정 보 보 호 학 회 지
제 33 권 제3호, 2023. 6

KpqC 공모전 1 라운드 격자 기반 PKE/KEM 알고리즘 분석

이 주 희*

요 약

양자컴퓨팅 기술이 발전함에 따라, 양자컴퓨터를 이용한 공격에도 안전한 암호인 양자내성암호(Post-Quantum Cryptography, PQC) 기술의 중요성이 대두되고 있다. NIST에서는 2016년부터 시작된 표준화 공모 1,2,3 라운드를 통해 2022년 공개키 암호 및 Key-establishment, 전자서명 분야의 양자내성암호 표준을 선정한 바 있으며, 현재는 4 라운드와 전자서명 분야 추가 선정 공모를 진행 중이다. 이러한 배경에서 2022년 국내에서도 양자내성암호 알고리즘 표준화 공모인 KpqC 공모전 1라운드를 시작하였고, 공개키 암호 및 Key-establishment 7종, 전자서명 9종의 알고리즘이 표준 후보로 제출되었다. 본고에서는 KpqC 공모전 1 라운드 공개키 암호 및 Key-establishment 알고리즘 중 격자 기반 공개키 암호/KEM(Key Encapsulation Mechanism) 알고리즘 3종 NTRU+, SMAUG, TiGER에 대해 분석 및 소개한다. 각 알고리즘의 기반 문제, 설계 방식, 특징, 안전성 분석 방식 등을 분석하고, 구현성능을 비교 분석한다.

Ⅰ. 서 론

Peter Shor[1]가 소인수분해 문제와 이산로그 문제를 풀기 위한 다항 시간 양자 알고리즘을 제안하면서, RSA, ECC와 같은 표준 공개키 암호의 안전성이 위협받게 되었다. 또한, 최근 양자컴퓨터 개발에 가시적인 성과가 드러나면서 양자컴퓨터의 발명 이후에도 안전한 암호인 양자내성암호의 연구 및 표준화 공모가 국내외로 활발하게 진행되고 있다.

양자내성암호는 기반 난제의 종류에 따라 격자(lattice) 기반, 코드 기반, 다변수다항식 기반, 대칭키 암호 기반 등으로 분류하며, 암호 알고리즘의 기능에 따라 공개키 암호(Public-Key Encryption, PKE) 및 Key Encapsulation Mechanism (KEM), 전자서명으로 분류한다.

본 고에서는 국내 양자내성암호 표준화 공모인 KpqC 공모전 1라운드 알고리즘[2] 중 격자 기반 PKE 및 KEM으로 제안된 NTRU+, SMAUG, TiGER에 대해 기반 문제, 설계 방식, 안전성 분석 방식, 대략적인 성능 등을 분석 및 소개하고자 한다.

본 논문은 다음과 같이 구성된다: II 장에서는 NTT, 격자 기반 난제, 선행 연구 등 기반 지식에 대해 소개한다. III 장에서는 본격적으로 KpqC 1 라운드 NTRU+, SMAUG, TiGER에 대해 특징을 분석하고 각 알고리즘을 소개한다. IV 장에서는 세 스킴의 공통적인 안전성 분석 방법으로 사용되는 LWE 공격 아이디어를 소개한다. V 장에서는 각 제안문서에서 제시한 구현 성능을 분석한다. VI 장에서는 결론을 서술한다.

Ⅱ. Preliminaries

2.1. NTT

Number Theory Transform (NTT)는 Discrete Fourier Transform의 일반화[3]로, 다항식환(polynomial ring) 위에서의 곱셈, 나눗셈 등의 연산을 효율적으로 계산하기 위한 변환 방법이다. 양의 정수 q와 n, 차수가 n인 다항식 $\phi(X)$이 있을 때, 다항식환 $R_q = Z_q[X]/(\phi(X))$을 정의할 수 있다. 또한, $\phi(X)$가 차수가 작은 다항식들의 곱으로 인수 분해되는 경우 Chinese Remainder Theorem을 사용하여 R_q

본 연구는 국가보안기술연구소 위탁과제(KpqC 공모전 격자기반 알고리즘 기반문제 안전성 분석 기술연구, 2023-080) 지원으로 수행되었습니다.
* 성신여자대학교 융합보안공학과 (교수, jooheelee@sungshin.ac.kr)

를 작은 ring 여러 개의 곱으로 나타낼 수가 있다. 이러한 성질을 이용하여 R_q에서의 다항식에 대해 NTT 변환을 적용할 시 다항식끼리의 곱셈 연산을 작은 ring 위에서의 벡터끼리의 Hadamard 곱(Component-wise 곱셈)으로 대체할 수 있으며, 예를 들어 $\phi(X)$가 일차식들의 곱으로 인수 분해되면 Z_q^n에서의 Hadamard 곱으로 대체할 수 있기 때문에 이를 병렬화하여 효율적으로 계산할 수 있다. NTT 연산은 특수한 환에서만 적용 가능하며 NTT 적용 가능한 환을 NTT-friendly ring이라고 한다. NTT-friendly ring에 대한 $\phi(X)$의 예시로는 $\phi(X)=X^n\pm 1, n=2^k$, $\phi(X)=X^n-X^{n/2}+1, n=2^i3^j$ 등이 있다.

2.2. 격자 기반 난제 소개

본 절에서는 격자 기반 난제인 NTRU, Learning with Errors (LWE), Learning with Rounding (LWR) 문제를 설명한다.

- **NTRU 문제**는 초창기 격자 기반 양자내성암호 알고리즘인 NTRU 공개키 암호[4]에서 제시되었으며, 정의는 다음과 같다 : 먼저 양의 정수 q와 n, 차수가 n인 다항식 $\phi(X)$이 있을 때, 다항식환 $R=Z[X]/(\phi(X))$과 $R_q=Z_q[X]/(\phi(X))$을 정의할 수 있다. 이때, $h\in R_q$에 대해 $\|f\|, \|g\| \le \sqrt{q}/\gamma$ (단, $\gamma>0$은 실수)이면서 $(f,g)\neq(0,0)$인 $f,g\in R$가 존재하여 $f\cdot h=g \bmod q$ 를 만족하면 h를 (γ,q)-NTRU 인스턴스라고 부른다. (γ,q)-NTRU 문제는 (γ,q)-NTRU 인스턴스의 분포와 R_q에서의 Uniform 분포를 구분하는 문제이다. 즉, NTRU 가정은 크기가 작은 두 다항식 $f,g\in R$을 샘플링하고 $h=g\cdot f^{-1} \bmod q$(단, f의 R_q에서의 역원이 존재할 때)를 계산하였을 때 이를 R_q 위의 Uniform 분포에서 샘플링한 것과 구분하기 어렵다는 것이다.
- **LWE 문제**는 Regev[5]에 의해 최초로 제시된 후, 기존 격자 기반 난제들과의 환원 관계로 인해 많은 주목을 받았다[6,7]. 양의 정수 n, q와 n 차원 벡터 $s\in Z^n$, Z^n 위의 분포 χ에 대해 LWE 분포 $A_{n,q,\chi}^{LWE}(s)$는 $Z_q^n\times Z_q$ 위에서 다음과 같은 샘플링을 통해 정의된다 : Z_q^n 에서 a를 Uniform 샘플링하고, 크기가 작은 에러 $e\leftarrow\chi$를 샘플링하여 $(a, b=\langle a,s\rangle+e)\in Z_q^n\times Z_q$ 를 계산 및 출력한다. LWE 문제는 $A_{n,q,\chi}^{LWE}(s)$ 분포와 $Z_q^n\times Z_q$ 위에서의 Uniform 분포를 구분하는 문제이다. 또한, LWE 문제를 다항식환 $R_q=Z_q[X]/(\phi(X))$ 또는 모듈(module) R_q^k 위에서의 문제인 Ring-LWE (RLWE)와 Module-LWE (MLWE)로 자연스럽게 변형할 수 있는데 이 경우 대수적 구조를 활용하여 효율적인 알고리즘 구성이 가능하다.
- Banerjee 등[8]에 의해 최초로 제시된 **LWR 문제**에서는 deterministic한 라운딩 과정을 통해 선형 방정식에 에러를 추가한다. 양의 정수 n, q, p와 n 차원 벡터 $s\in Z^n$에 대해 LWR 분포 $A_{n,q,p}^{LWR}(s)$는 $Z_q^n\times Z_p$ 위에서 다음과 같은 샘플링을 통해 정의된다 : Z_q^n 에서 a를 Uniform하게 샘플링하고, $(a, b=\lfloor p/q\cdot(\langle a,s\rangle \bmod q)\rceil)\in Z_q^n\times Z_p$ 를 계산 및 출력한다. LWR 문제는 $A_{n,q,p}^{LWR}(s)$ 분포와 $Z_q^n\times Z_p$ 위에서의 Uniform 분포를 구분하는 문제이다. LWR 문제도 LWE 문제와 마찬가지로 Ring-LWR (RLWR), Module-LWR (MLWR)로 자연스럽게 변형 및 확장할 수 있다.

2.3. NTRU 기반 PKE/KEM

NTRU 계열의 스킴들은 NIST PQC 공모 1 라운드에 NTRUEncrypt[9], NTRU-HRSS-KEM[10], NTRU-Prime[11] 총 3종이 제출되었으며 이 중 NTRUEncrypt와 NTRU-HRSS-KEM이 'NTRU'[12]로 merge되면서 3 라운드까지 진출했다. 한편, 'NTRU'는 polynomial inversion 연산이 가능하지만 대신 NTT 계산이 불가능한 다항식환을 사용하며 이로 인해 LWE/LWR 계열 스킴들보다 속도가 느리다는 단점이 있었다.

Lyubashevsky와 Seiler[13]는 NTT를 사용할 수 있는 polynomial ring인 $Z_q[X]/(X^n-X^{n/2}+1)$에

서의 NTRU 스킴을 새롭게 제안하였으며, Duman 등[14]은 이를 개량하여 NTRU의 단점들을 극복한 새로운 알고리즘으로 NTRU-A, NTRU-B, NTRU-C를 제시하였다.

기존 NTRU에서는 암복호화 시 복호화 성공 조건식이 메시지에 대한 관계식으로 나타나며, 메시지를 임의로 설정하여 복호화 오라클에 질의할 수 있는 공격자가 복호화 오류를 이용하여 비밀키에 대한 정보를 얻을 수 있기 때문에 q를 크게 설정함으로써 해당 공격을 원천 차단하는 방식을 사용할 수 있다(e.g., NTRU-HRSS-KEM에서 Security Category I에 해당하는 파라미터로 $q=8192$를 사용). 그러나 q가 커질수록 안전성을 유지하기 위해 파라미터 크기가 전반적으로 커지면서 효율성을 해치는 문제가 있다. 따라서 Duman 등은 새로운 메시지 인코딩 방식인 Generalized One-Time Pad (GOTP)를 제안하고 이를 이용하여 복호화 성공 조건식에서 메시지를 제거함으로써 이러한 복호화 오류를 공격자가 이용할 수 없도록 하였다. 이 방식에서는 기존 NTRU와는 다르게 q를 크게 설정하지 않아도(e.g., Security Category I에 대해 다항식환 $Z_{2917}[X]/(X^{648}-X^{324}+1)$를 사용) 해당 공격에 대한 안전성을 유지할 수 있다.

2.4. Lizard/RLizard

Lizard[15]는 최초의 LWR에 기반한 공개키 암호 및 KEM으로, NIST PQC 공모 1 라운드에 제안되었다. 정확하게는, Lizard는 키 생성 과정에서 LWE 인스턴스를 사용하고, 암호화 과정에서는 LWR 인스턴스를 사용하여 LWE와 LWR 두 난제 모두에 기반한 공개키 암호 및 KEM으로 설계되었다. 또한, RLizard[16]에서는 다항식환 구조를 도입하여 공개키 크기를 압축하였으며, 암호문 (c_1,c_2)에 대해 c_2의 하위비트를 추가로 소거하여 전송량을 최적화하였다. (R)Lizard의 대표적인 특징 중 하나는 Hamming weight가 미리 정해진 sparse한 벡터를 키 생성과 암호화 과정의 (R)LWE/(R)LWR 비밀값(s)으로 사용한다는 것이다. 이러한 sparse한 벡터를 사용하는 이점은 키 생성 및 암복호화 시 행렬끼리의 곱셈 연산 또는 다항식 간의 곱셈을 모두 덧셈으로 환원하여 효율적인 구현이 가능하다는 것에 있다. 또한 특정 파라미터 조건을 만족시킬 경우, 암호화에서 LWR 인스턴스 계산 시 라운딩 연산은 bitwise shift 연산으로 구현이 가능하기 때문에 효율적이다. 이러한 특징들로 인해 RLizard는 NTT를 사용하지 않음에도 불구하고 키 생성, 암복호화 속도가 매우 빠른 편이다.

2.5. CRYSTALS-KYBER

LWE/LWR 기반의 공개키 암호 및 KEM의 공개키 및 암호문 크기를 압축하기 위해 초기 연구들에서는 자연스러운 확장인 다항식환 버전의 RLWE/RLWR 기반 스킴을 사용하였다[16,17]. 다만 이때 polynomial ring을 구성하기 위해 사용한 다항식 $\phi(X)=X^n\pm1, n=2^k$의 차수가 $n=512, 1024, 2048, \ldots$ 등 sparse하게 존재하기 때문에 파라미터의 선택이 자유롭지 못했다.

NIST 공모에 제안된 CRYSTALS-KYBER(Kyber)[18]는 최초로 파라미터의 유연한 선택이 가능한 module 버전의 MLWE 기반 PKE/KEM을 도입하였다. MLWE는 LWE의 자연스러운 확장으로 MLWE 인스턴스를 샘플링하기 위해서는 R_q^k 에서 a를 Uniform하게 샘플링하고, 크기가 작은 에러 $e\leftarrow\chi^n\in R$를 샘플링하여 $(a, b=\langle a,s\rangle+e)\in R_q^k\times R_q$ 를 계산 및 출력한다. 이렇게 생성된 MLWE 인스턴스의 경우 R_q에 사용되는 다항식의 차수를 작은 값(e.g., 256)으로 고정해놓고 $k=1,2,3,\ldots$를 선택하여 파라미터를 조율할 수 있다. Kyber는 기반 문제의 안전성을 희생하지 않으면서도 속도나 사이즈 측면에서 균형 잡힌 효율적인 구성 방식을 사용하여 NIST 표준화에 최종 선정되었다.

Ⅲ. 격자 기반 PKE/KEM 알고리즘

이 장에서는 KpqC 1 라운드 알고리즘인 NTRU+, SMAUG, TiGER에 대해 다룬다.

3.1. NTRU+

NTRU+는 NIST 표준화 공모 3 라운드 후보였던 NTRU, 그리고 이를 개선한 NTTRU[13], Duman 등의 NTRU-(A,B,C)[14] PKE/KEM 알고리즘과 유사한

설계 사상을 가지고 제안된 스킴이다. NTRU+는 NTRU 문제, 그리고 비밀키와 에러가 특수한 분포에서 뽑히는 RLWE 문제에 기반한 스킴으로, 가장 큰 특징은 다음과 같다.

- NTT 연산 활용 ; NTRU+에서는 NTTRU, NTRU-(A,B,C)와 유사하게 NTT-friendly ring인 $Z_q[X]/(X^n - X^{n/2}+1)$(단, $n = 2^i3^j$, $q = 3457$)을 사용하여, 모든 연산을 NTT로 변환하여 수행한다.
- 공격자가 메시지와 암호화 시 사용하는 랜덤값을 임의로 선택했을 때의 복호화 오류율 상계(upper bound)가 안전성 파라미터에 대해 무시 가능(negligible)한 수준으로 도출될 수 있도록 메시지 인코딩 방식인 Semi-generalized One Time Pad (SOTP)를 제안하였다. SOTP ($SOTP$, Inv)는 메시지 $x \in \{0,1\}^n$와 또다른 입력값 $u = (u_1, u_2) \in \{0,1\}^{2n}$에 대해 다음과 같은 방식으로 인코딩을 수행한다 : $SOTP(m,u) = (m \oplus u_1) - u_2 \in \{-1,0,1\}^n$. 또한 다음과 같은 디코딩 방식(Inv)을 사용하여 원래의 메시지를 복구할 수 있다 :

$$\mathrm{Inv}(y,u) = (y + u_2) \oplus u_1.$$

- IND-CCA 변환 시 전통적인 Fujisaki-Okamoto (FO) 변환[19,20]을 이용하지 않았으며, 복호화에서 복구된 메시지를 재암호화하여 입력 암호문과 비교하거나, 복호화 결과의 범위를 확인하는 대신, 암호화에 사용한 랜덤값을 복구한 후 이를 비교하는 방식을 사용하였다.

안전성 파라미터 λ, $q = 3457$에 대해 $R_q = Z_q[X]/(X^n - X^{n/2}+1)$위에서 정의된 NTRU 비밀값 분포 ψ_1^n, 해시함수 H와 G에 대해 NTRU+의 IND-CCA KEM (KeyGen, Encaps, Decaps)은 다음과 같이 구성된다.

- KeyGen(1^λ)$\rightarrow (pk, sk)$
  1) $f', g \leftarrow \psi_1^n$
  2) $f = 3f' + 1$
  3) R_q에서 f, g의 역원이 존재하지 않을 경우 처음부터 다시 시작
  4) $(pk, sk) = (h = 3g \cdot f^{-1} \bmod q, f)$
- Encaps(pk)$\rightarrow (K, c)$
  1) $m \leftarrow \{0,1\}^n$
  2) $(r, K) = H(m)$
  3) $M = SOTP(m, G(r))$
  4) $c = h \cdot r + M \bmod q$
- Decaps(sk, c)$\rightarrow K$ or $\perp$
  1) $M = (c \cdot f \bmod q) \bmod 3$
  2) $r = (c - M) \cdot h^{-1} \bmod q$
  3) $m = \mathrm{Inv}(M, G(r))$
  4) $(r', K) = H(m)$
  5) $r = r'$이면 K 출력, $r \neq r'$이면 $\perp$ 출력

3.2. SMAUG

SMAUG는 NIST 표준화 공모에 제출된 Lizard[15]/RLizard[16]와 유사한 설계 사상을 가지며, Sparse 비밀값을 갖는 Module-LWE, Module-LWR 기반의 스킴이다. 가장 주요한 특징은 다음과 같다.

- 키 생성 시 MLWE, 암호화 시 MLWR의 인스턴스를 사용한다.
- 비밀키 생성 및 암호화 시 MLWE와 MLWR에 대한 비밀값의 Hamming weight를 정해두고 Sparse한 벡터로 추출하는 특징을 가지고 있다.
- 라운딩 연산을 bitwise shift 연산으로 대체하기 위해 MLWR 모듈러스 q, p를 2의 멱승으로 설정하였으며, NTT를 사용하지 않았다. 대신, Lizard/RLizard와 마찬가지로 다항식 간의 곱셈을 정해진 개수의 다항식 간의 덧셈으로 치환하여 효율적으로 연산한다.

안전성 파라미터 λ, eXtendable Output Function (XOF) XOF, 랜덤시드에 대한 XOF의 출력값으로 $R_q^{k \times k}$의 행렬 A를 구성하는 expandA 함수, Hamming weight h인 sparse 비밀값을 생성해주는 $HWT_h(\cdot)$, 이산 가우시안 샘플링을 수행하는

$DG(\cdot)$, 라운딩 연산 $\lfloor \cdot \rceil$, Key Derivation Function KDF, 해시함수 H와 G에 대해 SMAUG의 IND-CCA KEM (KeyGen, Encaps, Decaps)은 다음과 같이 구성된다.

- KeyGen(1^λ)$\rightarrow(pk, sk)$
  1) $seed \leftarrow \{0,1\}^{256}$
  2) $(\rho, \tau) \leftarrow XOF(seed)$
  3) $A \leftarrow \text{expandA}(\rho) \in R_q^{k \times k}$
  4) $s \leftarrow HWT_{h_s}(\tau) \in R^k$
  5) $e \leftarrow DG(\tau) \in R^k$
  6) $b = -A^T \cdot s + e \bmod q$
  7) $d \leftarrow \{0,1\}^{256}$
  8) $(pk, sk) = ((\rho, b), (s, d))$
- Encaps(pk)$\rightarrow(K, c)$
  1) $pk := (\rho, b), A \leftarrow \text{expandA}(\rho)$
  2) $m \leftarrow \{0,1\}^{256}$
  3) $r \leftarrow HWT_{h_r}(G(m, H(pk)))$
  4) $c_1 \leftarrow \lfloor p/q \cdot (A \cdot r \bmod q) \rceil \in R_p^k$
  5) $c_2 \leftarrow \lfloor p/q \cdot (b^T \cdot r + q/t \cdot m \bmod q) \rceil$
  6) $c = (c_1, c_2) \in R_p^k \times R_p$
  7) $K = KDF(m, H(c))$
- Decaps(sk, c)$\rightarrow K$
  1) $sk := (s, d)$
  2) $m' \leftarrow \lfloor t/p \cdot (c_2 + c_1^T \cdot s \bmod p) \rceil$
  3) $r' \leftarrow HWT_{h_r}(G(m', H(pk)))$
  4) $c_1' \leftarrow \lfloor p/q \cdot (A \cdot r' \bmod q) \rceil \in R_p^k$
  5) $c_2' \leftarrow \lfloor p/q (b^T \cdot r' + q/t \cdot m' \bmod q) \rceil$
  6) $c' = (c_1', c_2') \in R_p^k \times R_p$
  7) $c = c'$이면 $K' = KDF(m', H(c'))$ 출력, $c \neq c'$이면 $K' = KDF(d, H(c))$ 출력

이때, Decaps의 3)~6)은 복구된 메시지의 재암호화 과정으로, Encaps의 3)~6)과 동일하다.

3.3. TiGER

TiGER는 NIST 표준화 공모에 제출된 Lizard[15]/RLizard[16], LAC[21], Round5[22]에서 영향을 받아 제안된 스킴으로, Sparse 비밀값을 갖는 Ring-LWR, Ring-LWE 기반의 스킴이다. 가장 주요한 특징은 다음과 같다.

- 키 생성 시 RLWR, 암호화 시 RLWR의 인스턴스를 사용한다.
- 비밀키 생성 및 암호화 시 RLWR과 RLWE에 대한 비밀값의 Hamming weight를 정해두고 Sparse한 벡터로 추출하는 특징을 가지고 있다. 또한, RLWE의 에러 역시 sparse한 벡터로 샘플링한다.
- SMAUG와 마찬가지로 RLWR 모듈러스 q,p를 2의 멱승으로 설정하여 라운딩을 bitwise shift로 계산하였으며, NTT를 사용하지 않았다.
- 1 바이트 크기를 갖는 작은 q(q=256)를 사용하여 암호문과 공개키 사이즈가 비교적 작다. 예를 들어 Security Category V에 대해 공개키 928 바이트, 암호문 1,152 바이트의 크기를 갖는다.
- 메시지 인코딩 시 Error Correcting Code (ECC)를 사용하여 에러를 복구하는 방식으로 복호화 오류율을 무시가능한 수준으로 조절하였다. 이때, ECC의 종류로는 XEf [22] 또는 D2 [17]를 사용한다. LAC[21]에서도 이러한 방식을 사용하여 복호화 오류율을 조율하였으나, LAC은 ECC의 종류로 BCH[23]를 사용한다.

안전성 파라미터 λ, XOF $SHAKE256$, Hamming weight h인 sparse 비밀값을 생성해주는 $HWT_h^n(\cdot)$, 라운딩 연산 $\lfloor \cdot \rceil$, ECC $(ECC.Ecd, ECC.Dcd)$, 해시함수 H와 G에 대해, TiGER의 IND-CCA KEM (KeyGen, Encaps, Decaps)은 다음과 같이 구성된다.

- KeyGen(1^λ)$\rightarrow(pk, sk)$
  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) $s \leftarrow HWT_{h_2}^{n}(seed_s) \in R$

5) $b \leftarrow \lfloor (p/q) \cdot (a \bullet s \bmod q) \rceil$

6) $u \leftarrow R_2$

6) $pk \leftarrow (seed_a, b),\ sk \leftarrow (s, u)$

- Encaps$(pk) \rightarrow (K, c)$

1) $m \leftarrow \{0,1\}^d$

2) $r \leftarrow HWT_{h_r}^{n}(H(m))$

3) $seed_{e_1} \leftarrow (H(m) + Nonce),$
$seed_{e_2} \leftarrow (H(m) + Nonce + 1)$

4) $e_1 \leftarrow HWT_{h_e}^{n}(seed_{e_1}),$
$e_2 \leftarrow HWT_{h_e}^{n}(seed_{e_2})$

5) $pk := (seed_a, b),$
$a \leftarrow SHAKE256(seed_a, n/8)$

6) $c_1 \leftarrow \lfloor (k_1/q) \cdot (a \bullet r + e_1 \bmod q) \rceil$

7) $c_2 \leftarrow \left\lfloor k_2/q \cdot \begin{pmatrix} (q/2) \cdot ECC.Ecd(m) \\ +((q/p) \cdot b) \bullet r + e_2 \end{pmatrix} \right\rceil$

8) $c = (c_1, c_2) \in R_{k_1} \times R_{k_2}$

9) $K \leftarrow G(c, m)$

- Decaps$(sk, c) \rightarrow K$

1) $sk := (s, u),\ c := (c_1, c_2)$

2) $\widehat{m}' \leftarrow \left\lfloor (2/q) \cdot \begin{pmatrix} (q/k_2) \cdot c_2 \\ -((q/k_1) \cdot c_1) \bullet s \end{pmatrix} \right\rceil$

3) $m' \leftarrow ECC.Dcd(\widehat{m}')$

4) $r' \leftarrow HWT_{h_r}^{n}(H(m'))$

5) $seed'_{e_1} \leftarrow (H(m') + Nonce),$
$seed'_{e_2} \leftarrow (H(m') + Nonce + 1)$

6) $e'_1 \leftarrow HWT_{h_e}^{n}(seed'_{e_1}),$
$e'_2 \leftarrow HWT_{h_e}^{n}(seed'_{e_2})$

7) $pk := (seed_a, b),$
$a \leftarrow SHAKE256(seed_a, n/8)$

8) $c'_1 \leftarrow \lfloor (k_1/q) \cdot (a \bullet r' + e'_1 \bmod q) \rceil$

9) $c'_2 \leftarrow \left\lfloor k_2/q \left(\begin{matrix} (q/2) \cdot ECC.Ecd(m') \\ +((q/p) \cdot b) \bullet r' + e'_2 \end{matrix} \right) \right\rceil$

10) $c' = (c'_1, c'_2) \in R_{k_1} \times R_{k_2}$

11) $c = c'$이면 $K' = G(c', m')$ 출력, $c \neq c'$이면 $K' = G(c', u)$ 출력

이때, Decaps의 4)~10)은 복구된 메시지의 재암호화 과정으로, Encaps의 2)~8)과 동일하다.

3.4. 비교

본 절에서는 NTRU+, SMAUG, TiGER의 특징과 스킴 구조에 대한 비교를 통해 이해를 증진시키고자 한다.

- 기반 난제 : 각 알고리즘은 앞서 기술한 바와 같이 KeyGen과 Encaps에서 NTRU, RLWE, RLWR, MLWE, MLWR 등 여러 난제의 인스턴스를 활용하고 있으며, [표 1]의 KeyGen과 Encaps 행에 해당 내용을 분류하였다. LWE/LWR에 대해 비밀값의 분포는 n, q 등의 파라미터와 함께 안전성에 영향을 주는 중요한 요소이다. [표 1]의 LWE/LWR secret 행에서는 비밀값이 특정 형태로 사용될 경우, 즉, binary, ternary (각 성분이 $\{-1, 0, 1\}$의 원소) 이거나 정해진 Hamming weight 값을 갖는 sparse 형태인 경우 이를 분류하였다.
- SMAUG와 TiGER는 2의 멱승 n, q에 대해 $Z_q[X]/(X^n+1)$을 사용하며 NTT를 활용하지 않는다. NTRU+는 $n = 2^i 3^j$에 대해 $Z_q[X]/(X^n - X^{n/2} + 1)$을 사용하며 NTT 연산을 활용한다. [표 1]의 NTT와 Ring 행에 해당 내용을 표시하였다.

Ⅳ. 안전성 분석 방법

세 스킴 각각은 NIST Security Category I, III, V의 안전성을 갖는 파라미터를 제시하였다. 각 스킴에 대한 IND-CCA 안전성 증명이 제안문서를 통해 제시되어 있기 때문에 각 스킴이 기반하고 있는 난제의 공격량 분석을 통해 실질적인 안전성 수준을 측정할 수 있다.

SMAUG와 TiGER에서는 MLWR/RLWR 문제에 대해 $(-q/2p, q/2p]$에서의 균등 분포에서 랜덤 에러를 생성하는 MLWE/RLWE 문제로 간주한 후 이에

〔표 1〕 NTRU+, SMAUG, TiGER 비교 : KeyGen, Encaps는 각각 키 생성과 키 캡슐화(암호화)의 기반난제, LWE/LWR Secret은 (M,R)LWE/(M,R)LWR의 비밀값 형태, NTT는 NTT 사용 여부, Ring은 각 스킴에서 사용하는 polynomial ring를 나타낸다.

	NTRU+	SMAUG	TiGER
KeyGen	NTRU	MLWE	RLWR
Encaps	RLWE	MLWR	RLWE
LWE/ LWR secret	binary	sparse ternary	sparse ternary
NTT	O	X	X
Ring	$\frac{Z_q[X]}{(X^n-X^{n/2}+1)}$	$\frac{Z_q[X]}{(X^n+1)}$	$\frac{Z_q[X]}{(X^n+1)}$
공개키	$h=g/f$	$(\rho, b=-A^T s+e$	$(seed_a, b=\lfloor \frac{p}{q}(as) \rceil)$
암호문	$c=h\cdot r+M$	(c_1, c_2)	(c_1, c_2)
공개키 크기	$n\cdot log(q)$	$256+n\cdot log(q)$	$256+n\cdot log(p)$
암호문 크기	$n\cdot log(q)$	$n(k+1)\cdot log(p)$	$n\cdot log(k_1 k_2)$

대한 공격량을 측정한다. 또한, MLWE/RLWE 문제의 가장 효율적인 공격들에서는 모듈이나 다항식환의 대수적 구조를 사용하지 않고 해당 문제를 기본적인 형태의 LWE 문제로 환원하여 공격을 수행한다. 따라서, 이 절에서는 세 스킴이 공통적으로 고려하고 있는 LWE 공격 방법으로 Primal, Dual 공격의 아이디어를 간략히 소개한다.

4.1. Primal 공격

LWE 인스턴스 $(A, b=As+e) \in Z_q^{m\times n}\times Z_q^m$에 대해, $B=(A\,|\,I_m\,|\,b) \in Z_q^{m\times(n+m+1)}$라고 하자. 이때, 다음과 같은 격자를 정의할 수 있다.

$$\Lambda_{primal}=\{v \in Z_q^{n+m+1} : Bv \bmod q = 0\}$$

또한, $(-s, e, 1) \in Z_q^{n+m+1}$가 이와 같이 정의된 격자 위의 짧은 벡터라는 것을 알 수 있다. 따라서, 격자 Λ_{primal} 위에서의 짧은 벡터를 찾음으로써 확률적으로 $(-s, e, 1) \in Z_q^{n+m+1}$를 찾을 수 있고, 이로부터 LWE 비밀값 s와 에러 e를 얻을 수 있다. 이때, 격자 위에서의 짧은 벡터를 찾기 위해서는 BKZ 알고리즘[24]을 사용한다.

4.2. Dual 공격

LWE 인스턴스 $(A, b=As+e) \in Z_q^{m\times n}\times Z_q^m$에 대해, 다음과 같은 격자를 정의할 수 있다.

$$\Lambda_{dual}=\{(u,v) \in Z_q^m \times Z_q^n : A^T u = v\}$$

또한, 위와 같이 정의된 격자 위에서 짧은 벡터 (x,y)를 찾을 경우, $\langle x,b\rangle = \langle y,s\rangle + \langle x,e\rangle \bmod q$가 성립하는데, s, e의 크기가 작으므로 $\langle y,s\rangle + \langle x,e\rangle$값이 $q/2$보다 작은 값을 가질 것으로 추정할 수 있다. 따라서 이를 이용해 $Z_q^{m\times n}\times Z_q^m$ 위의 Uniform 분포에서의 샘플링과 LWE 샘플링 사이의 차이를 구분할 수 있다. 격자 위에서의 짧은 벡터 (x,y)를 찾기 위해서는 BKZ 알고리즘[24]을 이용한다.

Ⅴ. 구현 성능 비교

본 절에서는 KpqC 공모 1 라운드에 제출된 NTRU+, SMAUG, TiGER 각 스킴의 파라미터와 제시된 구현 성능을 조사 및 비교한다. 동일한 안전성을 갖는 경우의 성능을 조사하기 위해, NIST Security Category III에 해당하는 NTRU+864, SMAUG192, TiGER192 파라미터에 대해 해당하는 공개키 및 암호문 사이즈와 제안문서 상의 구현성능을 [표 2]에 표기하였다. 공개키, 암호문 사이즈는 바이트 수로 표기하였으며, KeyGen, Encaps, Decaps 각각에 대한 속도는 reference implementation에 대한 성능(괄호 안에 AVX2 최적화 구현성능 표시)을 cycle 수로 표기하였다. 참고로, 구현성능의 경우 NTRU+는 Intel Core i7-8700K(3700MHz), SMAUG는 AMD Ryzen 3700X(3589MHz), TiGER는 AMD Ryzen3 2200G(3.5GHz) 환경에서 측정되었다.

Ⅵ. 결 론

본 고에서는 KpqC 1 라운드 알고리즘 중 격자 기반 PKE/KEM에 해당하는 NTRU+, SMAUG, TiGER

〔표 2〕 NTRU+864, SMAUG192, TiGER192 파라미터와 제시된 성능 비교 : n은 다항식 차수(k는 모듈의 차원), q,p는 LWE/LWR 모듈러스, h_s,h_r은 비밀벡터의 Hamming weight 파라미터를 나타냄

	NTRU+	SMAUG	TiGER
$n \cdot k$	864	256 • 3	1024
q	3457	1024	256
p	-	256	64
(h_s, h_r)	-	(150, 147)	(84, 84)
pk (bytes)	1,296	992	800
ct (bytes)	1,296	1,024	1,024
$pk+ct$ (bytes)	2,592	2,016	1,824
KeyGen (cycles)	339,912 (14,068)	106,956	68,334
Encaps (cycles)	169,634 (19,293)	115,128	109,179
Decaps (cycles)	262,017 (17,671)	124,812	129,363

에 대해 살펴보았다. 각 스킴의 설계상의 특징과 기반 문제, IND-CCA KEM 구조, 그리고 제시된 파라미터와 구현 성능을 분석 및 비교하였다.

참 고 문 헌

[1] Shor, Peter W. “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”, *SIAM review* 41.2 (1999): pp. 303-332.

[2] KPQC Competition Round 1, available at https://www.kpqc.or.kr/competition.html

[3] John M Pollard. “The fast fourier transform in a finite field”, *Mathematics of computation*, 25(114), pp. 365–374, 1971

[4] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. “NTRU: A ring-based public key cryptosystem”, *In Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of LNCS, pp. 267–288. Springer, Heidelberg, Germany, June 1998.

[5] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”, *J. ACM*, 56(6), 2009.

[6] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. “Classical hardness of learning with errors”, *In Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 575–584. ACM, 2013.

[7] Chris Peikert. “Public-key cryptosystems from the worst-case shortest vector problem”, *In Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 333–342. ACM, 2009.

[8] Abhishek Banerjee, Chris Peikert, and Alon Rosen. “Pseudorandom functions and lattices”, *In Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 719–737. Springer, 2012.

[9] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. NTRUEncrypt. *Technical report, National Institute of Standards and Technology*, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[10] John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. NTRU-HRSS-KEM. *Technical report, National Institute of Standards and Technology*, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

[11] Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C. “NTRU Prime: reducing attack surface at low cost”, *In: International Conference on Selected Areas in Cryptography.* pp. 235–260. Springer (2017)

[12] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. NTRU. Technical report, National Institute of Standards and Technology, 2020.

available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[13] Vadim Lyubashevsky and Gregor Seiler. “NTTRU: truly fast NTRU using NTT”. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3): ppl. 180 – 201, 2019.

[14] Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G., Unruh, D. “A Thorough Treatment of Highly-Efficient NTRU Instantiations”, *In: Boldyreva, A., Kolesnikov, V. (eds) Public-Key Cryptography – PKC 2023.* PKC 2023. LNCS, vol 13940. Springer, Cham.

[15] Cheon, J.H., Kim, D., Lee, J., Song, “Lizard: Cut off the tail! A practical post-quantum public-key encryption from LWE and LWR“, *In Proc. 11th Conf. Secur. Cryptogr. Netw.,* Sep. 2018, pp. 160 – 177.

[16] Lee, J., Kim, D., Lee, H., Lee, Y., Cheon, J.H., “RLizard: Post-quantum key encapsulation mechanism for IoT devices”, *IEEE Access* 7, 2080 – 2091 (2018)

[17] Alkim, E., Ducas, L., P¨oppelmann, T., Schwabe, P. “Post-quantum key exchange—a new hope”, *In: 25th USENIX Security Symposium (USENIX Security 16)*. pp. 327 – 343 (2016)

[18] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. “CRYSTALS-KYBER”, *Technical report, National Institute of Standards and Technology*, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[19] Eiichiro Fujisaki and Tatsuaki Okamoto. “Secure integration of asymmetric and symmetric encryption schemes”, *In Michael J. Wiener, editor, Advances in Cryptology – CRYPTO’99,* volume 1666 of LNCS, pp. 537 – 554, Santa Barbara, CA, USA, August 15 – 19, 1999. Springer, Heidelberg, Germany

[20] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. “A modular analysis of the Fujisaki-Okamoto transformation”, *In Yael Kalai and Leonid Reyzin, editors, TCC 2017: 15th Theory of Cryptography Conference*, Part I, volume 10677 of LNCS, pp. 341 – 371, Baltimore, MD, USA, November 12 – 15, 2017. Springer, Heidelberg, Germany

[21] Lu, X., Liu, Y., Zhang, Z., Jia, D., Xue, H., He, J., Li, B., Wang, K., Liu, Z., Yang, H. “LAC: Practical ring-lwe based public-key encryption with byte-level modulus”, *IACR Cryptology ePrint Archive* 2018, 1009 (2018)

[22] Baan, H., Bhattacharya, S., Fluhrer, S.R., Garcia-Morchon, O., Laarhoven, T., Rietman, R., Saarinen, M.J.O., Tolhuizen, L., Zhang, Z. “Round5: Compact and fast post-quantum public-key encryption”, *IACR Cryptology ePrint Archive* 2019, 90 (2019)

[23] Ivan Djelic. “Bch source code”, https://github.com/jkent/python-bchlib/tree/master/bchlib.

[24] Chen, Y., Nguyen, P.Q. “Bkz 2.0: Better lattice security estimates”, *In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011.* pp. 1 – 20. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

〈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