Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.5, Oct. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.5.709

KpqC 1 라운드 TiGER KEM의 Meet-LWE 공격에 대한 안전성 분석*

이 주 희,[1†] 이 은 민,[2] 김 지 승[3‡]
[1,2]성신여자대학교 (교수, 학생), [3]전북대학교 (교수)

Security Analysis on TiGER KEM in KpqC Round 1 Competition Using Meet-LWE Attack*

Joohee Lee,[1†] Eun-min Lee,[2] Jiseung Kim[3‡]
[1,2]Sungshin women's University (Professor, Undergraduate student),
[3]Jeonbuk National University (Professor)

요 약

최근 양자컴퓨터를 이용한 공격에 안전한 양자내성암호(Post-Quantum Cryptography, PQC)에 대한 연구 및 표준화가 국내외로 활발히 진행되고 있다. 2022년 국내 양자내성암호 표준화 공모인 KpqC 공모전이 시작되었고, 총 16종의 양자내성암호 표준 후보 알고리즘이 제출되어 1라운드 진행 중이다. 본 고에서는 KpqC 공모전 1라운드 후보인 격자 기반 키 캡슐화 알고리즘 TiGER에 대해 Alexander May의 Meet-LWE 공격을 적용하여 구체적인 공격 복잡도를 분석하였다. TiGER의 제안된 파라미터에 대해 Meet-LWE 공격을 적용한 계산 결과, 192-bit 양자 안전성을 목표로 제안된 TiGER192 파라미터가 실제로 170-bit의 classical 안전성을 가진다는 것을 보였다. 또한, 본 고에서는 Meet-LWE 공격에 대한 안전성을 높이기 위한 파라미터 설정 방법을 제언한다.

ABSTRACT

Recently, Post-Quantum Cryptography (PQC), which is secure against attacks using quantum computers, has been actively studied. In 2022, the KpqC competition, a competition for domestic PQC standardization, was launched, and a total of 16 candidate algorithms were received, and the first round is underway. In this paper, we apply Alexander May's Meet-LWE attack to TiGER, a lattice-based key encapsulation mechanism that is a candidate for the first round of the KpqC competition, and analyze its concrete attack complexity. The computational results of applying the Meet-LWE attack to each of the proposed parameters of TiGER show that the proposed TiGER192 parameter, which targets 192-bit quantum security, actually achieves 170-bit classical security. In addition, we propose a parameter setting to increase the attack complexity against the Meet-LWE attack.

Keywords: Post-Quantum Cryptography, KpqC Competition, Key Encapsulation Mechanism, TiGER

Received(07. 10. 2023), Modified(08. 11. 2023), Accepted(08. 11. 2023)

* 이 연구는 ETRI부설연구소의 위탁연구과제[2023-080]로 수행한 연구결과입니다. 본 연구는 KpqC Round 1 공식 제안문서를 대상으로 수행되었으며, 본 연구 결과는 23/07/11에 KpqC Bulletin Board[13]를 통해 최초로 공개되었습니다("Combinatorial security estimation reports on TiGER KEM"). 그 이후, 23/07/17 eprint에 수정 게시된 TiGER 논문[14]의 파라미터 확인 결과 비밀키의 해밍웨이트 값이 변동되어 Meet-LWE 공격에 안전함을 확인하였습니다.

† 주저자, jooheelee@sungshin.ac.kr
‡ 교신저자, jiseungkim@jbnu.ac.kr(Corresponding author)

I. 서 론

1994년 Shor가 제안한 양자 알고리즘[1]에 의해 기존 RSA(Rivest-Shamir-Adelman), ECC (Elliptic-Curve-Cryptography)와 같은 표준 공개키 암호 알고리즘을 양자 컴퓨터를 이용하여 다항 시간 내에 해독할 수 있게 되었다. 양자 컴퓨팅 환경의 현실화 가능성이 커짐에 따라 양자 컴퓨팅 환경에서도 안전하면서 기존의 표준 공개키 암호를 대체할 수 있는 암호인 양자내성암호(Post-Quantum Cryptography, PQC)에 대한 연구가 활발히 진행되고 있다.

양자내성암호는 기반 난제의 종류에 따라 분류할 수 있는데, 대표적으로 격자(Lattice) 기반 암호, 코드(Code) 기반 암호, 다변수다항식(Multivariate polynomial) 기반 암호 등이 있다. 또한 알고리즘의 종류에 따라 공개키 암호화(Public-Key Encryption, PKE) 및 키 캡슐화 메커니즘(Key Encapsulation Mechanism, KEM), 그리고 전자서명(Signature)으로 분류한다. 미국 NIST에서는 2016년부터 양자내성암호 표준화 공모를 시작하였고, 2022년 표준 알고리즘으로 격자 기반 PKE/KEM인 CRYSTAL-Kyber (이하 Kyber), 격자 기반 전자서명인 CRYSTAL-Dilithium, Falcon, 해시(hash) 기반 전자서명인 SPHINCS+를 선정하였다[2]. 현재는 추가적인 표준 선정을 위한 4라운드와 전자서명에 대한 재공모(On ramp)를 진행 중이다. 국내에서도 2022년부터 KpqC 공모전을 개최하여 양자내성암호 연구 활성화를 독려하고, 국내 PQC 알고리즘 표준화 공모를 진행하고 있다. KpqC 공모전은 현재 1라운드 진행 중이며 총 16종의 PKE/KEM(7), 전자서명(9) 알고리즘이 표준 후보로 제출되었다[3].

본 논문에서는 KpqC 공모전 1 라운드의 격자 기반 KEM 3종 중 하나인 TiGER KEM에 대해 조합론적 공격의 일종인 Meet-LWE 공격[4]을 적용하여, 제안된 파라미터에 대한 구체적인 안전성 분석 결과를 제시한다. 특히, 기존 Meet-LWE 공격은 LWE의 비밀키와 에러가 삼진 벡터인 경우에 대한 공격으로 제안되었으나, 에러의 각 계수가 $[-B,B]$의 원소인 경우에 대해 공격 알고리즘을 확장하여 각 알고리즘에 대해 더욱 정확한 공격 복잡도를 도출하였다. Meet-LWE는 공격 복잡도를 계산하는 과정에서 별도의 가정이 사용되지 않기 때문에 타 격자 기반 공격에 비해 실질적인 공격량을 도출할 수 있다. Meet-LWE 공격 적용 결과 TiGER128, TiGER192, TiGER256에 대해 각각 184-bit, 170-bit, 298-bit classical 안전성을 도출하였다. **특히, TiGER192 파라미터의 경우 Meet-LWE 공격에 대해 170-bit의 classical 안전성을 가지며, TiGER 제안문서에서 달성 목표로 제시한 200-bit classical 안전성[1]을 달성하지 못함을 보였다**. 또한, 본 논문에서는 TiGER192 파라미터에 대해 Meet-LWE 적용 시 192-bit, 200-bit의 classical 안전성을 달성하기 위한 해밍웨이트(Hamming weight) 값을 각각 제시하였다. 본 논문이 시사하는 바는 다음과 같다.

1) 격자 기반 알고리즘의 파라미터 설정 시 대부분의 제안 문서에서 Lattice estimator[11]를 이용하여 안전성을 분석하고 있다. 그러나, Lattice estimator에서 다루는 공격의 종류가 제한적이기 때문에, Lattice estimator 내에 탑재되어 있는 공격뿐만 아니라 최신 공격을 고려한 파라미터를 도출해야 한다.

2) Sparse 비밀키나 ternary 비밀키를 사용하는 경우 특히 May의 Meet-LWE 공격을 고려할 필요가 있다. 공격을 수행할 때 에러의 각 계수가 $[-B,B]$의 원소인 경우에 대해 공격 알고리즘을 확장하여 적용하는 경우 정확한 공격량 계산이 가능하다.

본 논문은 다음과 같이 구성된다: II 장에서는 KEM의 정의와 TiGER KEM의 기반 난제인 LWE, LWR에 대해 소개한다. III 장에서는 TiGER의 주요 특징 및 알고리즘을 소개한다. IV 장에서는 Meet-LWE 공격의 아이디어와 공격복잡도에 대해 소개한다. V 장에서는 Meet-LWE 공격 적용시 스킴의 안전성을 분석하고 추가적인 논의사항을 제안한다. 마지막 VI 장에서는 결론을 서술한다.

II. 배 경

2.1 Key Encapsulation Mechanism

키 캡슐화 메커니즘(KEM)은 TLS/IPSec 등의 인터넷 프로토콜에서 대칭키 암호화에 사용될 비밀키(세션 키)를 공유하기 위한 공개키 암호 알고리즘이

1) 이는 192-bit의 양자 안전성에 대응된다.

다. KEM은 안전성 파라미터를 입력값으로 받아 공개키와 개인키 쌍을 생성하는 KeyGen, 공개키를 입력으로 받아 암호문과 비밀키 쌍을 생성하는 캡슐화(Encapsulation), 개인키와 암호문을 입력으로 하여 암호문을 복호화한 결과인 비밀키를 생성하는 디캡슐화(Decapsulation), 이렇게 3가지 알고리즘으로 구성된다 :

- KeyGen(1^{λ}): 키 생성 알고리즘 KeyGen은 보안 매개변수 λ를 입력으로 받아 공개키, 개인키 쌍(pk, sk)를 출력하는 무작위 알고리즘이다.
- Encaps(pk): 공개키 pk를 입력으로 받아 암호문 c와 공유키 K를 출력하는 무작위 알고리즘이다.
- Decaps(sk, c): 개인키 sk와 암호문 c를 입력으로 받아 공유키 K 또는 $\perp$를 출력한다.

2.2 기반 난제

본 절에서는 TiGER가 기반으로 두고 있는 격자 기반 난제 Learning With Errors(LWE)[5], Learning With Rounding (LWR)[6] 각각의 정의를 설명한다.

- Learning With Errors (LWE) 문제

주어진 비밀 벡터 $s \in \mathbb{Z}_q^n$에 대하여 $A_{m,n,q,\chi_e}^{LWE}(s)$분포를 다음과 같이 정의한다: 각 $i \in \{1,2,\cdots,n\}$에 대하여, 랜덤한 $a_i \in \mathbb{Z}_q^n$와 이산 가우시안 분포 χ_e에서 선택한 에러 $e_i \in \mathbb{Z}$를 선택하여 $b_i = \langle a_i, s \rangle + e_i \bmod q$를 생성하고, $\{(a_i, b_i)\}_{i=1}^m$를 결과로 내보낸다.

결정 LWE(decision-LWE) 문제는 $\{(a_i, b_i)\}_{i=1}^m$가 주어졌을 때, 이 순서쌍들의 집합이 $A_{m,n,q,\chi_e}^{LWE}(s)$를 따르는지, 랜덤하게 생성되었는지 결정하는 문제이다. 탐색 LWE(search-LWE) 문제는 분포 $A_{m,n,q,\chi_e}^{LWE}(s)$를 따라 생성된 $\{(a_i, b_i)\}_{i=1}^m$가 주어졌을 때, 비밀키 $s \in \mathbb{Z}_q^n$를 찾는 문제이다.

RLWE(Ring-LWE)는 n차 다항식 $f(x) \in \mathbb{Z}[x]$에 대해 환(ring) $R_q = \mathbb{Z}_q[x]/(f(x))$ 위에서 정의된 LWE 문제로 벡터와 환에서 정의된 다항식(polynomial)을 동일시한다. 주어진 비밀키 $s(x)$에 대하여, 분포 $A_{n,q,\chi_e}^{RLWE}(s(x))$를 다음과 같이 정의한다: 랜덤한 다항식 $a(x) \in R_q$와 각 계수들을 이산 가우시안 분포 χ_e에서 선택하여 만들어진 다항식 $e(x)$가 주어졌을 때, $b(x) = a(x)s(x) + e(x) \bmod R_q$를 계산하고, 순서쌍 $(a(x), b(x)) \in R_q^2$를 결과로 내보낸다. 결정 RLWE 문제는, $(a(x), b(x)) \in R_q^2$가 주어졌을 때, 이 순서쌍이 분포 $A_{n,q,\chi_e}^{RLWE}(s(x))$를 따르는지, 랜덤하게 생성되었는지 결정하는 문제이다. 탐색 RLWE 문제는 주어진 순서쌍 $(a(x), b(x)) \in R_q^2$이 분포 $A_{n,q,\chi_e}^{RLWE}(s(x))$를 따를 때, $s(x)$를 찾는 문제이다.

- Learning With Rounding (LWR) 문제

주어진 비밀 벡터 $s \in \mathbb{Z}_q^n$에 대하여 $A_{m,n,q,p}^{LWR}(s)$분포를 다음과 같이 정의한다: 각 $i \in \{1,2,\cdots,n\}$에 대하여, 랜덤한 $a_i \in \mathbb{Z}_q^n$와 modulus p, q를 이용하여 $b_i = \lfloor (p/q) \cdot (\langle a_i, s \rangle \bmod q) \rceil$를 계산하고, 순서쌍들의 집합 $\{(a_i, b_i)\}_{i=1}^m$를 결과로 내보낸다.

결정 LWR(decision-LWR) 문제는 $\{(a_i, b_i)\}_{i=1}^m$가 주어졌을 때, 이 순서쌍들의 분포가 $A_{m,n,q,p}^{LWR}(s)$를 따르는지, 랜덤하게 생성되었는지 결정하는 문제이다. 탐색 LWR(search-LWR) 문제는 분포 $A_{m,n,q,p}^{LWR}(s)$를 따라 생성된 $\{(a_i, b_i)\}_{i=1}^m$가 주어졌을 때, 비밀키 $s \in \mathbb{Z}_q^n$를 찾는 문제이다.

LWE 문제와 마찬가지로 LWR 문제도 Ring-LWR(RLWR)로 확장 가능하다.

III. TiGER KEM 알고리즘 소개

TiGER[3]는 RLWR과 RLWE 문제의 어려움을 기반으로 하여 설계된 IND-CPA PKE와, 이에 대해 Fujisaki-Okamoto(FO) 변환[7]을 적용한 결과인 IND-CCA 안전성을 만족하는 KEM으로 제안되었다. TiGER의 설계상 특징은 다음과 같다.

1) RLWR 인스턴스로 키 생성을, RLWE 인스턴스로 키 캡슐화 및 암호화를 한다. RLWR와 RLWE에 대한 비밀값은 정해진 해밍웨이트를 가지는 벡터로 정해진다. 또한 RLWE의 에러 역시 sparse한 벡터로 샘플링 한다.

2) RLWE(R)에 대한 modulus p, q를 2의 멱승으로 설정하여 Rounding을 비트 시프트 연산

(bitwise shift)으로 계산한다.

3) 다항식 간의 연산 시 NTT(Number Theoretic Transform)를 사용하지 않는다.

4) 다양한 보안 프로토콜로의 전환에 유리하도록 작은 대역폭을 만족하도록 하였다. 이를 위해 작은 크기의 modulus $q(=256)$를 사용한다는 특징이 있다.

5) 작은 q로 인해 증가하는 복호화 실패 확률에 대한 문제를 메시지 인코딩 시 오류정정부호(Error Correctiong Code, ECC)를 사용하여 해결하였다. ECC의 한 종류인 XEf[8] 또는 D2[9]를 사용하여 에러를 복구하는 방식으로 무시 가능한(negligible) 수준으로 복호화 오류율을 조절할 수 있다.

TiGER 파라미터는 다음과 같이 구성된다.

- λ: 안전성 파라미터
- $SHAKE256: \{0,1\}^* \times \mathbb{Z} \rightarrow \{0,1\}^*$, 입력 비트열과 양의 정수값(출력값 길이)을 받아 정해진 길이만큼 출력값을 생성하는 XOF(eXtendable Output Function)
- $HWT_n(\cdot)$: 해밍웨이트 h인 sparse 비밀키를 생성
- $\lfloor \cdot \rceil$: 다항식 계수에 대해서 가장 가까운 정수로 반올림하는 Rounding 연산
- $ECC.Ecd: \{0,1\}^d \rightarrow R_q$, 인코딩 시 사용되는 오류정정 코드
- $ECC.Dcd: R_q \rightarrow \{0,1\}^d$, 디코딩 시 사용되는 오류정정 코드
- H, G: 해시함수
- n: 2의 멱승, 다항식환 $R_q = \mathbb{Z}_q[X]/(X^n+1)$를 구성하기 위한 다항식의 차수 (LWE, LWR의 dimension)
- q: 2의 멱승, 다항식환 $R_q = \mathbb{Z}_q[X]/(X^n+1)$를 구성하기 위한 modulus (LWE, LWR의 modulus)
- p: 2의 멱승, 공개키 구성 시 Rounding 후 modulus (LWR의 Rounding modulus)
- k_1, k_2: 2의 멱승, 암호문 압축 후 modulus
- h_s: 양의 정수, 비밀키(s)의 해밍웨이트 (LWR의 비밀키의 해밍웨이트)
- h_r: 양의 정수, 키 캡슐화(Encaps) 시 임시 비밀키(ephemeral secret) r의 해밍웨이트 (LWE의 비밀키의 해밍웨이트)
- h_e: 양의 정수, 키 캡슐화(Encaps) 시 에러 벡터의 해밍웨이트 (LWE의 에러에 대한 해밍웨이트)
- d: 양의 정수, 키 캡슐화(Encaps) 시 랜덤 씨드의 비트 수
- f: 양의 정수, ECC를 사용하여 복구 가능한 에러의 비트 수

TiGER의 IND-CCA KEM은 다음과 같이 구성된다.

TiGER.KeyGen

Input: security parameter 1^λ
Output: pk, sk
1) $Seed_a \leftarrow \{0,1\}^{256}$
2) $Seed_s \leftarrow \{0,1\}^{256}$
3) $a \leftarrow SHAKE256(Seed_a, n/8) \in R_q$
4) $s \leftarrow HWT_n(h_s, Seed_s)$
5) $b \leftarrow \lfloor (p/q) \cdot a * s \rceil$
6) $u \leftarrow R_2$
return $pk \leftarrow (Seed_a \parallel b)$, $sk \leftarrow (s \parallel u)$

TiGER.Encaps

Input: pk
Output: c, K
1) $m \leftarrow \{0,1\}^d$
2) $r \leftarrow HWT_n(h_r, H(m))$
3) $Seed_{e_1} \leftarrow (H(m) + Nonce)$,
$Seed_{e_2} \leftarrow (H(m) + Nonce + 1)$
4) $e_1 \leftarrow HWT_n(h_e, Seed_{e_1})$,
$e_2 \leftarrow HWT_n(h_e, Seed_{e_2})$
5) $pk := (Seed_a, b)$,
$a \leftarrow SHAKE256(Seed_a, n/8)$
6) $c_1 \leftarrow \lfloor (k_1/q) \cdot (a * r + e_1 \bmod q) \rceil$,
$c_2 \leftarrow \left\lfloor (k_2/q) \cdot \begin{pmatrix} (q/2) \cdot ECC.Ecd(m) \\ +((q/p) \cdot b) * r + e_2 \end{pmatrix} \right\rceil$
7) $c \leftarrow (c_1, c_2) \in R_{k_1} \times R_{k_2}$
return c, $K \leftarrow G(c, m)$

TiGER.Decaps

Input: sk, c
Output: K
1) $sk := (s,u)$, $c := (c_1, c_2)$
2) $\widehat{m'} \leftarrow \left\lfloor (2/q) \cdot \begin{pmatrix} (q/k_2) \cdot c_2 \\ -((q/k_1) \cdot c_1) * s \end{pmatrix} \right\rceil$
3) $m' \leftarrow ECC.Dcd(\widehat{m'})$
4) $r' \leftarrow HWT_n(h_r, H(m'))$
5) $Seed'_{e_1} \leftarrow (H(m') + Nonce)$,
$Seed'_{e_2} \leftarrow (H(m') + Nonce + 1)$
6) $e'_1 \leftarrow HWT_n(h_e, Seed'_{e_1})$,
$e'_2 \leftarrow HWT_n(h_e, Seed'_{e_2})$
7) $pk := (seed_a, b)$,
$a \leftarrow SHAKE256(Seed_a, n/8)$
8) $c'_1 \leftarrow \lfloor (k_1/q) \cdot (a * r' + e'_1 \bmod q) \rceil$,
$c'_2 \leftarrow \left\lfloor k_2/q \begin{pmatrix} (q/2) \cdot ECC.Ecd(m') \\ +((q/p) \cdot b) * r' + e'_2 \end{pmatrix} \right\rceil$
9) $c' = (c'_1, c'_2) \in R_{k_1} \times R_{k_2}$
if $c = c'$ **then return** $K \leftarrow G(c', m')$ **else return** $K \leftarrow G(c, u)$

IV. Meet-LWE 공격 기법

이 절에서는 2021년 Alexander May가 제안한 조합론적 공격인 Meet-LWE 공격 알고리즘[10]을 간략히 소개하고, 공격 복잡도(Complexity)에 관해 다룬다.

4.1 기호

LWE 인스턴스 $(A, b = As + e \bmod q)$가 주어져 있다고 하자. $A \in \mathbb{Z}_q^{n \times m}$이고, $s \in \mathbb{Z}^n$와 $e \in \mathbb{Z}^m$는 ternary 벡터라고 가정한다. n차원 ternary 벡터들의 집합을 T^n으로 정의한다. 추가적으로 s의 해밍웨이트는 h로 고정하며, s의 1의 원소 개수와 -1의 원소 개수를 동일하게 설정한다. 위 성질을 만족하는 비밀키 s의 집합을 $T^n(h)$으로 표기한다.

4.2 공격 아이디어

LWE문제를 다음과 같은 리스트(list)를 만드는 것으로 생각할 수 있다.

$$L = \{s \in T^n(h) : b - As \bmod q \in T^n\}$$

Meet-LWE는 $s \in T^n(h)$의 다양한 표현(representation)에서 시작한다. Meet-LWE 공격은 s를 $s = s_1 + s_2$로 분할할 때, s_1, s_2가 $T^n(h/2)$의 원소가 되도록 하는 표현 방식이 다양할 것이라는 관찰에서 시작한다.[2] 이 관찰을 통해, $b = As + e \bmod q$의 정의에 따라 다음을 관찰할 수 있다.

$$b - As_2 + e = As_1 \bmod q \quad (1)$$

e를 분해하여 $e_1 \in T^{n/2} \times 0^{n/2}, e_2 \in 0^{n/2} \times T^{n/2}$로 정의한다면, 식 (1)을 다음처럼 이해할 수 있다.

$$b - As_2 + e_2 = As_1 + e_1 \bmod q \quad (2)$$

식 (2)을 통해 e가 ternary 벡터이기 때문에, As_1과 $b - As_2$의 각 성분의 차이가 최대 1이라는 사실을 관찰할 수 있다.

Odlyzko가 제안한 LSH(Locality Sensitive Hashing) 방법을 활용하여 s_1과 s_2를 찾고 이를 통해 비밀키 s를 복구할 수 있다. LSH로 사용할 함수 $\ell : \mathbb{Z}_q^n \rightarrow \{0,1\}^n$은 다음처럼 정의한다:

$$\ell(x)_i = \begin{cases} 0, & 0 \le x_i < \lfloor q/2 \rfloor - 1 \\ 1, & \lfloor q/2 \rfloor \le x_i < q - 1 \end{cases}$$

정의되지 않은 경계값(boarder values) $\lfloor q/2 \rfloor - 1$과 $q-1$에는 $\ell(x)_i$에 0과 1 모두 값을 할당하면, 다음 등식들이 높은 확률로 성공함을 기대할 수 있다.

$$\ell(As_1) = \ell(As_1 + e_1)$$
$$\ell(b - As_2 + e_2) = \ell(b - As_2)$$

따라서, $\ell(As_1) = \ell(b - As_2)$가 성립하는 s_1과 s_2를 찾고, 최종적으로 $b - (As_1 + As_2) \bmod q$가 ternary 벡터가 됨을 확인함으로써 비밀키 $s = s_1 + s_2$의 후보를 찾을 수 있다. 두 개의 리스트

2) $T^n(w), w \ge h/2$이거나, s_1, s_2가 ternary 벡터가 아닌 경우도 있으나, 편의상 해밍웨이트가 $h/2$인 경우만을 고려한다.

$$L'_1 = \{(s_1, \ell(As_1) : \ell(As_1) = \ell(As_1 + e_1)\}$$
$$L'_2 = \{(s_2, \ell(b - As_2) : \ell(b - As_2 + e_2) = \ell(b - As_2)\}$$

를 만들어 $\ell(As_1) = \ell(b - As_2)$, $s_1 + s_2 \in T^n(h)$을 만족하는 벡터 (s_1, s_2)를 찾으면 공격이 유의미한 확률로 성공하게 된다. 이 알고리즘의 경우 Meet-in-the-middle로 불리는 알고리즘의 형태로, 두 개의 리스트를 만들고, 리스트 사이의 충돌쌍(collision)을 만들어 비밀키 s를 찾는 방법이다. 두 개의 리스트를 넘어 리스트의 트리(Tree)로 확장시킨 것이 Meet-LWE의 기본 개념이다.

트리 구조로 확장시키기 위해 쓰이는 주요 아이디어는 정사영(projection)이다. s를 s_1, s_2로 표현할 수 있는 방법의 경우의 수를 R이라고 했을 때, $r = \lfloor log_q R \rfloor$라고 하자. 정사영 함수 $\pi_r : \mathbb{Z}_q^n \to \mathbb{Z}_q^r$을 다음과 같이 정의하고,

$$x = (x_1, \ldots, x_n) \mapsto \pi_r(x) = (x_1, \ldots, x_r)$$

π_r을 이용하여 다음 리스트들을 생각한다.

$$L_1(e_1) = \{(s_1, \ell(As_1)) : \pi_r(As_1 + e_1) = 0 \mod q\}$$
$$L_2(e_2) = \{(s_1, \ell(b - As_2)) : \pi_r(b - As_2 + e_2) = 0 \mod q\}$$

이 때, π_r의 치역의 원소의 개수는 q^r이고, s의 표현방법의 가짓수를 $R = q^{\log_q R}$로 표기한다면, $q^r < R$ 이므로 $\pi_r(As_1 + e_1) = \pi_r(As_1) + \pi_r(e_1) = 0 \mod q$가 성립하는 s_1이 항상 존재하는 것으로 기대할 수 있다. 또한, 식 (2)와 정사영 π_r의 성질을 활용하면 $\pi_r(b - As_2 + e_2) = 0$이 자동적으로 성립한다는 것 또한 관찰할 수 있다. 한편, $\pi_r(e_1)$과 $\pi_r(e_2)$는 비밀값이기 때문에, 가능한 모든 $\pi_r(e_1)$과 $\pi_r(e_2)$을 추측하는 방법을 사용한다. 이때, 전수조사에 소요되는 시간은 $O(3^{r/2})$이며, r을 작은 값으로 설정한다.

정리하자면, Meet-LWE의 기본적인 아이디어는 트리구조로 확장할 수 있는 리스트를 만들고, 리스트를 통해 s_1과 s_2를 복구하는 것으로 이해할 수 있다.

$$L_1 = \cup_{\pi(e_1)} L_1(e_1)$$
$$L_2 = \cup_{\pi(e_2)} L_2(e_2)$$

4.3 공격 알고리즘

본 절에서는 이전 절에서 언급한 Meet-LWE 공격 아이디어를 더 구체화하여 공격 알고리즘을 설명한다. 주로, 정사영을 이용하여 트리구조를 갖는 리스트를 설계하는 방법을 논의한다. 알고리즘의 정확성을 위한 자세한 파라미터 설정은 언급하지 않는다. 성능의 최적화를 위한 다양한 요소 또한 생략한다.

서술의 용이함을 위해 $L^{(i)}$를 레벨-i 리스트라고 하자. Meet-LWE의 공격 아이디어는, LWE 문제를 해결하는 것을 레벨-0의 리스트로 이해하고, 레벨-0의 리스트를 얻기 위해 레벨-1 리스트 두 개를 만드는 것으로 이해할 수 있다. 귀납적으로 생각하면 레벨-i 리스트 하나를 만들기 위해, 레벨-$(i+1)$ 리스트 두 개가 필요하다. 즉, 레벨-0의 리스트를 만들기 위해, 레벨-2 리스트 네 개가 필요하다. 리스트의 최고 레벨을 t라고 한다면, 레벨-0의 리스트를 만들기 위해 2^t개의 레벨-t 리스트가 필요하다.

편의상 $t = 3$이라고 하고, 각 레벨 리스트들의 생김새를 살핀 후, 알고리즘을 설명한다. 레벨-2 리스트를 설명하기 위해, 레벨-0, 레벨-1 리스트를 다음과 같이 정의한다.[3]

$$L^{(0)} = \{s^{(0)} \in T^n(h) : b - As^{(0)} \mod q \in T^n\}$$
$$L_1^{(1)} = \{(s_1^{(1)}, \ell(As_1^{(1)})) : \pi_{r^{(1)}}(As_1^{(1)} + e_1) = 0 \mod q\}$$
$$L_2^{(1)} = \{(s_2^{(1)}, \ell(b - As_2^{(1)})) : \pi_{r^{(1)}}(b - As_2^{(1)} + e_2) = 0 \mod q\}$$

이때, $s_1^{(1)}, s_2^{(1)} \in T^n(h/2)$가 된다. 만약, $L_1^{(1)}$을 설계하기 위해 레벨-2 리스트 $L_1^{(2)}, L_2^{(2)}$가 필요했다면, 레벨-2 리스트 원소는 동시에 $T^n(h/4)$의 원소이다.

구체적으로 레벨-2 리스트는 다음 과정을 통해 설계된다. 먼저, $s_1^{(1)} = s_1^{(2)} + s_2^{(2)}, s_2^{(1)} = s_3^{(2)} + s_4^{(2)}$를 만족하는 $T^n(h/4)$의 원소 $s_1^{(2)}, s_2^{(2)}, s_3^{(2)}, s_4^{(2)}$를 식 (2)에 대입하면, 다음 식을 얻는다.

$$b - (s_3^{(2)} + s_4^{(2)}) + e_2 = A(s_1^{(2)} + s_2^{(2)}) + e_1 \mod q$$

위 식을 통해 레벨-2 리스트는 다음 네 종류로 구성된다.[4]

3) $L^{(1)}(e_j)$이나, 앞서 언급한 것처럼 서술의 용이함을 위해 생략하였다. 고정된 $\pi_r(e_i)$에 관하여 알고리즘을 수행하는 것으로 이해하고, $\pi_r(e_i)$를 변경하면서 알고리즘을 $O(3^{r/2})$번 반복하는 것으로 생각해도 된다.

$$L_1^{(2)}=\left\{(s_1^{(2)},As_1^{(2)}):\pi_{r^{(2)}}(As_1)=t \bmod q\right\}$$
$$L_2^{(2)}=\left\{(s_2^{(2)},As_2^{(2)}):\pi_{r^{(2)}}(As_2+e_1)=-t \bmod q\right\}$$
$$L_3^{(2)}=\left\{(s_3^{(2)},b-As_3^{(2)}):\pi_{r^{(2)}}(b-As_3^{(2)})=t' \bmod q\right\}$$
$$L_4^{(2)}=\left\{(s_4^{(2)},As_4^{(2)}):\pi_{r^{(2)}}(As_4^{(2)}+e_2)=-t' \bmod q\right\}$$

이때, t,t'는 임의의 벡터이고, $r^{(2)}<r^{(1)}$을 만족해야한다. 파라미터 $r^{(2)}$는 레벨-1 리스트의 임의의 원소 $s^{(1)}$를 표현(representation)하는 방법의 경우의 수에 의존한다. 표현 경우의 수 $R^{(2)}$라고 하면, $r^{(2)}=\lfloor \log_q R^{(2)} \rfloor$ 이다.

한편, 최고 레벨인 레벨-3리스트는 기존과 다르게 설정한다. 임의의 $i\in\{1,2,3,4\}$에 대하여, 다음과 같이 정의한다.

$$L_{2i-1}^{(3)}=\left\{s_{2i-1}^{(3)}\in T^{n/2}(h/8)\times 0^{n/2}\right\}$$
$$L_{2i}^{(3)}=\left\{s_{2i}^{(3)}\in 0^{n/2}\times T^{n/2}(h/8)\right\}$$

이때, 레벨-3리스트는 전수조사하여 직접 만든다. 이 시간은 대략 $\binom{n/2}{h/8}$정도 소요되고, h/8이 작은 경우에 레벨-3리스트를 만들 수 있다.[5] 공격 알고리즘은 리스트를 설명했던 방식의 역순으로 진행된다.

1) 레벨-3 리스트 $L_i^{(3)}$를 만든다. ($i\in\{1,...,8\}$)

2) $L_{2i-1}^{(3)}$과 $L_{2i}^{(3)}$를 이용해 레벨-2 리스트 $L_j^{(2)}$를 만든다. ($j\in\{1,2,3,4\}$)

3) $L_{2j-1}^{(2)},L_{2j}^{(2)}$를 이용해 레벨-1 리스트 $L_1^{(1)},L_2^{(1)}$를 만든다.

4) $L_1^{(1)},L_2^{(1)}$를 이용해 비밀키 s를 찾는다.

4.4 공격 복잡도

Meet-LWE의 복잡도는 1) $\pi_r(e_1),\pi_r(e_2)$를 전수조사 하는 데 소요되는 시간, 2) 리스트를 만드는데 걸리는 시간 크게 두 부분으로 구성되어 있다. e_1과 e_2의 구조적 특성 때문에, $\pi_r(e_1),\pi_r(e_2)$를 전수조사하기 위한 공격 복잡도는 에러가 ternary 벡터인 경우 3^r로 고정되어 있으며, 만약 e_i의 계수가 삼진벡터가 아닌 $[-B,B]$의 원소라면, 복잡도가 $(2B+1)^r$이 된다. 리스트를 만드는데 소요되는 복잡도는 아래와 같다. 자세한 설명은 원논문[10]을 참조하면 된다. 먼저 레벨-i의 리스트 크기를 $|L^{(i)}|$로 표기하고, 만드는데 걸리는 소요시간(cost)를 $T^{(i)}$로 나타내자.

1) 레벨-3 리스트 소요시간 $T^{(3)}=\sqrt{S^{(2)}}$:

MitM(Meet-in-the-Middle)에 의해 성립한다. 이때, $S^{(2)}$는 $s_i^{(2)}$의 키 공간(key space)의 크기다.

2) 레벨-2 리스트 소요시간 $T^{(2)}=\dfrac{|L^{(3)}|^2}{q^{r^{(2)}}}$:

$\pi_{r^{(2)}}$의 정의와 레벨-2 리스트의 정의에 따라 $r^{(2)}$개의 좌표에서는 등식이 성립해야 한다. 각 좌표가 $\mathbb{Z}_q$위에서 정의되었기 때문에, $L_i^{(2)}$를 만드는데 소요되는 시간의 기대값은 $\dfrac{|L^{(3)}|^2}{q^{r^{(2)}}}$이다.

3) 레벨-1 리스트 소요시간 $T^{(1)}=\dfrac{|L^{(2)}|^2}{q^{r^{(1)}-r^{(2)}}}$:

$\pi_{r^{(1)}}$의 정의와 레벨-1 리스트의 정의에 따라 $r^{(1)}$개의 좌표에서는 항상 등식이 성립해야 한다. 추가적으로 레벨-1 리스트는 레벨-2 리스트들로부터 설계되기 때문에, $r^{(2)}$개의 좌표에서는 이미 등식이 성립하고 있는 상황이다. 따라서 남은 $r^{(1)}-r^{(2)}$개의 좌표에서 등식이 성립함을 확인하면 된다. $L_i^{(1)}$를 만드는데 소요되는 시간의 기대값은 $\dfrac{|L^{(2)}|^2}{q^{r^{(1)}-r^{(2)}}}$이다.

4) 레벨-0 리스트 소요시간 $T^{(0)}=\dfrac{|L^{(1)}|^2}{2^{n-r^{(1)}}}$:

레벨-1 리스트의 정의에 따라 $r^{(1)}$개의 좌표에서는 이미 등식이 성립한 상황이므로 남은 $n-r^{(1)}$개의 좌표에서 등식이 성립함을 보이면 충분하다. 기존의 레벨-($i\geq 1$) 리스트들과 달리, LSH를 사용하기 때문에 해시값의 각 좌표는 0 또는 1이다. 따라서 $L^{(0)}$를 만드는데 소요되는 시간의 기대값은 $\dfrac{|L^{(1)}|^2}{2^{n-r^{(1)}}}$이다.

따라서, Meet-LWE 알고리즘의 총소요시간은

4) 레벨-1리스트와 달리 LSH를 사용하지 않는다.

5) $\binom{n/2}{h/8}$가 전수조사 할 수 없을 정도로 큰 숫자라면, 레벨-4 혹은 더 높은 레벨을 최고레벨로 설정한다. 만약 최고레벨이 t라면, $\binom{n/2}{h/2^t}$의 시간이 소요될 것이다. t를 적절히 선택한다면 $\binom{n/2}{h/2^t}$시간을 들여 리스트를 만들 수 있다는 것이 무리한 가정이 아니다.

$3^r \max(T^{(0)}, T^{(1)}, T^{(2)}, T^{(3)})$이 되며, 만약 e_i의 계수가 삼진벡터가 아닌 $[-B,B]$의 원소인 경우에는 $(2B+1)^r \max(T^{(0)}, T^{(1)}, T^{(2)}, T^{(3)})$이 된다.

4.5 공격 최적화-다양한 표현방법

4.2절과 4.3절에서 표현의 정의를 $s \in T^n(h)$를 $s = s_1 + s_2$로 나눌 때, s_1, s_2가 $T^n(h/2)$의 원소가 되는 s_1, s_2의 방식으로 정의하였다. 원 논문[10]에서는 세 가지의 표현 방법을 사용한다. 4.2와 4.3절에서 사용했던 표현방법을 Rep-0이라고 부르고, 이를 확장한 방법론으로 Rep-1, Rep-2를 제안하였다. 구체적인 Rep-1과 Rep-2정의는 다음과 같다:

Rep-1: $s^{(i)} \in T^n(h^{(i)})$, $s^{(i)} = s_1^{(i+1)} + s_2^{(i+1)}$이면서 $s_1^{(i+1)}, s_2^{(i+1)} \in T^n(h^{(i+1)})$, $h^{(i+1)} > h^{(i)}/2$ 표현

Rep-2: $s^{(i)} \in T^n(h^{(i)})$, $s^{(i)} = s_1^{(i+1)} + s_2^{(i+1)}$이면서 $s_1^{(i+1)}, s_2^{(i+1)} \in \{-2,1,0,1,2\}^n$ 표현

각 표현 방법에 따라 표현 경우의 수 $R^{(i)}$가 바뀌며, 그에 맞춰서 정사영 파라미터 $r^{(i)}$의 값이 바뀌게 된다. 실험적으로 Rep-2 표현을 사용했을 때, 알고리즘의 효과가 가장 좋은 것으로 알려져 있다.

V. TiGER KEM 안전성 분석 결과

본 절에서는 TiGER KEM에 대한 파라미터와 안전성 목표를 살펴보고, Meet-LWE 공격 적용 시의 안전성 분석 내용을 다룬다.

5.1 TiGER 파라미터와 안전성 목표

TiGER는 NIST 안전성 레벨 (Security Level) 1, 3, 5에 대응하는 파라미터인 TiGER128, TiGER192, TiGER256을 제안하였으며, 각각은 AES128, AES192, AES256과 동등한 수준의 안전성을 기준으로 하여 선별되었다. 각 안전성 레벨에 따른 구체적인 파라미터 수치는 Table 1. 과 같다.

또한, TiGER 제안서 5.4절에서는 각각에 대한 공격 시 TiGER128은 120-bit, TiGER 192는 192-bit, TiGER256은 246-bit 이상의 양자 안전성을 만족할 것으로 보고하였다. 이러한 예측은 BKZ 알고리즘[9]의 복잡도를 해당 알고리즘의 부분 알고리즘 중 'SVP oracle'에 대한 복잡도로 환원하는 Core-SVP 모델에 기반한다.

Table 1. Recommended Parameters of TiGER

parameters	n	q	p	k_1	k_2	h_s	h_r	h_e	d	f
TiGER 128	512	256	128	64	64	160	128	32	128	3
TiGER 192	1024	256	64	64	4	84	84	84	256	5
TiGER 256	1024	256	128	128	4	198	198	32	256	5

5.2 안전성 분석

본 절에서는 TiGER KEM에 대해 Meet-LWE 적용 시의 공격량을 분석한다. TiGER KEM의 안전성은 sparse한 비밀키를 사용하는 LWE와 LWR에 기반한다. 또한, 키 캡슐화 과정에서 LWE 인스턴스에 대해 추가적인 Rounding을 적용하여 암호문을 압축했기 때문에, 분석의 입장에서는 LWE 에러와 modulus k_2로의 Rounding 에러의 합산을 고려해야 한다. 이러한 관점에서 TiGER192, TiGER256에 대해서는 $h_s = h_r$, $q > p > k_2$를 만족하기 때문에 키 생성에서 사용한 LWR 인스턴스에 대한 공격량이 키 캡슐화에서 사용한 LWE 인스턴스에 대한 공격량보다 더 작을 것이라고 유추할 수 있다. 따라서, TiGER192, TiGER256에 대한 안전성 분석에서는 키 생성에서 사용한 LWR 인스턴스에 대한 공격량을 집중적으로 분석하였다. 또한, TiGER128 파라미터에 대해서는 LWE와 LWR 인스턴스 각각에 대한 공격량 중 작은 값을 표기하였다. 또한, LWE/LWR 에러의 크기에 따라 Meet-LWE의 알고리즘을 변형 및 확장하여 분석하였다. 아래 Table 2. 와 Table 3. 은 각각

Table 2. Estimated log time complexity ($\log_2(time\ complexity)$) of Meet-LWE attack for the paramter sets TiGER128, TiGER192, TiGER256.

	Rep-0	Rep-1	Rep-2
TiGER128	225	175	**167**
TiGER192	220	194	**170**
TiGER256	387	309	**298**

Table 3. Estimated log memory complexity of Meet-LWE attack for the paramter sets TiGER128, TiGER192, TiGER256.

	Rep-0	Rep-1	Rep-2
TiGER128	213	150	**142**
TiGER192	210	165	**141**
TiGER256	370	269	**258**

Meet-LWE 공격에 대한 시간복잡도와 공간복잡도의 로그값을 나타낸 결과(소수점 아래 첫째 자리에서 반올림함)이다.

5.3 Discussion

5.2절에서 TiGER KEM에 대한 Meet-LWE 공격 적용 시의 공격량을 분석한 결과, Fig.1. 과 같이 TiGER128, TiGER192, TiGER256 파라미터에 대해 각각 167-bit, 170-bit, 298-bit의 classical 안전성을 갖는다는 것을 확인하였다. **특히, TiGER192의 경우 Rep-2를 이용하는 Meet-LWE에 대해 170-bit 안전성을 갖기 때문에 TiGER 제안문서에서 주장한 192-bit의 양자안전성을 만족하지 못한다는 것을 확인할 수 있었다.**

Meet-LWE 공격에 대한 안전성을 보장하기 위해 가장 유효하고 효율적인 방법 중 하나는 LWE/R 비밀키의 해밍웨이트를 늘리는 것이다. 계산 결과, **TiGER192 파라미터에 대해 비밀키의 해밍웨이트를 100 이상(resp. 104 이상)으로 설정하면 Meet-LWE 공격에 대해 192-bit(resp. 200-bit)의 classical 안전성을 만족**한다는 것을 확인하였다. 비밀키의 해밍웨이트 범위 84~110일 때 TiGER 파라미터에 대한 Meet-LWE 공격의 시간복잡도를 Fig.2. 에 표시하였다.

```
****************
 Meet-LWE Rep-0
****************
TiGER128: time= 225.3 = 213.4 + 11.9, memory= 213.4
TiGER192: time= 220.3 = 209.9 + 10.4, memory= 209.9
TiGER256: time= 387.7 = 369.5 + 18.2, memory= 369.5
****************
 Meet-LWE Rep-1
****************
TiGER128: time= 175.5 = 150.2 + 25.4, memory= 150.2 with (11, 2, 1)
TiGER192: time= 194.0 = 164.9 + 29.0, memory= 164.9 with (8, 1, 1)
TiGER256: time= 309.5 = 269.1 + 40.4, memory= 269.1 with (16, 1, 1)
****************
 Meet-LWE Rep-2
****************
TiGER128: time= 167.0 = 141.7 + 25.4, memory= 141.7 with (11, 0, 2, 0)
TiGER192: time= 170.1 = 141.1 + 29.0, memory= 141.1 with (8, 0, 1, 0)
TiGER256: time= 298.5 = 258.1 + 40.4, memory= 258.1 with (16, 0, 3, 0)
```

Fig. 1. Test Results for the Python code to compute Meet-LWE attack time and memory complexities for Rep-0, Rep-1, Rep-2, respectively.

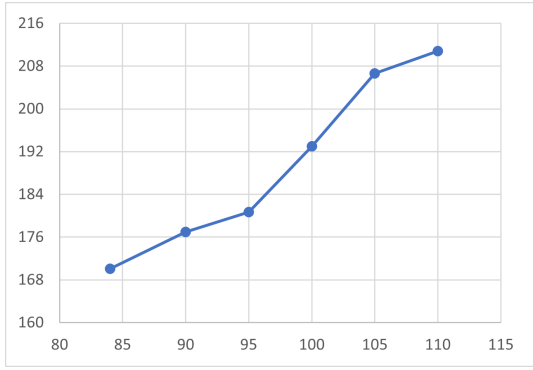

Fig. 2. Meet-LWE Time Complexity for the LWR Instance in TiGER when increasing h_s

VI. 결 론

본 논문에서는 KpqC 1 라운드 TiGER KEM에 대해 Meet-LWE 공격 적용 시의 분석 결과를 다루었다. 특히, TiGER192 파라미터에 대해 Meet-LWE 공격 적용 시 170-bit의 classical 안전성을 달성하기 때문에 목표하는 안전성을 만족하지 못함을 보였다.

〔Appendix〕

본 부록에서는 타 격자 기반 KEM에 대한 Meet-LWE 공격 적용 시 공격복잡도를 분석한다. KpqC 1 라운드 격자 기반 KEM 3종 중 SMAUG, NTRU+ 파라미터에 Meet-LWE 공격 적용 시 시간복잡도의 로그값은 각각 Table 4. , Table 5. 와 같다.

Table 4. Estimated log time complexity ($\log_2(complexity)$) of Meet-LWE attack for the paramter sets SMAUG128, SMAUG192, SMAUG256.

	Rep-0	Rep-1	Rep-2
SMAUG128	231	183	**176**
SMAUG192	289	233	**228**
SMAUG256	342	287	**277**

Table 5. Estimated log time complexity ($\log_2(complexity)$) of Meet-LWE attack for the paramter sets NTRU+576, NTRU+768, NTRU+864, NTRU+1152.

	Rep-0	Rep-1	Rep-2
NTRU+576	334	258	**250**
NTRU+768	446	347	**332**
NTRU+864	502	395	**372**
NTRU+1152	670	549	**493**

References

[1] P.W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," Proceedings 35th annual symposium on foundations of computer science. IEEE, pp. 124-134, Nov. 1994.

[2] J.Y. Park, Y.H. Moon, W. Lee, S.H. Kim and K. Sakurai, "A survey of polynomial multiplication with RSA-ECC coprocessors and implementations of NIST PQC round3 kem algorithms in Exynos2100," in IEEE Access, vol. 10, pp. 2546-2563, Dec. 2021.

[3] KpqC Competition Algorithms, (10/5, 2023), https://kpqc.or.kr/competition.html

[4] J. Buchmann, et al. "On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack," International Conference on Cryptology in Africa (AFRICACRYPT 2016), LNCS 9646, pp. 24-43, Apr. 2016.

[5] V. Lyubashevsky, C. Peikert, and O. Regev. "On ideal lattices and learning with errors over rings," Advances in Cryptology, EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, LNCS 6110, pp. 1-23, May. 30 - Jun. 3, 2010.

[6] F. Luo, F. Wang, K. Wang and K. Chen, "Fully homomorphic encryption based on the ring learning with rounding problem," IET Information Security, vol. 13, no. 6, pp. 639-648, Nov. 2019.

[7] D. Hofheinz, K. Hövelmanns, and E. Kiltz. "A modular analysis of the Fujisaki-Okamoto transformation," Theory of Cryptography Conference (TCC 2017), LNCS 10677, pp 341-371, Nov. 2017.

[8] H. Baan, S. Bhattacharya, S. Fluhrer, O. Garcia-Morchon, T. Laarhoven, R. Rietman, M.O. Saarinen, L. Tolhuizen and Z. Zhang, "Round5: Compact and Fast Post-quantum Public-Key Encryption," Post-Quantum Cryptography (PQCrypto 2019), LNCS 11505, pp. 83-102, Jul. 2019.

[9] E. Alkim, L. Ducas, T. Poppelmann and P. Schwabe, "Post-quantum key Exchange－A new hope," 25th USENIX Security Symposium (USENIX Security 16). pp. 327 - 343, Aug. 2016.

[10] A. May, "How to meet ternary LWE keys" Proceedings of the 2021 Advances in Cryptology-CRYPTO 2021, LNCS 12826, pp.701-731, Aug. 2021.

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

[12] K. Boudgoust, C. Jeudy, A. Roux-Langlois and W. Wen. "On the hardness of module-LWE with binary secret," Cryptographers' Track at the RSA Conference, pp. 503-526, May 2021.

[13] KpqC Bulletin Board, (10/5, 2023),

https://groups.google.com/g/kpqc-bulletin
[14] S. Park, C. Jung, A. Park, J. Choi, and H. Kang. "TiGER: Tiny bandwidth key encapsulation mechanism for easy miGration based on RLWE(R)," IACR ePrint 2022-1651, Nov. 2022.

〈저자소개〉

이 주 희 (Joohee Lee) 정회원
2013년 2월: 고려대학교 수학교육과 졸업
2019년 8월: 서울대학교 수리과학부 박사 (암호학 전공)
2019년 8월~2022년 2월: 삼성SDS 연구소 보안알고리즘Lab Senior Engineer
2022년 3월~현재: 성신여자대학교 융합보안공학과 교수
〈관심분야〉 암호학, 양자내성암호, 프라이버시 보호

이 은 민 (Eun-min Lee) 학생회원
2021년 3월~현재: 성신여자대학교 융합보안공학과 학사과정
〈관심분야〉 정보보호, 암호응용, 머신러닝

김 지 승 (Jiseung Kim) 정회원
2013년 2월: 전남대학교 수학과 졸업
2020년 2월: 서울대학교 수리과학부 박사 (암호학 전공)
2020년 3월~2022년 2월: 고등과학원 계산과학부 Research Fellow
2022년 3월~현재: 전북대학교 컴퓨터인공지능학부 교수
〈관심분야〉 암호학, 격자기반암호, 난제분석