정 보 보 호 학 회 지
제 33 권 제3호, 2023. 6

KpqC 공모전에 제출된 Hash-and-Sign 구조의 격자 기반 서명 기법 분석

김 주 언*, 박 종 환**

요 약

쇼어 알고리즘으로 기존의 공개키 암호 시스템이 무력화될 수 있음이 밝혀지면서 양자 컴퓨팅 환경에서도 안전한 격자 기반 PQC(Post Quantum Cryptography)가 대두되고 있다. FALCON은 NIST PQC 공모전 표준 후보로 최종 선정된 기법으로 fast fourier 트랩도어 샘플러(trapdoor sampler)를 이용한 hash-and-sign 구조의 격자 기반 서명이다. FALCON은 공개키와 서명의 크기가 작고 안전성이 높지만, 구현이 어렵고 빠르지 않다. KpqC(Korea PQC) 공모전에 제출된 Peregrine과 SOLMAE는 FALCON의 샘플링 방식을 바꾸어 효율성을 개선하였으나, 그로 인해 안전성 손실이 발생한다. 본 논문에서는 Peregrine과 SOLMAE를 분석하고 FALCON과 함께 비교하며 한계점과 향후 개선할 부분에 대해서 제시한다.

Ⅰ. 서 론

기존 컴퓨터로는 풀기 어려웠던 RSA, ECC 등의 공개키 알고리즘은 소인수 분해 문제나 이산대수 문제의 어려움을 기반으로 한다. 그러나 양자컴퓨터를 이용하여 이러한 수학적 난제들을 다항 시간 내에 푸는 쇼어 알고리즘[1]이 제시되면서 양자컴퓨터로도 풀기 어려운 격자 기반 공개키 알고리즘에 관한 연구가 활발히 이루어지고 있다. NIST가 주관한 양자내성암호 표준화 공모전인 NIST PQC에서 표준 후보로 최종 선정된 기법 4가지 중 3가지(Kyber, Dilithium, FALCON)는 격자 문제의 어려움을 기반으로 설계되었다.

그중 FALCON[2]은 GPV 프레임워크의 hash-and-sign 서명 구조로, NTRU 격자를 기반으로 하며 fast fourier 트랩도어 샘플러를 이용한다. FALCON은 공개키와 서명 길이가 짧고 안전성이 높지만, 서명 알고리즘이 복잡해 구현이 어렵고 빠르지 않으며 부동소수점 연산으로 인해 부채널 공격을 막기 어렵다는 단점이 있다.

GPV 프레임워크의 hash-and-sign 서명은 메시지를 해시 함수에 입력한 후 트랩도어 샘플러를 이용하여 서명을 생성한다. 이때 서명 생성 단계의 샘플링 과정에서 효율성과 안전성의 트레이드 오프(trade-off)가 존재하여 FALCON의 후속 연구들은 FALCON과 다른 트랩도어 샘플러를 사용하여 효율성을 개선하였다[3-5]. 국내 양자내성암호 국가공모전 KpqC에 제출된 Peregrine[6]과 SOLMAE[7]도 hash-and-sign 격자 기반 서명으로, 표 1과 같이 샘플링 방식을 바꾸어 효율성을 높이고자 하였다.

FALCON은 가우시안(Discrete Gaussian) 분포에서 랜덤화된 fast fourier nearest plane 알고리즘을 트랩도어 샘플러로 사용하는 반면 Peregrine은 이항분포(Centered Binomial)에서 랜덤화된 round off 알고리즘을 사용하여 속도가 빠르고 간단하며 병렬성을 만족한다. Peregrine은 이항분포를 사용함으로써 정수 연산이 가능하여 FALCON보다 부채널 공격에 대해 더 안전하지만, 랜덤화를 하여도 개인키에 대한 정보를 노출할 수 있다. 즉 Peregrine은 FALCON의 효율성은 개선하였지만, 서명 위조에 대한 안전성이 더 낮고 개인키를 온전히 숨기지 못하는 문제점이 있다.

SOLMAE는 하이브리드(nearest plane 알고리즘과 round off 알고리즘을 결합) 샘플러를 사용하는 FALCON의 후속 연구 MITAKA[5]를 발전시킨 연구다. MITAKA는 속도가 빠르고, 병렬성과 파라미터 선

본 연구는 고려대 암호기술 특화연구센터(UD210027XD)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.
* 고려대학교 정보보호학과 (대학원생, jjjuon@korea.ac.kr)
** 상명대학교 컴퓨터과학과 (교수, jhpark@smu.ac.kr)

〔표 1〕 FALCON, Peregrine, SOLMAE 샘플러 비교

	FALCON	Peregrine	SOLMAE
Lattice decoding algorithm (CVP Solver)	Fast fourier nearest plane algorithm	Round-off algorithm	Hybrid algorithm
Randomization distribution	Discrete Gaussian	Centered Binomial	Discrete Gaussian

택에 유연성을 제공하며 구현이 간단하다. 그러나 FALCON보다 안전성이 낮고 서명 크기가 더 크다는 단점이 있다. 따라서 SOLMAE는 MITAKA의 안전성 향상 및 키 생성 시간 감소를 위해 새로운 최적화 알고리즘을 제안하였다. SOLMAE는 MITAKA의 효율성을 유지하면서 안전성도 높이고자 하였지만, 여전히 FALCON보다는 안전성이 낮다.

본 논문의 구성은 다음과 같다. 2장에서는 배경지식을 소개하고, 3장에서는 FALCON을 설명한다. 4장과 5장에서는 KpqC에 제출된 Peregrine과 SOLMAE를 각각 설명하고, 6장에서는 Peregrine과 SOLMAE를 분석하고 FALCON과 비교한다. 마지막으로 7장에서는 결론을 맺는다.

Ⅱ. 배경지식

정의 1. [SIS 문제] 행렬 $A \in Z_q^{n \times m}$가 주어졌을 때 $Ax = 0 \bmod q$에서 바운드 γ을 넘지 않는 0이 아닌 작은 벡터 $x \in Z^m$ $(||x|| \leq \gamma)$을 찾는 문제를 SIS(Shortest Integer Solution) 문제라고 한다.

정리 1. GPV 프레임워크에서 A의 트랩도어를 알면 SIS 문제를 풀기 쉽지만, A의 트랩도어를 알지 못하면 SIS 문제를 풀기 어렵다.

정의 2. [SVP] 격자 L이 주어졌을 때, 영점 0과 거리가 가장 가까운 L에 있는 벡터 v를 찾는 문제를 SVP(Shortest Vector Problem)라고 한다.

정의 3. [CVP] 격자 L의 기저 B와 L에 있지 않은 임의의 벡터 c가 주어졌을 때, c와 거리가 가장 가까운 L에 있는 벡터 v를 찾는 문제를 CVP(Closest Vector Problem)라고 한다.

정의 4. [가우시안 분포] 평균과 표준편차가 각각 $\mu, \sigma(\sigma > 0)$이고 $\exp = e^x$라고 하자. 가우시안 함수가 $\rho_{\sigma,\mu}(x) = \exp(-|x-\mu|^2/2\sigma^2)$일 때, 정수 상에서 가우시안 분포는 $D_{\mathbb{Z},\sigma,\mu}(x) = \dfrac{\rho_{\sigma,\mu}(x)}{\sum_{z \in \mathbb{Z}} \rho_{\sigma,\mu}(z)}$이다.

정의 5. [이항분포] X는 연속된 μ번의 독립적 시행에서 확률 p를 가질 시행의 횟수를 나타내는 확률변수이고, 이때 x의 범위는 $-\mu/2 \leq x \leq \mu/2$라 하자. 각 시행이 확률 p를 가질 확률이 $(\mu/2)+x$번이고, 확률 $(1-p)$를 가질 확률이 $(\mu/2)-x$일 때 이항분포는 $P[X=x] = \dfrac{\mu!}{((\mu/2)+x)! \cdot ((\mu/2)-x)!} \cdot 2^{-\mu}$이다.

〔표 2〕 표기(Notation)

표기	부르는 약칭
α	Trapdoor quality (트랩도어 퀄리티)
γ	Rejection bound (바운드)
m	Message (메시지)
s	Signature (서명)
q	Modulus (모듈러)
n	Dimension (차수)
L	Lattice (격자)
H	Hash function (해시 함수)
Υ	Distortion factor (왜곡 파라미터)
$\phi(x)$	Polynomial modulus (다항식 모듈러)
$R = Z[x]/\phi(x)$	Polynomial ring (다항식 환)
$K = Q[x]/(x^n+1)$	Cyclotomic field (원분체)
(A,B)	Trapdoor pair (트랩도어 쌍)

2.1. GPV 프레임워크

GPV (Gentry, Peikert, Vaikuntanathan) 프레임워크는 격자 공간의 기저로 표현된 키 쌍으로부터 안전한 서명을 만들기 위한 방법론으로 $As = H(m)$의 "hash-and-sign" 구조이다[8]. GPV 프레임워크의 안전성은 일방향 트랩도어 함수를 이용한 SIS 문제에 기반한다. 따라서 GPV 프레임워크는 $As = H(m)$에서 서명 s의 길이가 짧을수록 더 안전하다.

KeyGen(1^λ):
1. $(A,B) \leftarrow$ **TrapGen**(λ)
2. return pk = A, sk = B

Sign(B, m, γ):
1. $M \leftarrow H(m)$
2. choose c s.t. $c \cdot A^t = M$
3. $v \leftarrow$ **Sample**(B, c)
4. $s \leftarrow c - v$
5. return s

Verify(A, m, s, γ):
1. $M \leftarrow H(m)$
2. **if** $s \cdot A^t = M$ and $||s|| \leq \gamma$ **then**
3. return accept
4. **else**
5. return reject

〔그림 1〕 GPV 프레임워크의 Hash-and-Sign 서명

GPV 프레임워크의 hash-and-sign 구조 서명은 그림 1과 같이 키를 생성하는 KeyGen, 서명을 생성하는 Sign, 그리고 서명을 검증하는 Verify 세 알고리즘으로 이루어져 있다. **KeyGen** 알고리즘에서 **TrapGen** 알고리즘은 $B \times A^t = 0$를 만족하는 트랩도어 쌍 (A,B)을 생성하여 출력한다. 먼저 q를 법으로 하는 격자 $L_q = L(A^t)$을 생성하는 풀 랭크(full-rank) 행렬 $A \in Z_q^{n \times m}$ $(m > n)$를 생성한 후 L_q에 직교하는 격자 $L_q^\perp = L(B)$을 생성하는 행렬 $B \in Z_q^{m \times m}$를 생성한다. **Sign** 알고리즘은 메시지 m을 해시 함수 H에 입력해 $M = H(m)$을 얻고, $c \cdot A^t = M$을 만족하는 벡터 c를 선택한다. 그리고 B를 이용하여 **Sample** 알고리즘으로 c와 가까운 벡터 $v \in L_q^\perp$를 찾아 길이가 짧은 서명 $s = c - v$를 출력한다. **Verify** 알고리즘은 서명의 크기가 작고, 즉 $||s|| \leq \gamma$이고, $s \cdot A^t = c \cdot A^t - v \cdot A^t = M$을 만족하면 accept를 출력하고, 그렇지 않으면 reject를 출력한다.

2.2. NTRU 격자

다항식 환 $R = Z[x]/\phi(x)$에서 $f \in R$가 양의 정수인 모듈러 q에 대해 역변환이 존재한다고 하자. NTRU 격자는 랜덤으로 생성한 $f, g \in R$와 NTRU 방정식 $fG - gF = q \bmod \phi(x)$을 통해 계산한 $F, G \in R$ 4개의 다항식을 이용한다. 이때 4개의 다항식의 계수는 모두 작은 정수이다. $h = g/f \bmod q$이고 h의 $n \times n$ 퇴플리츠 행렬을 $[h]$로 표기할 때, h와 관련된 NTRU 모듈은

$$M_{NTRU} = \{(u,v) \in R^2 \mid u + vh = 0 \bmod q\}$$

이고, Z^{2n}에서의 NTRU 격자는

$$L_{NTRU} = \{(u,v) \in Z^{2n} \mid u + v[h] = 0 \bmod q\}$$

이며 $A_h = \begin{pmatrix} -[h] & I_n \\ qI_n & O_n \end{pmatrix}$와 $B_{f,g} = \begin{pmatrix} [g] & -[f] \\ [G] & -[F] \end{pmatrix}$를 기저로 갖는다[7,9].

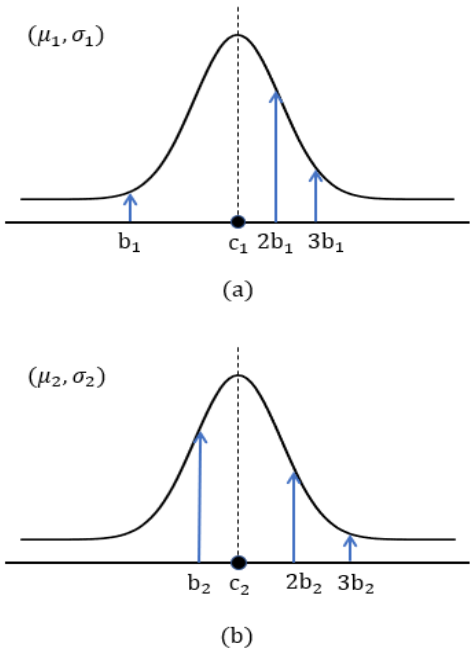
NTRU 격자는 GPV 프레임워크의 키 생성 단계에 적용될 수 있다[10]. Hash-and-sign 서명 기법에서 서명 생성 시 사용되는 공개키는 $A = (1\ h)$이고, 서명 검증 시 사용되는 비밀키는 $B = \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$이다. A와 B는 $B \times A^t = (g - hf, G - hF) = 0$를 만족하여 서로 직교하므로 트랩도어 쌍이 된다.

2.3. 트랩도어 샘플러

$$\text{Sampler} = \text{CVP Solver} + \text{Randomization}$$

GPV 프레임워크는 서명 생성 시 (그림 1 Sign 3번) 격자 위에 있는 점 v를 찾기 위해 트랩도어를 이용하여 근사 CVP(Approximate CVP)를 푼다. 근사 CVP를 푸는 격자 디코딩 알고리즘으로는 Babai의 nearest plane 알고리즘과 round off 알고리즘이 있다. 그러나 이 두 알고리즘은 결정적(deterministic)이라서 같은 메시지에 대한 서명이 여러 번 생성될 경우 개인키인 격자의 기저 B에 대한 정보가 노출될 수 있다[11]. 따라서 c를 중심으로 하는 랜덤화 분포에서 근사 CVP 문제의 답인 v를 샘플링 하여 결정적 알고리즘을 랜덤화한 확률적(probabilistic)인 트랩도어 샘플러를 사용한다.

그림 2는 결정적 알고리즘을 가우시안 분포에서 랜덤화하여 확률적으로 바꾸는 방법을 나타낸 예시이다. $B = (b_1, b_2, ..., b_w)$, $c = (c_1, c_2, ..., c_w)$일 때, 결정적 알고리즘일 경우 c_1은 항상 가장 가까운 $2b_1$으로, c_2는 항상 b_2로 샘플링된다. 그러나 이를 랜덤화 하면 c_1은



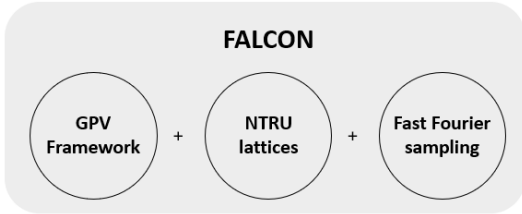
〔그림 2〕 가우시안 분포에서의 랜덤화 (a) b_1 기저에 대한 랜덤화 (b) b_2 기저에 대한 랜덤화

그림 2의 (a)와 같은 분포에서 b_1이나 $3b_1$으로도 샘플링될 수 있고, c_2는 (b)와 같은 분포에서 $2b_2$나 $3b_2$로도 샘플링될 수 있다. 가우시안 분포에서 랜덤화하는 이유는 랜덤화를 하여도 서명의 크기를 너무 크게 하지 않게 하기 위함이다.

대표적인 트랩도어 샘플러에는 Klein 샘플러와 Peikert 샘플러가 있다. Klein 샘플러는 Babai의 nearest plane 알고리즘을 가우시안 분포에서 랜덤화하고[12], Peikert 샘플러는 Babai의 round off 알고리즘을 가우시안 분포에서 랜덤화한다[13]. Peikert 샘플러는 순차적으로 샘플링 해야 하는 Klein 샘플러보다 속도가 빠르지만, 서명 길이가 더 길어 안전성은 더 낮다.

Ⅲ. FALCON

FALCON(Fast Fourier Lattice-based Compact Signatures over NTRU)은 GPV 프레임워크의 hash-and-sign 서명 구조를 바탕으로 설계되었으며, NTRU 격자를 기반으로 하여 키를 생성하고 fast fourier 트랩도어 샘플러를 이용하여 서명을 생성한다. FALCON은 $n=2^k$ (k는 양의 정수)인 원분체



〔그림 3〕 FALCON 서명 기법 요약

$K=Q[x]/(x^n+1)$에서 동작하며 NTRU 격자로 트랩도어 키 쌍을 생성하여 공개키의 크기가 작다.

Nearest plane 알고리즘은 CVP를 푸는 격자 디코딩 알고리즘으로 서명 길이가 짧고 안전성이 높지만, 구현이 복잡하고 속도가 느리다. 따라서 FALCON은 연산 속도 향상을 위해 fast fourier nearest plane 알고리즘을 가우시안 분포에서 랜덤화한 FFS(Fast Fourier Sampling)를 트랩도어 샘플러로 사용하며 tower-of-ring 구조를 이용한다. 그러나 FALCON은 여전히 빠르지 않고 부동소수점으로 FFT(Fast Fourier Transform) 연산을 수행하여 부채널 공격을 막기 어렵다.

Tower 구조를 이용하면 n을 차수로 하는 다항식 1개를 $n/2$을 차수로 하는 다항식 2개로 분할 할 수 있다. FALCON은 n차 다항식을 $2n$개의 다항식 환의 원소로 분해하여 RCDT(Reverse Cumulative Distribution Table)를 이용하여 가우시안 분포와 유사한 분포에서 정수를 샘플링하고 다시 합치는 과정을 수행한다.

FALCON을 포함한 hash-and-sign 격자 기반 서명의 안전성은 GPV 프레임워크의 안전성을 따라 SIS 문제의 어려움을 기반으로 한다. 따라서 서명의 길이가 짧을수록 안전한데, 이는 근사 CVP를 풀기 어려울수록 안전함을 의미한다. 근사 CVP를 푸는 방법은 샘플러에 따라 다르므로 hash-and-sign 서명은 샘플링 단계에서 효율성과 안전성의 트레이드 오프가 존재한다. 이에 따라 FFS을 이용하는 FALCON은 안전하지만 복잡하고 상대적으로 속도가 느리다.

3.1. 키 생성

FALCON의 키 생성 알고리즘은 NTRU 격자를 기반으로 키 쌍$(\mathrm{pk}, \mathrm{sk})$을 출력한다. 그림 4의 `NTRUGen` 알고리즘은 다항식 f와 g를 가우시안 분포에서 랜덤으로 생성한 후 NTRU 방정식으로 다항

Input: $\phi \in Z[x]$, q
Output: pk, sk

1. $f, g, F, G \leftarrow$ **NTRUGen**(ϕ, q)
2. $B \leftarrow \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$
3. $\hat{B} = FFT(B)$
4. $T \leftarrow$ **ffLDL**$(\hat{B} \times \hat{B}^*)$
5. $h = gf^{-1} \bmod q$
6. return pk $= h$, sk $= (\hat{B}, T)$

〔그림 4〕 FALCON 키 생성 알고리즘

식 F와 G를 계산한다. FALCON의 공개키 pk는 $h = gf^{-1} \bmod q$이고, 개인키 sk는 $B = \begin{pmatrix} g & -f \\ G & -F \end{pmatrix}$의 각 행렬 요소를 고속 푸리에 변환(FFT)한 $\hat{B} = FFT(B)$이다.

FALCON은 구현의 효율성을 위해 정규화된 트리 T도 개인키로 출력한다. 트리 T는 **ffLDL** 알고리즘으로 $\hat{B} \times \hat{B}^*$을 차수가 낮은 작은 환으로 분할하여 재귀적으로 연산을 수행하여 생성한다.

3.2. 서명

FALCON의 서명 알고리즘은 메시지 m과 개인키 sk, 바운드 γ를 입력받아 랜덤 r과 서명 s를 출력한다. 먼저 균일하게 생성한 r을 m과 이어붙여 해시 함수에 입력하여 c를 얻고, 격자 위에 있지 않은 점 $t = (FFT(c), FFT(0)) \cdot \hat{B}^{-1}$와 가까운 격자 위의 점 v를 **ffSampling** 알고리즘으로 찾는다. 서명 s는 두 점 t와 v의 차로, FALCON은 서명의 크기를 작게 하도록 $s_1 + s_2 h = c \bmod q$를 만족하는 s_2만 **Compress** 알고리즘에 입력해 압축하여 출력한다.

Input: m, sk, γ
Output: r, s

1. $r \leftarrow \{0,1\}^{320}$ uniformly
2. $c \leftarrow H(r||m)$
3. $t \leftarrow (FFT(c), FFT(0)) \cdot \hat{B}^{-1}$
4. **do**
5. $\quad v \leftarrow$ **ffSampling** (t, T)
6. $\quad s = (t - v)\hat{B}$
7. **while** $\| s \| > \gamma$
8. $(s_1, s_2) \leftarrow FFT^{-1}(s)$
9. $s \leftarrow$ **Compress**(s_2)
10. return r, s

〔그림 5〕 FALCON 서명 알고리즘

그림 5의 **ffSampling** 알고리즘은 연산 속도 향상을 위해 tower-of-ring 구조를 이용하여 샘플링 하는 FFS이다. Ducas와 Prest[14]가 제안한 fast fourier nearest plane은 서명의 길이가 짧은 Klein 샘플러와 속도가 빠른 Peikert 샘플러의 장점을 모두 가지고 있어 Falcon은 이를 랜덤화한 FFS를 트랩도어 샘플러로 이용한다.

3.3. 서명 검증

FALCON의 서명 검증 알고리즘은 메시지 m, 랜덤 값 r, 서명 s, 공개키 pk, 그리고 바운드 γ을 입력받아 accept 또는 reject를 출력한다. 먼저 r을 m과 이어붙여 해시 함수에 입력하여 c를 얻고, **Decompress** 알고리즘으로 압축을 풀어 $s = \| (s_1, s_2) \|$가 γ보다 같거나 작으면 accept을, 그렇지 않으면 reject을 출력한다.

Input: m, r, s, pk, γ
Output: Accept or reject

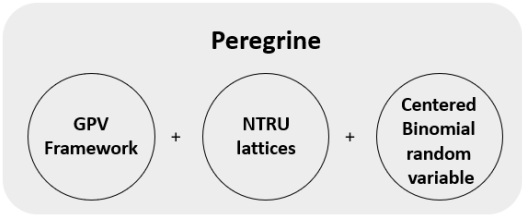
1. $c \leftarrow H(r||m)$
2. $s_2 \leftarrow$ **Decompress**(s)
3. $s_1 \leftarrow c - s_2 h \bmod q$
4. **if** $\| (s_1, s_2) \| \leq \gamma$ **then**
5. $\quad$ return accept
6. **else**
7. $\quad$ return reject

〔그림 6〕 FALCON 서명 검증 알고리즘

Ⅳ. Peregrine

Peregrine은 FALCON과 같이 NTRU 격자를 기반으로 하는 hash-and-sign 서명이지만 샘플링 방식이 다르다. Peregrine은 round off 격자 디코딩 알고리즘을 이항분포에서 랜덤화하여 정수 도메인에서 NTT 연산을 수행한다.

Round off 알고리즘은 nearest plane 알고리즘보다 구현이 간단하고 빠르나 서명 길이가 더 길고 안전성이 낮다. 이항분포는 가우시안 분포보다 더 간단하여 서명을 빠르게 할 수 있다. 따라서 Peregrine은 FALCON보다 간단하고 속도가 더 빠르며, 서명 생성



〔그림 7〕 Peregrine 서명 기법 요약

시 FALCON과 동일한 바운드를 설정하여 짧은 서명의 크기도 유지한다.

그러나 샘플링 시 이항분포를 사용하는 것은 개인키를 보호하기에 충분하지 않다는 단점이 있다. FALCON에서 사용하는 가우시안 분포의 경우 그림 2와 같이 기저마다 다른 분산을 가지는 분포에서 랜덤화하지만 이항분포의 경우 모든 기저가 같은 분포를 가지는 랜덤 변수로 랜덤화하므로 결국 개인키에 완전히 독립적이지 않기 때문이다.

FALCON은 기저마다 랜덤 변수 분포가 엄밀하게 계산된 가우시안 분포를 사용하지만, Peregrine은 엄밀한 계산이 필요 없는 이항분포를 사용한다. 따라서 Peregrine은 서명 단계에서 엄밀한 연산을 위한 소수점 연산을 하지 않는다.

4.1. 키 생성

Peregrine의 키 생성 알고리즘은 NTRU 격자를 기반으로 키 쌍$(\mathrm{pk},\mathrm{sk})$을 출력한다. 그림 8의 **PolyGen** 알고리즘은 다항식 f와 g를 이항분포에서 랜덤으로 생성한 후 NTRU 방정식으로 다항식 F와 G를 계산한다. Peregrine은 FALCON에서 공개키를 생성할 때 사용하는 가우시안 분포와 동일한 분포를 가지는 이항분포를 사용한다.

Input: $\phi \in Z[x]$, q
Output: pk, sk

1. $f, g, F, G \leftarrow$ **PolyGen**(ϕ, q)
2. $\mathrm{B} \leftarrow \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$
3. $h = gf^{-1} \bmod q$
4. return pk$=h$, sk$=$B

〔그림 8〕 Peregrine 키 생성 알고리즘

4.2. 서명

Peregrine의 서명 알고리즘은 메시지 m과 개인키 sk, 바운드 γ를 입력받아 랜덤 r과 서명 s를 출력한다. 먼저 균일하게 생성한 r을 m과 이어붙여 해시 함수에 입력하여 c를 얻고, 격자 위에 있지 않은 점 $\mathrm{t} = (\mathrm{c},0) \cdot \mathrm{B}^{-1} = (\mathrm{t}_1,\mathrm{t}_2) = (-\frac{\mathrm{c}\cdot F}{q}, \frac{\mathrm{c}\cdot f}{q})$와 가까운 격자 위의 점 v를 **ModDown** 알고리즘과 **Rand** 알고리즘으로 찾는다. Peregrine은 FALCON과 동일한 압축 알고리즘인 **Compress** 알고리즘으로 서명을 압축한다.

그림 9의 **ModDown** 알고리즘은 round off 알고리즘을 이용하여 격자 위에 있지 않은 점 t의 정수 부분을 찾는다. 즉 R이 소수$(-0.5 < R \le 0.5)$이고 I는 정수일 때, $\mathrm{t} = (\mathrm{t}_1,\mathrm{t}_2) = (R_1 + I_1, R_2 + I_2)$에서 I_1, I_2를 출력한다. **Rand** 알고리즘은 I_1, I_2에 이항분포에서 랜덤화한 랜덤 변수 J_1, J_2를 더하여 t와 가까운 $\mathrm{v} = (I_1 + J_1, I_2 + J_2)$를 출력한다. 이때 J는 이항분포에서 샘플링한 작은 계수들이다. 그림 9의 11번째 줄 서명 s는 $(\mathrm{t}-\mathrm{v}) \cdot \mathrm{B}$이고, 서명의 길이가 바운드보다 작으면 s_2만 압축하여 출력한다.

Input: m, sk, γ
Output: r, s

1. $\mathrm{r} \leftarrow \{0,1\}^{320}$ uniformly
2. $\mathrm{c} \leftarrow \mathrm{H}(\mathrm{r}||\mathrm{m})$
3. $\mu \leftarrow 26$
4. $I_1 \leftarrow$ **ModDown**$(-c \cdot F, Q, q)$
5. $I_2 \leftarrow$ **ModDown**$(c \cdot f, Q, q)$
6. **do**
7. $\quad J_1 \leftarrow$ **Rand**(c, I_1)
8. $\quad J_2 \leftarrow$ **Rand**(c, I_2)
9. $\quad \mathrm{s}_1 \leftarrow \mathrm{c} - (I_1 + J_1)g - (I_2 + J_2)G \bmod q$
10. $\quad \mathrm{s}_2 \leftarrow (I_1 + J_1)f + (I_2 + J_2)F \bmod q$
11. $\quad \mathrm{s} \leftarrow (\mathrm{s}_1, \mathrm{s}_2)$
12. **while** $\| s \| > \gamma$
13. s $\leftarrow$ **Compress**(s_2)
14. return r,s

〔그림 9〕 Peregrine 서명 알고리즘

4.3. 서명 검증

Peregrine의 서명 검증 알고리즘은 FALCON의 서명 검증 알고리즘(그림 6)과 같다. 메시지 m, 랜덤 값 r, 서명 s, 공개키 pk, 그리고 바운드 γ을 입력받아 accept 또는 reject를 출력한다.

4.4. 안전성

서명에 대한 공격으로는 키 복구(key recovery) 공격과 위조(forgery) 두 가지가 있다. 격자 기반 서명에서 키 복구 공격은 개인키를 바로 찾아내는 SVP를 푸는 공격이며, 위조는 크기가 작은 서명을 위조하기 위해 주어진 점과 가까운 격자 위의 점을 찾아내는 CVP를 푸는 공격이다. 표 3은 두 공격에 대한 Peregrine의 안전성 비트(classical/quantum)와 NIST 레벨이다.

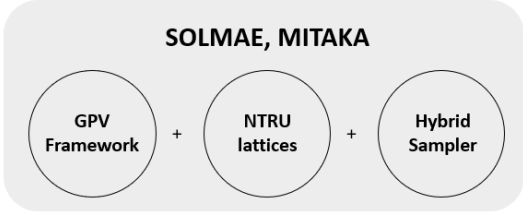
Peregrine은 샘플링 시 가우시안 분포가 아닌 이항분포로 랜덤화를 한다. 그러나 이항분포로 랜덤화 하면 모든 기저가 같은 분포를 가지는 랜덤 변수로 랜덤화되어 결국 격자의 기저인 개인키를 보호하기에 안전하지 않다.

〔표 3〕 Peregrine의 안전성

	Peregrine-512	Peregrine-1024
Key recovery attack(C/Q)	133/121	273/248
Forgery attack(C/Q)	120/108	248/222
NIST level	I	V

Ⅴ. SOLMAE

SOLMAE는 FALCON과 MITAKA의 장점을 합친 NTRU 격자 기반 hash-and-sign 서명이다. FALCON은 키와 서명 길이가 짧고 안전성이 높지만, 속도가 빠르지 않고 파라미터 선택이 유연하지 않으며 구현이 복잡하고 부채널 공격을 막기 어렵다. MITAKA는 하이브리드 샘플러[15]를 이용하여 속도가 빠르고 파라미터 선택이 유연하며 병렬성으로 구현이 쉽지만, FALCON과 비교하여 서명 길이가 더 길고 안전성이 낮다.



〔그림 10〕 SOLMAE, MITAKA 서명 기법 요약

SOLMAE는 MITAKA와 같이 하이브리드 샘플러를 이용하여 빠른 속도, 파라미터 선택의 유연성, 병렬성, 간단함의 이점을 가져오면서 최적화된 키 생성 알고리즘을 제안하여 MITAKA보다 키 생성 시간은 더 줄이고 안전성은 더 높인다. 그러나 여전히 SOLMAE-1024는 FALCON-1024보다 서명 길이가 더 길고 안전성이 더 낮다.

5.1. 키 생성

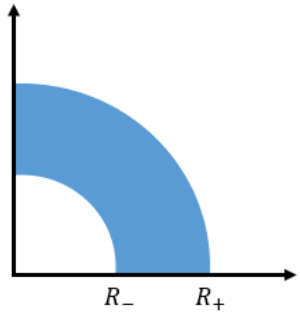
SOLMAE의 키 생성 알고리즘은 모듈러 q, 퀄리티 α, 그리고 두 반지름 $0 < R_- < R_+$을 입력받아 키 쌍 $(\mathrm{pk}, \mathrm{sk})$을 출력한다. NTRU 기저의 퀄리티 α는 안전성과 관련 있으며 트랩도어의 샘플러에 따라 그 값이 다르다. FALCON에서 사용하는 fast fourier 샘플러가 다른 Peikert 샘플러나 하이브리드 샘플러보다 안전한 이유는 α의 크기가 더 작기 때문이다.

SOLMAE는 하이브리드 샘플러를 사용하지만, 안전성 향상을 위해 FALCON과 같은 퀄리티 값을 사용하여 키 생성 시 제한을 둔다. 다시 말해 FALCON은 (f,g)를 랜덤으로 생성한 후 가능한(퀄리티가 좋은) NTRU 쌍을 찾는데, 퀄리티가 낮으면 (f,g)를 다시 생성한다. SOLMAE는 사전에 설정한 범위에서 가능한 (f,g)을 한 번에 생성하여 키를 생성하는 과정을 다시 반복할 필요가 없다. 그림 11의 **PairGen** 알고리즘은 트랩도어 퀄리티가 FALCON에서 사용하는 퀄

Input: $q,\ \alpha,\ R_-,\ R_+$
Output: pk, sk

1. $\mathrm{b}_1 = (f, g) \leftarrow$ **PairGen**(q, α, R_-, R_+)
2. $\mathrm{b}_2 = (F, G) \leftarrow$ **NtruSolve**(q, f, g)
3. $\widetilde{\mathrm{b}_2} = (\widetilde{F}, \widetilde{G}), \beta_1, \beta_2 \leftarrow$ **Precomputation**$(\mathrm{b}_1, \mathrm{b}_2)$
6. return $\mathrm{pk} = h,\ \mathrm{sk} = (\mathrm{b}_1, \mathrm{b}_2, \widetilde{\mathrm{b}_2})$

〔그림 11〕 SOLMAE 키 생성 알고리즘



〔그림 12〕 PairGen 샘플링 방법

리티 범위와 같도록 (f,g)를 샘플링한다. 그림 12는 ψ가 이산 푸리에 변환이고 δ는 작은 정정 파라미터일 때, $|\psi_i(f)|$와 $|\psi_i(g)|$를 샘플링 하는 방법이다. 퀄리티 α의 범위는 $\frac{q}{\alpha^2} \le |\psi_i(f)|^2 + |\psi_i(g)|^2 \le \alpha^2 q$ 이므로 $R_- = (\frac{1}{\alpha} + \delta)\sqrt{q}$, $R_+ = (\alpha - \delta)\sqrt{q}$이며, 이때 α는 $\alpha_{512} = 1.17$, $\alpha_{1024} = 1.64$이다.

그림 11의 **NTRUSolve** 알고리즘은 f,g가 주어졌을 때 NTRU 연산으로 F,G를 구한다. **Precomputation** 알고리즘은 효율성을 위해 푸리에 도메인에서 샘플링 시 필요한 데이터를 미리 계산한다.

5.2. 서명

SOLMAE의 서명 알고리즘은 메시지 m과 개인키 sk, 바운드 γ을 입력받아 랜덤 값 r과 서명 s를 출력한다. 먼저 균일하게 생성한 r을 m과 이어붙여 해시 함수에 입력하여 c를 얻고, 격자 위에 있지 않은 점 c와 가까운 격자 위의 점 v를 **HybridSample** 알고리즘으로 찾는다. 그 후 서명을 압축하는 과정은 FALCON과 동일하다. **HybridSample** 알고리즘은 MITAKA에서 사용하는 하이브리드 샘플러로, 전반적인 동작 원리는 Klein 샘플러를 따르지만, 그 과정에서 Peikert 샘플러가 서브 루틴으로 사용된다.

그림 14는 **HybridSample**을 자세히 나타낸 것으로, $\beta_1, \beta_2 \in K_R^2$는 각각 K_R^2에서 $K_R \cdot \mathrm{b}_1$과 $K_R \cdot \widetilde{\mathrm{b}}_2$로 정사영 시킨 벡터들이다. **PeikertSampler**는 푸리에 도메인에서 다루는 N 샘플러와 정수를 다루는 Z 샘플러 두 단계를 이용하여 샘플링 한다.

Input: m, sk, $\lfloor \beta^2 \rfloor$
Output: r, s

1. $\mathrm{r} \leftarrow \{0,1\}^{320}$ uniformly
2. $\mathrm{c} \leftarrow (0, \mathrm{H}(\mathrm{r}||\mathrm{m}))$
3. $\hat{\mathrm{c}} \leftarrow \mathrm{FFT}(\mathrm{c})$
4. **do**
5. $\quad \mathrm{v} \leftarrow$ **HybridSample**$(\hat{\mathrm{c}}, \mathrm{sk})$
6. $\quad \mathrm{s} \leftarrow \hat{\mathrm{c}} - \mathrm{v}$
7. **while** $\| s \|^2 > \lfloor \beta^2 \rfloor$
8. $(\mathrm{s}_1, \mathrm{s}_2) \leftarrow \mathrm{FFT}^{-1}(\mathrm{s})$
9. $\mathrm{s} \leftarrow$ **Compress**(s_2)
10. return r,s

〔그림 13〕 SOLMAE 서명 알고리즘

Input: $\mathrm{c} = (0, \mathrm{c}')$, sk
Output: A vector **v** close to c

1. $\mathrm{t} \leftarrow \mathrm{c}, \mathrm{v} \leftarrow 0$
2. **for** $\mathrm{i} = 2$ to 1 **do**
3. $\quad \mathrm{t}_\mathrm{i} \leftarrow \langle \beta_\mathrm{i}, \mathrm{t} \rangle$
4. $\quad \mathrm{z}_\mathrm{i} \leftarrow$ **PeikertSampler**(t_i)
5. $\quad \mathrm{t} \leftarrow \mathrm{t} - \mathrm{z}_\mathrm{i}\mathrm{b}_\mathrm{i}, \mathrm{v} \leftarrow \mathrm{v} + \mathrm{z}_\mathrm{i}\mathrm{b}_\mathrm{i}$
6. **end**
7. return v

〔그림 14〕 HybridSample 알고리즘

MITAKA는 N 샘플러와 Z 샘플러에서 모두 구형 가우시안 분포를 이용했지만 SOLMAE는 Z 샘플러에서는 구형 가우시안 분포를, N 샘플러에서는 타원형 가우시안 분포를 이용하였다. 이는 SOLMAE가 추가로 적용하려는 T. Espitau et al.[3]의 압축 기술이 타원형 가우시안 분포에서 동작하기 때문에 호환성을 맞추기 위함이다.

5.3. 서명 검증

SOLMAE의 서명 검증 알고리즘은 FALCON의 서명 검증 알고리즘(그림 6)과 같다. 메시지 m, 랜덤 값 r, 서명 s, 공개키 pk, 그리고 바운드 γ을 입력받아 accept 또는 reject를 출력한다.

5.4. 안전성

표 4는 core-SVP 방법론[16]에 의해 계산한 SOLMAE의 위조에 대한 안전성 비트와 NIST 레벨이

〔표 4〕 SOLMAE의 안전성

	SOLMAE-512	SOLMAE-1024
Forgery attack(C/Q)	127/115	256/232
NIST level	I	V

다. SOLMAE는 FALCON과 동일한 방식으로 위조 공격에 대한 안전성 비트를 계산하였다.

Ⅵ. 비교 및 분석

6.1. 비교

6.1.1 키, 서명 크기

표 5와 6은 차수 n이 각각 512, 1024일 때 FALCON, Peregrine, 그리고 SOLMAE의 공개키와 서명 크기를 비교한 표이다. 공개키에는 차수와 모듈러를 나타내는 1바이트 헤더가 포함되어 있는데, 아래 두 표는 헤더를 포함한 크기이다. 차수 n이 512일 때는 공개키와 서명의 크기가 세 개의 기법에서 모두 동일하지만, n이 1024일 때는 SOLMAE의 서명 크기가 다른 두 기법보다 조금 더 크다.

〔표 5〕 $n=512$일 때 키, 서명 크기 비교 (단위:bytes)

$n=512$	FALCON	Peregrine	SOLMAE
Public key	897	897	897
Signature	666	666	666

〔표 6〕 $n=1024$일 때 키, 서명 크기 비교 (단위:bytes)

$n=1024$	FALCON	Peregrine	SOLMAE
Public key	1793	1793	1793
Signature	1280	1280	1375

6.1.2 속도

표 7과 8은 Intel(R) Core(TM) i5-3.40GHz 16.0GB RAM에서 차수 n이 각각 512, 1024일 때 FALCON, Peregrine, 그리고 SOLMAE의 키 생성, 서명, 서명 검증 시간을 비교한 표이다. 각 알고리즘을 200번 수행하였을 때의 평균적인 성능으로, 키 생성과 서명 알고리즘이 Peregrine, SOLMAE, FALCON 순으로 빠르다.

〔표 7〕 $n=512$일 때 시간 비교 (단위:kcycles)

$n=512$	FALCON	Peregrine	SOLMAE
KeyGen	59,119	18,834	42,707
Sign	16,552	420	591
Verify	122	45	131

〔표 8〕 $n=1024$일 때 시간 비교 (단위:kcycles)

$n=512$	FALCON	Peregrine	SOLMAE
KeyGen	169,947	65,719	83,817
Sign	36,768	900	1,130
Verify	262	94	266

6.1.3 안전성

표 9와 10은 차수 n이 각각 512, 1024일 때 FALCON, Peregrine, 그리고 SOLMAE의 키 복구 공격과 위조의 어려움에 대한 안전성 비트를 비교한 표이다. SOLMAE는 키 복구 공격에 대한 안전성 비트를 제공하지 않았다. 차수 n이 512일 때 FALCON과 Peregrine의 안전성은 동일하며, SOLMAE가 위조에 대한 안전성은 조금 더 높다. 차수 n이 1024일 때 키

〔표 9〕 $n=512$일 때 안전성 비교

$n=512$	FALCON	Peregrine	SOLMAE
Key recovery attack(C/Q)	133/121	133/121	-
Forgery attack(C/Q)	120/108	120/108	127/115

〔표 10〕 $n=1024$일 때 안전성 비교

$n=1024$	FALCON	Peregrine	SOLMAE
Key recovery attack(C/Q)	273/248	273/248	-
Forgery attack(C/Q)	277/252	248/222	256/232

복구 공격에 대한 안전성은 FALCON과 Peregrine이 동일하나, 위조에 대한 안전성은 FALCON, SOLMAE, Peregrine 순으로 높다. 세 가지 기법의 NIST 안전성 레벨은 n이 512, 1024일 때 각각 I, V로 모두 같다.

6.1.4 샘플러

FALCON, Peregrine, 그리고 SOLMAE는 NTRU 격자를 기반으로 하는 hash-and-sign 구조의 서명으로 전반적인 프레임워크는 같지만, 샘플링 방식이 다르다. 표 11은 트랩도어 샘플러를 비교한 표로, FALCON은 fast fourier nearest plane 알고리즘을 가우시안 분포에서 랜덤화하고 Peregrine은 round off 알고리즘을 이항분포에서 랜덤화 한다. SOLMAE는 두 격자 디코딩 알고리즘을 모두 사용하는 하이브리드 샘플러를 가우시안 분포에서 랜덤화한다.

FALCON은 트랩도어 샘플러로 FFS를 이용하여 속도가 빠르지 않으며 부채널 공격에 취약하다. Peregrine과 SOLMAE는 FALCON과 다른 트랩도어 샘플러를 사용하여 속도를 개선하였으며, 부채널 공격에 대해 더 안전하다.

〔표 11〕 샘플러 비교 (r.v.: random variable)

Sampler	CVP solver	Randomization
Klein	Nearest plane algorithm	Discrete Gaussian r.v.
Peikert	Round off algorithm	Discrete Gaussian r.v.
FALCON (fast-fourier)	Fast fourier nearest plane algorithm	Discrete Gaussian r.v.
Peregrine	Round off algorithm	Centered Binomial r.v.
SOLMAE	Hybrid algorithm (nearest plane +round off)	Discrete Gaussian r.v.

6.2. 분석

6.2.1 Peregrine

Peregrine은 FALCON의 설계 흐름을 따라가며 각 기술 요소를 상황에 맞게 바꿔보았다는 점에서 그 의의가 있다. Peregrine은 구현이 복잡하고 시간이 오래 걸리며 소수점 연산을 해야 하는 FALCON의 단점을 가장 원초적인 방법으로 극복하고자 하였다.

FALCON은 샘플링 단계에서 CVP를 풀기 위해 nearest plane 격자 디코딩 알고리즘을 사용하여 복잡하고 시간이 오래 걸린다. Peregrine은 이를 극복하기 위해 단순히 다른 격자 디코딩 알고리즘인 round off 알고리즘을 적용하였다. 하지만 round off 알고리즘을 적용하면 서명의 분포가 기울어져서 랜덤화를 하기 위해 가우시안 분포를 두 번 사용하여 서명의 분포를 균형적으로 맞춰줘야 한다[13]. Peregrine은 FALCON의 효율성을 개선하고자 시도한 연구이기 때문에, 가우시안 분포를 두 번 사용하는 Peikert 샘플러 대신 랜덤화를 위해 이항분포를 사용하였다. 이항분포를 사용함으로 인해 Peregrine은 구현이 간단하고 속도가 빠르다는 장점이 있다.

한편 이항분포에서 랜덤화 하면 생기는 치명적인 문제점도 있다. FALCON은 각 기저에 대해 다른 분포로 랜덤화 하였기 때문에 개인키인 격자 B에 대해 독립적이지만, Peregrine은 각 기저에 대해 같은 분포로 랜덤화 하여 개인키에 대해 독립적이지 않다. 랜덤화의 목적은 결정적인 알고리즘을 확률적인 알고리즘으로 바꾸어 개인키를 숨기기 위함인데, 이항분포로 랜덤화를 하면 개인키를 온전히 숨기지 못하므로 Peregrine은 안전성 측면에서 문제가 있다.

또한, 격자 디코딩 알고리즘으로 round off 알고리즘을 사용하는 Peregrine-1024는 nearest plane 알고리즘을 사용하는 FALCON-1024보다 위조 공격에 대한 안전성 비트가 더 낮다. Peregrine은 아직 기법의 안전성을 이론적으로 증명하지 않은 상태이다.

결론적으로 Peregrine은 FALCON의 효율성을 개선하였으나, 이항분포를 사용함으로써 개인키를 온전히 숨기지 못하는 문제점이 있다. 따라서 이항분포를 사용하여도 기저마다 다른 랜덤화 분포를 만드는 추가적인 연구가 필요할 것으로 예상한다.

6.2.2 SOLMAE

SOLMAE는 FALCON과 FALCON의 후속 연구인 MITAKA의 장점을 합쳐보고자 한 점에서 그 의의가 있다. MITAKA는 FALCON보다 빠르고 구현이 간단

하며 병렬성과 유연성이 있지만, FALCON과 비교하였을 때 안전성이 낮고 서명 길이가 길다.

SOLMAE는 새로운 최적화 키 생성 알고리즘을 제안하여 FALCON보다 안전성이 낮은 MITAKA의 안전성을 향상하고, 키 생성 속도도 더 빠르게 하였다. 그 결과 SOLMAE-1024는 MITAKA-1024보다 안전성이 높지만, FALCON-1024보다는 여전히 낮다.

또한, SOLMAE는 MITAKA의 단점 중 하나인 FALCON보다 긴 서명 길이를 줄이기 위해 T. Espitau et al.[3]의 압축 기술을 사용하고자 하였다. SOLMAE는 압축 기술을 적용하기 위해 샘플러의 일부분을 타원형 분포로 바꾸었지만, 아직 압축 기술을 실험에 적용하지는 않은 상태이다.

한편 SOLMAE는 압축 기술을 적용하였을 때 생길 수 있는 문제에 대해서는 고려하지 않았다. T. Espitau et al.는 FALCON과 MITAKA에 각각 압축 기술을 적용하였을 때 왜곡 파라미터 Υ의 크기에 따른 안전성 비트를 계산하였으며, Υ가 커질수록 안전성이 감소한 것을 보였다[3]. SOLMAE는 MITAKA를 변형한 기법이므로 압축 기술을 적용하였을 때 마찬가지로 안전성 비트가 줄어들 것으로 예상하여 이에 대한 추가적인 분석이 필요할 것으로 보인다.

Ⅶ. 결 론

본 논문에서는 KpqC 공모전에 제출된 Peregrine과 SOLMAE를 분석하고 NIST PQC 표준 후보로 최종 선정된 FALCON과 비교하였다. Peregrine, SOLMAE, 그리고 FALCON은 NTRU 격자 기반 hash-and-sign 서명으로 구조는 같으나 트랩도어 샘플러 방법이 달라 효율성과 안전성 측면에서 장단점이 극명하다. 본 논문은 Peregrine과 SOLMAE 기법에 추가하거나 보완해야 할 부분에 대해 제시하며 향후 더 구체적인 안전성 분석 및 검증을 기대한다.

참 고 문 헌

[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, November 1994.

[2] P. A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, “FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU,” *Submission to the NIST’s post-quantum cryptography standardization process*, 35(5), 2018.

[3] T. Espitau, M. Tibouchi, A. Wallet, and Y. Yu, “Shorter hash-and-sign lattice-based signatures,” *Advances in Cryptology-CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II.* Cham: Springer Nature Switzerland, pp.245-275.

[4] M. Plançon, T. Prest, ”Exact Lattice Sampling from Non-Gaussian Distributions,“ *Public-Key Cryptography-PKC 2021: 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I.* Cham: Springer International Publishing, pp. 573-595.

[5] T. Espitau, P. A. Fouque, F. Gérard, M. Rossi, A. Takahashi, M. Tibouchi, A. Wallet, and Y. Yu, “MITAKA: A Simpler, Parallelizable, Maskable Variant of FALCON,” *Advances in Cryptology-EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30-June 3, 2022, Proceedings, Part III.* Cham: Springer International Publishing, pp. 222-253.

[6] E. Y. Seo, Y. S. Kim, J. W. Lee, J. S. No, “Peregrine: Toward Fastest FALCON Based on GPV Framework,” *Cryptology ePrint Archive*, 2022.

[7] K. Kim, M. Tibouchi, A. Wallet, T. Espitau, A. Takahashi, Y. Yu, and S. Guilley, “SOLMAE,” *Submission to the Korea post-quantum cryptography standardization process*, 2022.

[8] C. Gentry, C. Peikert, and V. Vaikuntanathan, “Trapdoors for Hard Lattices and New Cryptographic Constructions,” *Proceedings of the fortieth annual ACM symposium on Theory of*

computing, pp.197-206, May 2008.

[9] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient Identity-Based Encryption over NTRU Lattices," Advances in Cryptology-ASIACRYPT 2014: 20th International *Conference on the Theory and Applications of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20,* Springer Berlin Heidelberg, pp. 22-41.

[10] D. Stehlé, R. Steinfeld, "Making NTRU as Secure as Worst-Case Problems over Ideal Lattices," *Advances in Cryptology-EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011, Proceedings 30,* Springer Berlin Heidelberg, pp. 27-47.

[11] P. Q. Nguyen, and O. Regev, "Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures," *Journal of Cryptology,* 22(2), pp. 139-160, 2009.

[12] P. Klein, "Finding the closest lattice vector when it's unusually close," *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms,* pp. 937-941, February 2000.

[13] C. Peikert, "An Efficient and Parallel Gaussian Sampler for Lattices," *Advances in Cryptology-CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010, Proceedings 30,* Springer Berlin Heidelberg, pp. 80-97.

[14] L. Ducas, T. Prest, "Fast Fourier orthogonalization," *Proceedings of the ACM on International Symposium on Symbolic and aAlgebraic Computation*, pp. 191-198, July 2016.

[15] J. Folláth, "Gaussian sampling in lattice-based cryptography," *Tatra Mountains Mathematical Publications,* 60(1), pp. 1-23, 2014.

[16] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange-A New Hope," *USENIX security symposium*, Vol. 2016, August 2016.

〈저 자 소 개〉

김 주 언 (Juon Kim)

2022년 2월 : 이화여자대학교 사이버보안학과 졸업
2022년 3월~현재 : 고려대학교 정보보호학과 석사과정
<관심분야> 암호 알고리즘, 생체인증 보안, 양자내성암호

박 종 환 (Jong Hwan Park)

1999년 2월 : 고려대학교 수학과 졸업
2005년 2월 : 고려대학교 정보보호학과 석사
2008년 8월 : 고려대학교 정보보호학과 박사
2013년 9월~2019년 8월 : 상명대학교 컴퓨터과학과 조교수
2019년 9월~현재 : 상명대학교 컴퓨터과학과 부교수
<관심분야> 함수 암호, 양자내성암호, 영지식 증명