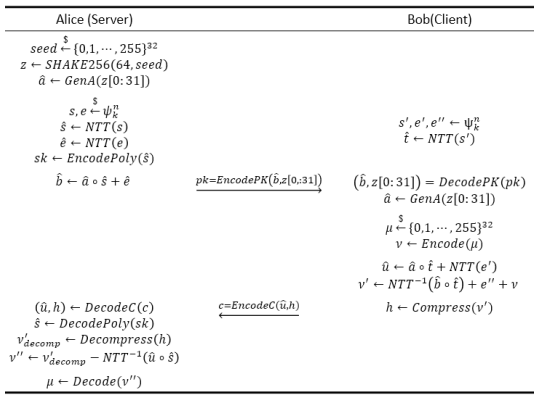


Fig. 1. NewHope key exchange protocol.

2.3.1 샘플링

NewHope의 샘플링은 공개키 a를 샘플링 하는 것과 e, s를 샘플링 하는 것으로 나뉜다. 두 경우 모두에서 표준으로 선정된 해시 함수인 SHAKE를 사용한다[11]. 먼저 공개키 a의 각 계수는 $\mathbb{Z}_q$상의 유니폼 분포를 따른다. 이를 샘플링 하는 함수는 $GenA$로 32-byte seed를 입력으로 하여 512개의 계수를 가진 a를 출력한다. 이때, 32-byte seed를 512개의 계수로 늘리기 위해서 SHAKE128을 내부 함수로 사용한다.

s, e를 샘플링하는 부분은 Fig. 1.의 $s, e \leftarrow \psi_k^n$

이다. $s,e \leftarrow \psi_k^n$은 다항식 s,e의 각 계수를 ψ_k에서 샘플링 함을 의미한다. 즉, s,e의 각 계수는 ψ_k를 따른다. 샘플링 과정에서 SHAKE256$(128, seed)$를 내부 함수로 사용하는데, $seed$를 입력으로 하여 128-byte의 결과를 출력하는 함수이다. 이때, NewHope에서는 $k=8$을 사용하므로 SHAKE256을 한 번 호출할 때 64개의 계수를 생성할 수 있다. 즉, 다항식 하나를 생성하기 위해서 해시 함수를 8번 호출해야 한다[10].

2.3.2 Number Theoretic Transform

Fig. 1.에 $NTT(s)$와 같이 표기된 NTT는 Fast Fourier Transform(FFT)를 정수 위에서 정의한 것으로 Toom-Cook, Karatsuba와 함께 최적화된 다항식 곱셈 알고리즘 중 하나로 알려져 있다[12]. 세 알고리즘 중에서도 NTT는 속도와 메모리 효율성 측면에서 우수하여 격자 기반 암호 시스템에서 자주 사용된다[10][13][14]. 특히 RLWE에서 연산 부하가 가장 큰 부분 중 하나는 다항식 곱셈 부분으로 암호 알고리즘의 성능 향상을 위해서는 최적화가 필수적인 부분이다. 일반적으로 다항식 곱셈은 text-book multiplication을 사용했을 때 $O(n^2)$의 계산 복잡도를 가지지만 NTT를 적용할 경우 $O(n\log n)$의 복잡도로 감소하게 된다.

NTT를 사용한 다항식 곱셈은 $a,b,c \in R_q$에 대하여 $q \equiv 1 \pmod{2n}$, $n=2^k$이 되도록 n,q가 잘 정의되어 있을 때 $c = a \cdot b \pmod{(x^n+1)}$을 효율적으로 계산할 수 있다. $\tau \in \mathbb{Z}_q$를 primitive 4-th root of unity라 할 때, 다음 식(2)와 같은 성질을 이용한다.

$$\begin{aligned} &\mathbb{Z}_q[x]/(x^{2^k}+1) \\ &\cong \mathbb{Z}_q[x]/(x^{2^{k-1}}-\tau) \times \mathbb{Z}_q[x]/(x^{2^{k-1}}+\tau) \end{aligned} \tag{2}$$

위와 같은 동형사상(isomorphism)을 $8, 16, \cdots 2n-th$ root of unity에 대하여 반복하면 다음의 최종 단계에 도달한다.

$$\mathbb{Z}_q[x]/(x^{2^k}+1) \cong \prod_{i=1}^{2^k} \mathbb{Z}_q[x]/(x-\zeta_i) \tag{3}$$

$,\zeta_i$: primitive $2n-th$ root of unity

식(3)의 최종 단계에 도달하면 $a,b \in R_q$는 R_q와 동형(isomorphic) 환(ring)에서 정의할 수 있게 되고, 이는 $n-1$차 다항식을 n개의 상수항으로 생각할 수 있음을 의미한다. 따라서 기존의 다항식 곱셈은 식(3)의 동형 환에서의 n개의 상수항 곱셈으로 처리된다.

일반적으로 식(3)과 같은 형태로 분해하는 과정을 Forward NTT라 정의하며 $NTT(a)$로 표기한다. Inverse NTT는 Forward NTT의 역과정으로 $NTT^{-1}(a)$로 나타낸다. 즉, $a = NTT^{-1}(NTT(a))$가 성립한다. 이를 이용하여 $a,b \in R_q$ 두 다항식의 곱은 다음 식(4)와 같이 표현할 수 있으며 $\circ$는 component wise 곱셈을 의미한다.

$$c = NTT^{-1}(NTT(a) \circ NTT(b)) \tag{4}$$

2.3.3 암호문 압축 기법

Fig. 1.의 Bob 측에서 사용되는 *Compress*와 Alice 측에서 사용되는 *Decompress*는 RLWE 기반 알고리즘에서 자주 사용되는 기법이다. 이 기법은 암호문의 크기를 줄일 수 있어 RLWE의 단점을 일부 해결할 수 있다. *Compress*는 입력으로 다항식을 받아 각 계수를 d-bit로 압축한 후 byte 표현으로 바꿔주는 함수이다. 이때, d는 압축하려는 bit 수를 나타내며 NewHope에서는 $d=3$을 사용하고 있다. *Compress* 함수와 *Decompress* 함수는 역함수에 가까운 관계로 *Compress*, *Decompress*를 순서대로 거쳐 나온 데이터를 x'이라 하면 $|x'-x| \le \frac{q}{2^{d+1}}$의 관계가 성립한다[13].

2.3.4 인코딩과 디코딩

Fig. 1.의 NewHope 키 교환 프로토콜에서 사용되는 인코딩(encoding)과 디코딩(decoding)은 크게 두 가지로 분류할 수 있다. 하나는 다항식 표현을 byte 표현으로 바꾸는 *EncodePoly*이고, 다른 하나는 평문을 다항식 표현으로 바꿔주는 *Encode*이다. 먼저 *EncodePoly*는 각 계수가 14-bit로 이루어진 512개의 계수를 가지는 다항식을 byte 단위로 변환하는 과정이다. 이는 데이터를 전송할 때 대역폭을 줄이기 위함이다. 예를 들어 14-bit 데이터는 16-bit 자료형에 들어갈 수 있지만, 2-bit의 낭비가

발생한다. 이를 방지하기 위해 14-bit 데이터를 8-bit 자료형 두 개에 담고 또 다른 14-bit 데이터에서 2-bit를 가져와 남은 공간에 담아주는 과정이다. $Encode\,C$와 $Encode\,PK$역시 다항식 표현을 byte 표현으로 바꿔주는 함수로 두 함수 모두 내부적으로 $Encode\,Poly$를 이용한다. $Encode\,C(\hat{u},h)$에서 $\hat{u}$는 다항식 표현이므로 $Encode\,Poly$를 이용하여 byte 표현으로 바꿔주고, h는 byte 표현으로 함수에 입력으로 주어지므로 뒤에 이어 붙여준다. $Encode\,PK$역시 $Encode\,C$와 유사하게 작동한다.

반면 Fig. 1.의 $Encode(\mu)$는 256-bit 데이터인 평문 μ를 512개의 계수를 가진 다항식으로 변환하게 된다. μ의 각 bit에 $q/2$를 곱해주는 방식으로 변환을 진행한다. 즉, 256-bit를 256개의 계수로 변환하는 과정으로 생각할 수 있다. NewHope에서는 512개의 계수를 필요로 하므로 나머지 256개의 계수를 생성하기 위해 이미 생성된 데이터를 복사하여 총 512개의 계수를 만든다. 즉, 하나의 bit가 2개의 계수로 대응되는 것이며 이를 1-bit to 2 coefficient 기법이라 한다.

디코딩 과정은 인코딩 과정의 역과정이다. 두 과정은 모두 단순히 데이터의 표현 방식만 바꾸는 것이므로 인코딩과 디코딩을 진행한다고 하더라도 데이터의 손실은 발생하지 않는다. 인코딩과 디코딩은 암호 알고리즘마다 변환하는 방식을 구성하는 데 차이가 있을 수 있다.

2.3.5 비트 에러와 복호화 실패

복호화 실패는 격자 기반 암호시스템에서 발생하는 현상이다. Fig. 1.에서 평문을 암호화하는 과정(Bob side)과 그런 암호문을 복호화하는 과정(Alice side)에서 Alice와 Bob이 동일한 평문을 가지지 못하는 경우를 말한다. 일반적으로 RLWE 기반 암호 알고리즘에서는 평문에 가해지는 error, compression noise(식 (5)의 c)들의 크기의 합에 의하여 bit error의 발생이 결정되고, 이런 bit error가 발생할 확률을 bit error rate라 한다. 이때 bit error rate를 누적해서 최종 복호화 실패율(Decryption Failure Rate, DFR)을 계산할 수 있다.

NewHope 프로토콜에서 발생하는 bit error rate는 암호화와 복호화 과정에서 평문 μ에 가해지는 오류의 총합을 식(5)로 표현하여 계산할 수 있다. 단, 프로토콜 상의 NTT, NTT^{-1}의 표기는 편의상 생략한다.

$$\begin{aligned} v'_{decomp} - us &= h + c - (at + e')s \\ &= bt + e'' + v + c - ats - e's \\ &= (as + e)t + e'' + v + c - ats - e's \\ &= et + e'' + v + c - e's \\ &= et + e'' + c - e's + Encode(\mu) \\ &= et + e'' + c - e's + \frac{q}{2}\mu \end{aligned} \tag{5}$$

$v'_{decomp} - us$는 $Decode$ 함수에 의해 평문 m으로 복원된다. 이때, $et + e'' + c - e's$를 w라 정의하면 $|w_i| > \frac{q}{4}$일 때 평문의 i번째 bit에 오류가 발생하여 Decode 함수에 의해 복원된 평문이 올바르지 않게 된다. 따라서, bit error rate는 $1 - \Pr[-\frac{q}{4} < w_i < \frac{q}{4}]$로 정의된다.

Decryption Failure Rate(DFR)는 복호화된 평문이 암호화되기 전 평문과 1-bit라도 다를 확률을 의미한다. 일반적으로 DFR의 계산은 모든 bit가 독립이라는 가정하에 다음 식으로 계산된다.

$$1 - \prod_{i=0}^{n-1} \Pr[-\frac{q}{4} < w_i < \frac{q}{4}]$$

2장의 내용을 정리하자면, 실제 활용에서 NewHope는 소수로 $q = 12289$를 사용하고 s, e에 대한 분포로 ψ_8을 사용한다. AES-128 안전성에 준하는 NIST category 1의 경우 $n = 512$를 사용하고 AES-256 안전성에 준하는 NIST category 5에 대해서는 $n = 1024$를 사용한다. 이때, 12289는 $n = 512, 1024$ 모두에 대하여 NTT를 사용할 수 있게 해주는 가장 작은 소수로써 NewHope의 성능에 큰 영향을 주는 파라미터이다. 하지만 NTT의 사용을 위해 선택한 12289는 동일한 안전 강도를 가지는 다른 RLWE 기반 알고리즘보다 큰 키 사이즈를

Table 1. Sizes of public keys, secret keys and ciphertexts of NewHope and Kyber in bytes

	pk	*sk*	*ct*	Total
NewHope	928	1,888	1,120	3,936
Kyber	800	1,632	736	3,168

가지게 된다. 이는 다음의 Table 1.에 잘 나타나며, NIST category 1에 속한 NewHope-CCA-KEM512와 Kyber-CCA-KEM512를 비교한 것이다.

2.4 오류 정정 부호 - XEf

본 절에서는 NewHope의 modulus를 변경하고 이로 인해 발생하는 에러를 정정하기 위해 사용한 ECC 알고리즘에 대해 간단히 설명한다. 격자 기반 암호에 추가적인 ECC의 사용은 암호 알고리즘의 성능에 부하를 주므로 어떤 ECC를 사용할지에 대한 선택은 매우 중요하다. 본 논문의 3.4절에서 필요한 오류 수정량을 계산한 결과, 1-bit의 오류만 수정하면 충분히 작은 DFR에 도달할 수 있다. 따라서 우리는 1-bit 오류 수정 한도를 가지는 XE1을 채택했으며 본 절에서는 이에 대하여 간략하게 언급한다.

XEf는 f-bit 오류 수정 한도를 가진 ECC로 [15]에서 XE5로 소개된 이후 [16]으로 넘어오면서 XEf로 일반화되었다. XEf는 linear block code의 일종으로 인코딩과 디코딩 과정이 단순하여 구현이 간단하고 다른 분류의 ECC보다 빠른 속도를 가진다. 하지만 일반적으로 linear block code는 오류 수정 한도가 커지면 복잡도가 급격하게 증가한다는 단점이 있다. 이러한 관점에서 XE1은 1-bit만 수정하므로 복잡도를 많이 증가시키지 않는다. 따라서, μ-Hope에서 사용하는 ECC는 LAC[6]에서 사용하는 BCH와 Round5[16]에서 사용하는 XE5보다 연산 부하가 적다.

XE1은 인코딩 과정에서 256-bit message $m = \{m_0, m_1, \cdots, m_{255}\}$에 대하여 32-bit redundancy data $r = \{r_0, r_1\}, |r_i| = 16$ for $i = 0,1$를 생성하며 생성 방법은 다음과 같다.

$$r_0[j] = \sum_{k \equiv j \bmod 16} m_k \pmod 2$$
$$r_1[j] = \sum_{i=0}^{15} m_{j \cdot 16+i} \pmod 2$$

이때, $r_i[j]$는 r_i의 j번째 bit를 의미한다. 이렇게 생성된 r로 288-bit code word $c = m \| r$을 전송하고, 수신된 code word가 $c' = m' \| r'$이라 할 때 m'을 이용하여 새로운 r''을 만들고 다음을 만족할 경우 오류로 판단하여 m_k'을 반전시킨다.

$$\sum_{i=0}^{1}((r_i'[k \bmod 16] - r_i''[k \bmod 16]) \bmod 2) \geq 2$$

III. μ-Hope

본 장에서는 NewHope의 파라미터를 변경하고 ECC를 적용한 μ-Hope에 대하여 설명한다. NewHope의 파라미터를 변경하는 과정에서 커진 복호화 실패율에 대하여 분석하고, 이를 완화하기 위해 몇 비트의 에러를 수정해야 하는지 분석한다.

μ-Hope를 소개하기에 앞서, Table 2.는 NIST category 1에 대한 μ-Hope의 파라미터를 정리하여 NewHope와 비교한 것이다. 본 장에서는 μ-Hope의 파라미터 설정의 정당성과 이로 인해 변경되어야 하는 사항을 중점적으로 서술한다.

RLWE 기반 암호 알고리즘에서 modulus는 키 사이즈와 성능에 가장 많은 영향을 미치는 파라미터다. 에러를 주어 안전성을 확보하는 RLWE의 특성상 q가 커질수록 표준편차가 더 큰 분포를 사용해야 한다. 이는 키 사이즈의 증가와 다항식 곱셈의 연산량을 증가시키는 요소가 된다. 이러한 효과를 줄이기 위해 μ-Hope는 소수로 769를 사용한다. 769는 $3 \cdot 256 + 1$로 표현되며 $769 \equiv 1 \pmod{256}$을 만족하게 되어 [4],[5]에 소개된 방식의 NTT를 이용한 효율적인 연산이 가능한 수다. 또한, 기존 NewHope에서 사용한 소수인 12289는 14bit 소수로 10-bit 소수인 769보다 40% 정도 더 큰 키 사이즈를 가지게 된다. 따라서 769를 사용할 경우 속도와 키 사이즈 측면에서 더 좋은 결과를 얻을 수 있다.

Table 2. Parameter comparison of μ-Hope and NewHope

Parameter set	μ-Hope	NewHope
Dimension n	512	512
Modulus q	769	12289
Noise parameter k	1	8
NTT parameter γ	7	10968
Compression parameter d	4	3
Decryption failure rate	2^{-157}	2^{-213}
Post-quantum bit security	112	101
NIST category	1	1

Fig. 2.~ Fig. 4.은 μ-Hope-CPA-PKE에 대한 알고리즘을 제시한 것이다. Fig. 2.은 키 생성, Fig. 3.는 암호화, 그리고 Fig. 4.는 복호화 과정을 나타낸다. 기본적으로 μ-Hope는 NewHope를 변형한 것이므로 알고리즘 대부분은 NewHope와 같다. 또한, 본 논문에서 성능 측정은 NewHope-CCA-KEM512와 비교하여 진행한 결과이지만, NewHope의 KEM에서 사용되는 CPA-PKE를 μ-Hope의 CPA-PKE로 변형하여 적용하면 되기 때문에 μ-Hope의 KEM 알고리즘은 따로 제시하지 않는다.

Alg 1. μ-Hope.CPAPKE.KeyGen

Output : $sk \in B^{5 \cdot n/4}, pk \in B^{5 \cdot n/4+32}$

1. $seed \xleftarrow{\$} B^{32}$
2. $z \leftarrow \text{SHAKE256}(64, seed)$
3. $publicseed \leftarrow z[0:31]$
4. $noiseseed \leftarrow z[32:63]$
5. $\hat{a} \leftarrow GenA(publicseed)$
6. $s \leftarrow Sample(noiseseed, 0)$
7. $\hat{s} \leftarrow NTT(s)$
8. $e \leftarrow Sample(noiseseed, 1)$
9. $\hat{e} \leftarrow NTT(e)$
10. $\hat{b} \leftarrow \hat{a} \circ \hat{s} + \hat{e}$
11. $pk = EncodePK(\hat{b}, publicseed)$
12. $sk = EncodePoly(\hat{s})$
13. **return** (pk, sk)

Fig. 2. μ-Hope.CPAPKE key generation.

Alg 2. μ-Hope.CPAPKE.Enc

Input : $(pk \in B^{5 \cdot n/4+32},\ \mu \in B^{32}, coin \in B^{32})$
Output : $c = (c_1, c_2)$

1. $(\hat{b}, publicseed) \leftarrow DecodePK(pk)$
2. $\hat{a} \leftarrow GenA(publicseed)$
3. $s' \leftarrow Sample(coin, 0)$
4. $e' \leftarrow Sample(coin, 1)$
5. $e'' \leftarrow Sample(coin, 2)$
6. $\hat{t} \leftarrow NTT(s')$
7. $\hat{u} \leftarrow \hat{a} \circ \hat{t} + NTT(e')$
8. $v \leftarrow EncodeM(\mu)$
9. $v' \leftarrow (NTT^{-1}(\hat{b} \circ \hat{t}) + e'')_{l_v} + v$
10. $h \leftarrow Compress(v')$
11. $c = EncodeC(\hat{u}, h)$
12. **return** c

Fig. 3. μ-Hope.CPAPKE encryption.

Alg 3. μ-Hope.CPAPKE.Dec

Input : $c \in B^{5 \cdot n/4+n/2}$
Output : $\mu \in B^{32}$

1. $(\hat{u}, h) \leftarrow DecodeC(c)$
2. $\hat{s} \leftarrow DecodePoly(sk)$
3. $v' \leftarrow Decompress(h)$
4. $\mu \leftarrow DecodeM(v' - (NTT^{-1}(\hat{u} \circ \hat{s}))_{l_v})$
5. **return** μ

Fig. 4. μ-Hope.CPAPKE decryption

3.1 ψ_1위에서의 샘플링

μ-Hope에서는 $k=1$로 하여 평균이 0, 표준편차가 $\sqrt{\frac{1}{2}}$인 ψ_1을 error와 secret의 샘플링을 위한 분포로 사용한다. 이러한 선택은 $q=769$로 설정했을 때 NIST category 1에 속하도록 하기 위함이다. ψ_1에서 샘플링을 하기 위한 함수로 $Sample(coin, i)$를 사용하고 $coin$은 seed를, i는 nonce값을 의미한다.

한편, ψ_1을 사용할 경우 NewHope512에서 사용한 ψ_8을 사용할 때보다 샘플링 과정에서 많은 클럭 사이클을 절약할 수 있다. 그러한 이유는 NewHope에서 SHAKE256을 한 번 호출할 때 128-byte의 랜덤 값을 선택하게 된다. 이를 이용하여 coefficient를 구성하는 과정에서 ψ_8을 사용할 땐 64개의 coefficient를 만들 수 있지만, ψ_1을 사용할 경우 512개의 coefficient 전체를 만들 수 있기 때문이다. 이는 해시 함수를 적게 호출하게 되어 효율성을 높이게 된다.

3.2 NTT 기법의 적용

μ-Hope에서는 modulus를 줄이면서 NTT의 사용 조건인 $q \equiv 1 \pmod{2n}$를 만족하지 못하여 기존의 NTT를 사용할 수 없게 되었다. 최근 NTT의 사용 조건을 완화할 수 있는 격자 기반 암호에 최적화된 NTT가 많이 연구되었고, 그 결과 작은 소수를 사용하면서도 더 빠른 연산속도를 가지게 되었다 [4][5]. 두 논문에 소개된 연산 방식은 모두 좋은 성능을 가졌지만, [4]의 방식이 직관적으로 이해하기 쉬우므로 μ-Hope에서는 [4]에 소개된 방식을 따른다.

[4]방식의 NTT는 식(3)의 동형사상을 이용하는 것이 아닌 식(6)을 이용한다.

$$\mathbb{Z}_q[x]/(x^{2^k}+1) \cong \prod_{i=1}^{2^{k-2}} \mathbb{Z}_q[x]/(x^4-\zeta_i) \quad (6)$$

$,\zeta_i$: primitive $\frac{n}{2}-th$ root of unity

위의 동형 환까지 도달하는 과정은 기존의 NTT와 같다. 하지만 최종 단계의 동형 환까지 도달하는데 필요한 연산량이 감소한다. 이러한 방식을 사용할 경우 NTT 이후 진행되던 coefficient-wise multiplication은 3차 다항식 곱셈으로 대체된다. 일반적으로 다항식 곱셈은 coefficient-wise multiplication보다 연산량이 많지만, one-iteration Karatsuba를 적용하여 최적화할 수 있다[17]. 그 결과 [4]방식의 NTT를 사용한 다항식 곱셈은 기존의 NTT를 사용한 다항식 곱셈보다 15%정도 좋은 성능을 가진다[4].

3.3 암호문 압축 기법의 적용

μ-Hope는 4-bit compression을 사용한다. Compression 기법은 암호문의 크기를 줄이지만, compression noise라 불리는 새로운 에러 $\left[-\text{round}\left(\frac{q}{2^{d+1}}\right), \text{round}\left(\frac{q}{2^{d+1}}\right)\right]$가 추가된다[13]. 이때, round$(x)$는 일반적으로 사용되는 반올림의 정의와 같다. 예를 들어 round $(1.5)=2$를 만족한다. Compression noise는 e외의 또 다른 에러가 추가되는 것이므로 안전 강도를 높이지만 복호화 실패율을 높이는 부작용도 존재한다. 대부분의 RLWE 기반 암호 알고리즘에서는 compression noise에 의한 안전성 증가는 미미하므로 안전성 분석에서 생략한다. 따라서, μ-Hope의 안전성 역시 compression noise에 대한 부분은 생각하지 않으며, 실제 안전성은 Table 2.의 안전성보다 높거나 같다. 단, 복호화 실패율에 미치는 영향에 대해서는 분석하도록 한다.

3.4 오류 정정 부호 XE1의 적용

μ-Hope에서는 modulus를 작게 하여 발생하는 에러를 수정하기 위해 ECC를 사용한다. ECC는 안전성에 영향을 주지 않지만, 성능 저하의 요인이 될 수 있으므로 적절한 ECC를 선택해야 성능 저하를 최소화할 수 있다. 따라서 먼저 ECC를 사용하지 않은 경우의 DFR을 분석하고 몇 bit의 에러를 수정해야 하는지 결정한다.

DFR의 계산은 bit error rate의 계산이 선행되어야 한다. Compression noise가 없다는 가정하에 bit error rate $\delta \approx 1-erf\left(\frac{q/4}{\sqrt{2}(\sigma^2\sqrt{2n})}\right)$로 계산된다[6]. 이때, μ-Hope는 $n=512$, $q=769$ 그리고 분포로 표준편차 $\sigma=1/\sqrt{2}$를 가지는 ψ_1을 사용하고 있으므로 bit error rate는 다음의 식(7)과 같다.

$$\delta \approx 1-erf\left(\frac{769/4}{\sqrt{2}((1/\sqrt{2})^2\sqrt{2\cdot 512}}\right) \approx 2^{-108} \quad (7)$$

위 식(7)의 error rate에 compression noise인 $[-24,24]$위에서의 uniform random error가 더해지면 최종 bit error rate는 2^{-87}으로 계산된다. bit error rate의 계산은 Kyber의 계산 방식을 따랐다[13]. 여기서 우리는 모든 bit가 독립이라는 가정하에 식(8)과 같이 DFR을 계산할 수 있다.

$$(1-(1-2^{-87})^{512}) \approx 2^{-78} \quad (8)$$

NIST에서는 RLWE 기반 암호 알고리즘의 DFR을 2^{-128} 이하로 유지할 것을 요구하고 있다. 하지만 식(8)로 구해진 DFR은 이 요구사항을 만족시키지 못한다. 여기서 1-bit를 수정하게 되면 다음 식(9)의 DFR을 가진다.

$$1-((1-2^{-87})^{512}+(512\cdot 2^{-87})(1-2^{-87})^{511}) \approx 2^{-157} \quad (9)$$

위 식은 512-bit 중 1-bit의 오류가 있을 확률과 1-bit의 오류도 없을 확률의 합을 나타낸다. 즉, 1-bit의 오류만 수정할 수 있으면 적당한 DFR을 달성할 수 있다. 따라서 2.4절에서 언급한 1-bit 에러를 수정할 수 있는 XE1을 사용한다.

μ-Hope의 인코딩 과정은 먼저 XE1을 사용하여 인코딩을 진행한다. 이후 NewHope처럼 각 bit에 $q/2$를 곱하는 과정을 거치면 μ-Hope의 인코딩 과정이 완료된다. 이때, 평문의 XE1 인코딩 결과는 $l_v=$ 288-bit이므로 μ-Hope에서는 1-bit to 2

coefficient 기법을 사용하지 않는다. 따라서, Fig. 3.의 9번째 단계에서 진행되는 연산인 $(NTT^{-1}(\hat{b} \circ \hat{t}) + e'')_{l_v} + v$를 잘 작동하게 하기 위해서는 $(NTT^{-1}(\hat{b} \circ \hat{t}) + e'')$의 앞부분 l_v개의 계수만 사용한다. 이러한 과정은 $(NTT^{-1}(\hat{b} \circ \hat{t}) + e'')$에 v가 더해질 때 계수별로 연산이 진행되고 서로 다른 위치의 계수에는 영향을 미치지 않기 때문에 가능하다. 디코딩은 인코딩의 역과정으로 진행된다. 먼저 인코딩의 결과인 288개의 계수를 확인하여 각 계수가 $[q/4, 3q/4)$에 속할 때 1로 그 외에는 0으로 변환한다. 그 결과 288-bit를 얻을 수 있고 이를 XE1의 디코딩 과정을 거치면 μ-Hope의 디코딩이 완료된다.

IV. 안전성 분석

본 장에서는 μ-Hope의 안전성 분석을 진행한다. μ-Hope는 NewHope의 파라미터를 변형한 것이므로 NewHope 알고리즘의 안전성에 의존적이다. 따라서 IND-CPA-KEM과 IND-CCA-KEM에 대한 증명은 NewHope가 안전하다는 가정하에 생략한다. 하지만, modulus의 변경에 따라 해당하는 격자의 기반 문제에 대한 안전성이 변경될 수 있으므로, 알려진 공격법인 Primal attack과 Dual attack에 대한 계산 복잡도를 측정하여 안전 강도를 제시한다. RLWE 분석에 LWE 문제에 대한 공격법을 사용하는 이유는 알려진 공격법들이 환 구조를 사용하지 않기 때문이다. 복잡도 계산 방식은 [18]을 따른다. 다음의 Table 3.은 두 공격에 대한 복잡도를 측정한 것이다. C는 classical 알고리즘에 대한 복잡도를, Q는 quantum 알고리즘에 대한 복잡도를 그리고 B는 block size를 나타낸다.

Table 3. Concrete security of μ-Hope and NewHope

		μ-Hope	NewHope
Primal	C	125	112
	Q	114	101
	B	430	384
Dual	C	124	112
	Q	112	101
	B	425	383

4.1 Primal attack

Primal attack은 LWE 문제로부터 unique-SVP 문제를 구성하고 그것을 BKZ 알고리즘을 이용해서 풀어내는 과정으로 진행된다[19]. 따라서 우리는 LWE 문제가 주어졌을 때 BKZ를 이용하여 unique-SVP의 해를 찾는 데 필요한 block size b를 측정한다. LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$, $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$가 주어지면 차원을 $d = m + n + 1$로 하는 격자 Λ를 생성할 수 있다.

$$\Lambda = \{\mathbf{x} \in \mathbb{Z}^{m+n+1} : (\mathbf{A}|\mathbf{I_m}|-\mathbf{b})\mathbf{x} = 0 \bmod q\}$$

이때 위 격자는 s와 e가 충분히 작을 때 $v = (s, e, 1)$을 unique-SVP solution으로 가진다. [20]에 따르면 공격 성공의 필요 충분 조건은 $\sigma\sqrt{b} \leqq \delta^{2b-d-1} \times q^{m/d}$이며 σ는 에러와 비밀정보의 표준편차를, $\delta = ((\pi b)^{1/b} b / 2\pi e)^{1/(2(b-1))}$을 의미한다.

Block size b를 dimension으로 가지는 sub-lattice에 대한 shortest vector 문제를 풀기 위해서는 BKZ 알고리즘에서 sieving 알고리즘을 subroutine으로 이용한다. 이러한 방식을 BKZ-core-Sieving이라 하며, 복잡도는 block size b에 의존적이다. 알려진 최상의 복잡도는 classical sieving 알고리즘에 대하여 $2^{0.292b}$이고 quantum sieving 알고리즘에 대해서는 $2^{0.265b}$이다.

4.2 Dual attack

Dual attack은 앞서 Primal attack에서 언급한 격자의 dual-lattice를 구성하여 decision-LWE 문제를 푸는 데 이용한다. 먼저 dual-lattice $\hat{\Lambda}$를 다음과 같이 구성한다.

$$\hat{\Lambda} = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^m \times \mathbb{Z}^n : \mathbf{A}^t\mathbf{x} = \mathbf{y} \bmod q\}$$

이후 BKZ 알고리즘을 이용하여 $l = \delta^{d-1} q^{n/d}$를 길이로 하는 벡터 $\boldsymbol{v} = (\mathbf{x}, \mathbf{y})$를 찾아낼 수 있다. 이때 LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$, $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$에 대하여 $\boldsymbol{v}^t\mathbf{b}$와 uniform distribution 사이의 거리는

$\epsilon = 4e^{(-2\pi^2\tau^2)}$ for $\tau = l\sigma/q$로 bound 된다. 즉, 길이를 l로 하는 벡터를 가지면 decision-LWE에 대하여 ϵ만큼의 이점을 가진다.

Primal attack과 마찬가지로 dual attack의 복잡도는 BKZ 알고리즘의 복잡도를 따라간다. 공격자는 앞서 언급한 것처럼 ϵ만큼의 이점을 가지는데 sieving 알고리즘을 $Max(1,(1/(\gamma\epsilon^2))$만큼 반복함으로써 1/2의 이점을 가지도록 할 수 있다. 이때 $\gamma = 2^{0.2075b}$이다. 즉, block size에 의하여 복잡도가 결정된다고 할 수 있다.

μ-Hope는 classical과 quantum 환경에서 dual attack에 더 낮은 안전성을 가지는 것으로 확인되었다. 따라서 3장의 Table 2.의 post quantum bit security 역시 dual attack의 복잡도를 기준으로 표기했다.

V. 구현과 비교

본 장에서는 μ-Hope의 기본 연산, 키 생성, 암호화 그리고 복호화 과정의 속도와 키, 암호문 사이즈를 NewHope-CCA-KEM512와 비교한 결과를 서술한다. 측정에 사용한 CPU는 3.0GHz의 동작 주파수를 가지는 Intel Core i7-9700 Coffeelake Refresh를 사용했으며, Ubuntu 18.04.2 LTS 운영체제상에서 gcc-7.4.0 컴파일러를 이용했다. 또한, 모든 결과는 최적화되지 않은 reference C implementation을 10,000번 실행한 중간값을 나타낸다. 표에 나오는 Ratio는 NewHope의 결과를 1로 생각했을 때 μ-Hope의 비율을 나타낸 것으로 작을수록 좋음을 의미한다.

5.1 키와 암호문 사이즈 비교

Table 4.에서 알 수 있듯이, μ-Hope는 NewHope와 비교하여 공개키에서 38%, 개인키에서 37%, 암호문에서 37%정도 작은 것으로 나타난다.

Table 4. Sizes of public keys, secret keys and ciphertexts of μ-Hope and NewHope in bytes

	pk	*sk*	*ct*	Total
NewHope	928	1,888	1,120	3,936
μ-Hope	672	1,376	816	2,864
Ratio	0.619	0.628	0.628	0.626

여기서 암호문 사이즈는 modulus q의 크기뿐 아니라 압축하려는 bit 수에 따라 달라진다. μ-Hope는 3-bit까지 압축해도 NIST에서 요구한 DFR 수치인 2^{-128}이하를 달성할 수 있지만, DFR에 충분한 margin을 두기 위하여 4-bit 압축만 진행했다.

5.2 성능 비교

Table 5.는 NIST category 1에 속하는 NewHope512와 μ-Hope의 Keypair, Encapsulation, Decapsulation의 cpu cycle을 비교한 것이다. 실험 결과 μ-Hope는 NewHope512보다 Keypair에서 43%, Encapsulation에서 30%, Decapsulation에서 1 12% 빠른 것으로 나타났다. 이러한 결과는 [4]에서 제안한 최적화된 NTT를 사용했고, modulus q가 작아짐에 따라 다항식 샘플링 과정이 단순해지는 등을 원인으로 생각할 수 있다.

Table 5. Cycle counts of NewHope and μ-Hope

	Keypair	Encap	Decap	Total
NewHope	83,278	120,887	131,760	335,925
μ-Hope	57,867	92,893	117,934	268,694
Ratio	0.561	0.699	0.882	0.75

VI. 결 론

본 논문에서는 NIST round 2의 RLWE 기반 암호 알고리즘인 NewHope를 변형한 μ-Hope를 제안했다. 그 결과 공개키, 개인키, 암호문 사이즈의 총 합이 약 37% 감소했으며, 속도는 약 25% 빠름을 알 수 있었다. 또한 μ-Hope에서 사용된 소수 769는 RLWE 기반 암호 알고리즘에서 처음 사용되는 소수로, 1-bit만 수정할 수 있는 ECC를 사용했을 때 충분한 DFR을 확보할 수 있으므로 [9]에서 제안한 공격법에 안전하며, ECC를 사용하면서도 독립 가정이 가능하다. 따라서 μ-Hope는 좋은 성능을 가지면서도 분석이 간단하다는 장점이 있는 암호 알고리즘이다.

한편, 소수 769는 NTT, 수정 한도가 적은 ECC 그리고 중심이항분포를 적용할 암호 알고리즘에서 사용할 수 있는 가장 작은 소수이다. 만약 더 작은 소수를 이용한다면, LAC에서 사용한 BCH처럼 수정

한도가 큰 ECC를 사용해야 한다. 이는 독립 가정을 할 수 없게 만들어 분석에 어려움을 주고 성능에도 큰 부하를 주므로 769는 현 상황에서 RLWE에 적용하기에 최적의 소수일 것으로 보인다.

References

[1] P.W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM review, vol. 41, no. 2, pp. 303-332, Aug. 1999.

[2] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 1-23, May, 2010.

[3] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa, "Efficient public key encryption based on ideal lattices," International Conference on the Theory and Application of Cryptology and Information Security, pp. 617-635 Dec, 2009.

[4] E. Alkım, Y.A. Bilgin, and M. Cenk, "Compact and simple RLWE based key encapsulation mechanism," International Conference on Cryptology and Information Security in Latin America, pp. 237-256, Oct, 2019.

[5] Y. Zhu, Z. Liu, and Y. Pan, "When NTT Meets Karatsuba: Preprocess-then-NTT Technique Revisited," IACR Cryptology ePrint Archive, vol. 2019, pp. 1079, Sep, 2019.

[6] X. Lu, Y. Liu, Z. Zhang, D. Jia, H. Xue, J. He, B. Li, and K. Wang "LAC: Practical Ring-LWE Based Public-Key Encryption with Byte-Level Modulus," IACR Cryptology ePrint Archive, vol. 2018, pp. 1009, Oct, 2018.

[7] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange-a new hope," In 25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 327-343, Aug, 2016.

[8] C. Peikert, O. Regev, and N. Stephens-Davidowitz, "Pseudorandomness of ring-LWE for any ring and modulus," Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, pp. 461-473, Jun, 2017.

[9] J.P. D'Anvers, F. Vercauteren, and I. Verbauwhede, "On the impact of decryption failures on the security of LWE/LWR based schemes," IACR Cryptology ePrint Archive, vol. 2018, pp. 1089, Nov, 2018.

[10] E. Alkim, et al., "NewHope - algorithm specifications and supporting documentation," NIST PQC Round 2, vol. 2, pp. 4-11, Jan, 2019.

[11] Dworkin and Morris J. "SHA-3 standard: Permutation-based hash and extendable-output functions," NIST FIPS-202, Aug, 2015.

[12] D.J. Bernstein, "Multidigit multiplication for mathematicians," Citeseer, 2001.

[13] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM," 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 353-367, Apr, 2018.

[14] V. Lyubashevsky and G. Seiler, "NTTRU: truly fast NTRU using NTT," IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 180-201, May, 2019.

[15] M. J. O. Saarinen, "HILA5: On reliability, reconciliation, and error correction for Ring-LWE encryption,"

International Conference on Selected Areas in Cryptography, pp. 192-212, Dec, 2017.

[16] H. Baan, S. Bhattacharya, S. Fluhrer, O. Garcia-Morchon, T. Laarhoven, , R. Rietman and Z. Zhang, "Round5: Compact and fast post-quantum public-key encryption," International Conference on Post-Quantum Cryptography. Springer, Cham, pp. 83-102, Jul, 2019.

[17] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba Algorithm for Efficient Implementations," IACR Cryptology ePrint Archive, vol. 2006, pp. 224, Jul, 2006.

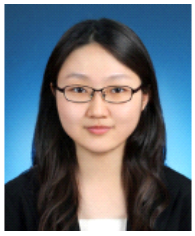
[18] M. R. Albrecht, R. Player, S. Scott, "On the concrete hardness of learning with errors," Journal of Mathematical Cryptology, vol. 9, no. 3 pp. 169-203, Oct, 2015

[19] C.P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," Mathematical programming, vol. 66, no. 1-3 pp. 181-199. Aug, 1994.

[20] E. Alkim, L. Ducas, T. Pöppelmann, P. Schwabe, "NewHope without reconciliation," IACR Cryptology ePrint Archive, vol. 2016, pp. 1157, Dec, 2016.

〈저 자 소 개〉

이 주 엽 (Juyeop Lee) 학생회원
2019년 2월: 숭실대학교 수학과 학사
2019년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 후양자암호



김 수 리 (Suhri Kim) 학생회원
2014년 2월: 고려대학교 수학과 학사
2016년 8월: 고려대학교 정보보호학과 석사
2020년 2월: 고려대학교 정보보호대학원 박사
2020년 7월~현재: K.U. Leuven ESAT/SCD-COSIC 박사후 연구원
〈관심분야〉 후양자암호, 아이소제니 기반 암호 최적화 구현

김 창 한 (Chang Han Kim) 종신회원
1985년 2월: 고려대학교 수학과 이학사
1987년 2월: 고려대학교 수학과 이학석사
1992년 2월: 고려대학교 수학과 이학박사
1992년 8월~현재: 세명대학교 정보통신학부 교수
〈관심분야〉 정수론, 공개키암호, 암호프로토콜

홍 석 희 (Seokhie Hong) 종신회원
1995년: 고려대학교 수학과 학사
1997년: 고려대학교 수학과 석사
2001년: 고려대학교 수학과 박사
1999년 8월~2004년 2월: ㈜시큐리티 테크놀로지 선임연구원
2003년 3월~2004년 2월: 고려대학교 정보보호기술연구센터 선임연구원
2004년 4월~2005년 2월: K.U. Leuven ESAT/SCD-COSIC 박사후 연구원
2005년 3월~2013년 8월: 고려대학교 정보보호대학원 부교수
2013년 9월~현재: 고려대학교 정보보호대학원 정교수
〈관심분야〉 대칭키 및 공개키 암호 알고리즘, 부채널 공격 및 대응기법, 디지털 포렌식