정 보 보 호 학 회 지
제 33 권 제3호, 2023. 6

KpqC 공모전 1라운드 암호 (그래프/다변수/아이소제니/영지식) 암호 소개

석 병 진*, 엄 혜 진**, 조 민 정**, 이 창 훈***

요 약

현재 국내에서는 양자내성암호 자체 기술 확보를 위해 양자내성암호 공모전인 KpqC 공모전이 진행되고 있다. KpqC 공모전을 통해 제안된 암호 알고리즘은 총 16종으로 격자 기반 - 8종, 코드 기반-4종, 그래프 기반-1종, 다변수 기반-1종, 아이소제니 기반-1종, 영지식 기반-1종과 같이 구성되어 있으며, 1라운드를 통해 안전성 및 효율성에 대한 평가가 수행되고 있다. 본 논문에서는 KpqC 공모전 1라운드 암호 중 그래프, 다변수, 영지식, 아이소제니 기반 암호 알고리즘를 소개하고 설계원리, 동작과정과 안전성 및 효율성에 대하여 살펴보고자 한다.

Ⅰ. 서 론

현대 공개키 암호시스템에 근간이 되는 소인수분해와 이산대수 문제의 어려움은 양자컴퓨터를 통해 다항시간 내에 효과적으로 해소될 수 있음에 따라, 세계적으로 양자내성암호 개발에 대한 연구가 활발히 수행되고 있다. 이와 관련하여 NIST에서는 2017년부터 양자내성암호 공모사업[1](NIST PQC Standardization)를 수행하였으며 지난 2022년 7월 최종적으로 4종의 암호를 선정하였다. 국내에서도 이런 흐름에 발맞추어 양자내성암호 기술을 자체적으로 확보하기 위하여 2022년부터 양자내성암호 공모 사업 KpqC 공모전[2] 개최를 통해 양자내성암호 개발을 수행하고 있다. KpqC 공모전은 현재 1라운드를 통해 제안된 16개 후보 알고리즘들(격자 기반 - 8종, 코드 기반-4종, 그래프 기반-1종, 다변수 기반-1종, 아이소제니 기반-1종, 영지식 기반-1종)에 대하여 안전성 및 효율성 평가를 수행하고 있다. 본 논문에서는 KpqC 공모전 1라운드의 후보 알고리즘 중 그래프, 다변수, 아이소제니, 영지식, 그래프 기반 알고리즘에 대해 소개하고 설계원리, 동작과정, 안정성 및 효율성에 대해 살펴보고자 한다.

Ⅱ. KpqC 1라운드 후보 알고리즘 소개 (그래프/다변수/아이소제니/영지식)

2.1. 그래프 기반 공개키암호 - IPCC

IPCC[3]는 Improved Perfect Code Cryptosystem의 약자로 그래프 기반 양자내성암호이다. IPCC의 안전성은 그래프에서 PDS(Perfect Domination Set)의 존재 여부를 결정하는 NP-Complete 문제를 기반으로 제공된다. 일반적으로 PCC(Perfect Code Cryptosystem)[4] 기반 암호 알고리즘은 효율성과 안전성을 동시에 제공하기 힘들다. 효율성 개선을 위해 PCC 문제의 파라미터를 조정하는 경우 안전성이 감소되기 때문이다. PCC에서 암호문 다항식의 최대 차수 k가 작으면 평문 복구 공격 가능성을 높이고 반대로 크면 효율성을 낮춘다. IPCC는 그래프를 여러개 사용함으로써 동일한 값의 k에 대해 암호화에서 효율적인 연산을 할 수 있도록 설계했다. 단, 키 생성과 복호화 과정은 기존 PCC 설계를 차용하였다.

IPCC는 3-regular graph 상에서 PDS를 결정짓는 문제를 기반으로 한다. k-regular graph란 임의의 양의 정수 k가 주어졌을 때, 그래프에 속한 모든 정점 v의 차수가 k인 모양을 의미한다. 즉, 3-regular graph는

본 연구는 국가보안기술연구소 위탁과제(KpqC 공모전 알고리즘 비교 분석 연구, 2023-116) 지원으로 수행되었습니다.
* 서울과학기술대학교 전기정보기술연구소 (선임연구원, bjseok@seoultech.ac.kr)
** 서울과학기술대학교 컴퓨터공학과 (대학원생, eomhj@seoultech.ac.kr, chomj@seoultech.ac.kr)
*** 서울과학기술대학교 컴퓨터공학과 (교수, chlee@seoultech.ac.kr, 교신저자)

주어진 그래프에 속한 모든 정점의 차수가 3이다. PDS는 정점 집합 V의 하위집합을 D라고 할 때, 각 정점에서 연결된 정점 집합이 D의 원소 중 한 개를 포함하면 집합 D를 PDS라고 한다. PDF(Perfect Domination Function)는 각 정점에서 간선으로 연결된 정점들의 합이 모두 1일 때, 그래프에서 정점의 집합 V를 0 또는 1로 사상시키는 함수 f이다. 이때 PDS에 속하는 정점의 값을 1로 두고 나머지 정점의 값을 0으로 둔다.

IPCC는 양자내성 공개키 암호로 키 생성 함수, 암호화 함수, 복호화 함수로 구성된다. 키 생성 함수는 각 그래프에 대한 정점 집합들을 입력으로 하며, 공개키와 개인키를 생성한다. 세부적인 키 생성 알고리즘은 [표 1]과 같다. 여기서, $a \xleftarrow{\$} A$는 집합 B에서 한 개의 원소 a를 랜덤하게 선택함을 의미하고, $B \xleftarrow[i]{\$} A$는 집합 A에서 랜덤하게 i개의 원소를 뽑아 하위집합 B를 구성함을 의미한다.

〔표 1〕 IPCC 키 생성 알고리즘

Require: the number of vertices n_1, n_2, ... for each graph
Ensure: Public Key pk, Private Key sk

for $i = 1,2,...$ **do**
  $V_i \leftarrow 1,2,...,n_i$
  $E_i \leftarrow \varnothing$
  for $j = 1,2,3,4$ **do**
    $D_{ij} \xleftarrow[n_{i/4}]{\$} V_i - \sum_{k=1}^{j-1} D_{ik}$
  end for
  $E \leftarrow D_{i1} \xleftrightarrow{\$} D_{i2} \cup E,\ E \leftarrow D_{i1} \xleftrightarrow{\$} D_{i3} \cup E,$
  $E \leftarrow D_{i1} \xleftrightarrow{\$} D_{i4} \cup E,\ E \leftarrow D_{i2} \xleftrightarrow{\$} D_{i3} \cup E,$
  $E \leftarrow D_{i2} \xleftrightarrow{\$} D_{i4} \cup E,\ E \leftarrow D_{i3} \xleftrightarrow{\$} D_{i4} \cup E$
end for
for $i = 1,2,...$ **do**
  $j \xleftarrow{\$} 1,2,3,4$
  $PDS_i \leftarrow D_{ij}$
  $PDS \leftarrow PDS_i \cup PDS$
end for
for $i = 1,2,...$ **do**
  $G_i \leftarrow (V_i, E_i)$
end for
for all v in V **do**
  if $v \in PDS$ **then**
    $PDF(v) = x_v = 1$
  else
    $PDF(v) = x_v = 0$
$pk \leftarrow g = \{G_1, G_2, ...\}$
$sk \leftarrow PDF$
return pk, sk

IPCC의 개인키 sk는 PDF 함수이며 공개키 pk는 그래프 G들의 집합이다. IPCC는 그래프들의 정점 집합 원소 수를 입력받아 공개키 pk와 개인키 sk를 생성한다. 가장 먼저, 그래프 각각에 대해 $G_i = (V_i, E_i)$를 초기화하고 V_i를 4개의 집합 $D_{ij}(j = 1,2,3,4)$에 $n_{i/4}$개의 원소를 랜덤하게 배정한다. 상기 선정된 4개의 집합 $D_{i1}, D_{i2,} D_{i3}, D_{i4}$에 대해 PDS 집합을 생성한다. PDS에 각 정점 v가 포함되었는지에 따라 PDF가 결정되고, 해당 PDF가 개인키 sk로 사용된다.

IPCC의 암호화는 먼저 그래프의 정점 집합 G와 최대 degree k에 대응하는 불변다항식(invariant polynomial) f_G^k를 생성한다. 이후 해당 불변다항식을 이용해 암호화한다. 세부적인 IPCC 암호화 알고리즘은 [표 2], [표 3]와 같으며 복호화 알고리즘은 [표 4]와 같다.

〔표 2〕 IPCC 암호화 알고리즘 f_G^k 생성

Require: message m, maximum degree k, graph $G = \{V, E\}$
Ensure: polynomial f_G^k

$I \xleftarrow{\$} P^*(V_i)$ such that $\forall S \in I, |S| \le K$
for $i = 1$ to $|I| - 1$ **do**
  $c_i \xleftarrow{\$} Z_p$
end for
$c_{|I|} = m - \sum_{i=1}^{|I|-1} c_i \pmod p$
$f(X_{v1}, X_{v2}, ...) = \sum_{i=1}^{|I|-1} c_i \prod_{u \in N_{[k]}} x_u$
Expand polynomial f
Replace all higher powers of a variable by

its first power
Delete a term consisting of a product of variables corresponding to a vertices with a distance equal to or less than 2
return f_G^k

〔표 3〕 IPCC 암호화 알고리즘

Require: message m, maximum degree k, public key graph set $G=\{G_1,G_2,...\}$
Ensure: ciphertext ct

$F \xleftarrow{\$} F^k$
$m = F(m_1, m_2, ...)$
$f_{G_k}^{k_i} \leftarrow SubEnc(m_i, k_i, G_i)$
$ct \leftarrow F_G^k(x_{v_1}, x_{v_2}, ...) = F(f_{G_1}^{k_1}, f_{G_2}^{k_2}, ...)$
return ct

〔표 4〕 IPCC 복호화 알고리즘

Require: ciphertext ct, private function PDF
Ensure: message m

$ct \leftarrow F_G^k(x_{v_1}, x_{v_2}, ...)$
$m = ct(PDF(v_1), PDF(v_2), ...)$
return m

2.2. 다변수 기반 전자서명 - MQ-Sign

MQ-Sign[5]은 다변수 이차식 기반 전자서명 알고리즘으로, 다변수 이차식 기반 서명 기법은 다변수 이차식 문제 (Multivariate Quadratic problem)와 다항식의 확장 동형(Extended Isomorphism of Polynomials) 문제의 어려움에 기반한 서명 기법이다. MQ-Sign은 1999년 Aviad Kipnis 등에 의해 제안된 단일 레이어를 갖는 UOV(Unbalanced Oil and Vinegar) 구조[6]를 이용하여 설계되었다.

UOV의 개인키는 두 개의 맵 F와 T로 구성되며, F는 다음과 같은 o개의 다항식 $F(X)$로 구성된다.

$$F^{(k)}(X) = \sum_{i\in O, j\in V} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i,j\in V, i\le j} \beta_{i,j}^{(k)} x_i x_j + \sum_{i\in V\cup O} \gamma_i^{(k)} x_i + \eta^{(k)}. \quad (1)$$

이때, $n(=v+o)$개의 변수 $x=(x_1,...,x_n)$는 v개의 Vinegar 변수 $(x_1,...x_v)$와 o개의 Oil 변수 $(x_{v+1},...,x_{v+o})$으로 구성된다. (1)의 이차 다항식 $F^{(k)}$는 아래의 식으로 표현할 수 있다.

$$F^{(k)} = F_V^{(k)} + F_{OV}^{(k)} + F_{L,C}^{(k)} \quad (2)$$

여기서 $F_V^{(k)}$는 Vinegar×Vinegar 변수의 곱으로 이루어진 이차다항식이며, $F_{OV}^{(k)}$는 Vinegar×Oil 변수의 곱인 이차다항식이다. $F_{L,C}^{(k)}$는 일차항과 상수항 부분이다. 개인키 T는 역행렬이 존재하는 선형 함수 $T: F_q^n \to F_q^m$를 유한체 F_q에서 랜덤하게 선택하며, UOV의 공개키는 두 개의 개인키 F와 T의 합성으로 이루어진 $P=F\circ T$를 계산하여 생성한다.

MQ-Sign은 키 생성 과정, 서명 생성 과정, 검증 과정 등 총 세 가지 과정으로 구성되어 있다. 먼저 키 생성 과정에서는 서명 생성에 사용할 공개키/개인키 쌍을 생성하는 과정이다. 개인키는 두 개의 맵 F와 T로 구성된다. $F^{(k)}$는 (2)의 $F_V^{(k)}, F_{OV}^{(k)}, F_{L,C}^{(k)}$ 의 여러 조합에 따라 결정된다.

$F_V^{(k)}, F_{OV}^{(k)}$는 랜덤하게 선택되는 경우를 $F_V^{(k)}=F_{V,R}^{(k)}$, $F_{OV}^{(k)}=F_{OV,S}^{(k)}$로 표기하며, $k=1,...,o$에 대하여, 다음 두 이차다항식으로 선택할 수 있다.

$$F_V^{(k)} = F_{V,S}^{(k)} = \sum_{i=1}^{v} \alpha_i^k x_i x_{(i+k-1 (\mathrm{mod}\, v))+1},$$
$$F_{OV}^{(k)} = F_{OV,S}^{(k)} = \sum_{i=1}^{v} \beta_i^k x_i x_{(i+k-2 (\mathrm{mod}\, o))+v+1}$$

전체 비밀키는 아래의 네 가지 조합 중 선택하여 사용할 수 있다.

$$F_{SS}^{(k)} = F_{V,S}^{(k)} + F_{OV,S}^{(k)} + F_{L,C}^{(k)},$$
$$F_{RS}^{(k)} = F_{V,R}^{(k)} + F_{OV,S}^{(k)} + F_{L,C}^{(k)},$$
$$F_{SR}^{(k)} = F_{V,S}^{(k)} + F_{OV,R}^{(k)} + F_{L,C}^{(k)},$$
$$F_{RR}^{(k)} = F_{V,R}^{(k)} + F_{OV,R}^{(k)} + F_{L,C}^{(k)}.$$

개인키 T는 역행렬이 존재하는 선형 함수

$T: F_q^n \rightarrow F_q^n$ 를 유한체 F_q에서 랜덤하게 선택한 후 T^{-1}를 구한다. 공개키 P는 $P = F \circ T$ 를 계산하여 생성한다. 세부적인 MQ-Sign 키 생성 알고리즘은 [표 5]와 같다.

〔표 5〕 MQ-Sign 키 생성 알고리즘

Require: parameters (q,v,o), length of salt l
Ensure: key pair (sk,pk)

$m \leftarrow o$
$n \leftarrow m + v$
repeat
  $M_T \leftarrow \text{Matrix}(q,n,n)$
until $\text{IsInvertible}(M_T) = \text{TRUE}$
$T \leftarrow M_T$
$InvT \leftarrow M_T^{-1}$
$F \leftarrow \text{MQmap}(q, v, o)$
$P \leftarrow F \circ T$
$sk \leftarrow (F, InvT, l)$
$pk \leftarrow (P, l)$
Return (sk,pk)

MQ-Sign의 서명은 주어진 메시지 M에 대하여 비밀키 $< F, T^{-1} >$를 이용하여 서명을 생성한다. 길이가 l인 랜덤한 salt 값 r을 선택한 후 $H(M \| r)$를 계산한다. 또한, Vinegar 값 $s_V = (s_1,...,s_v) \in F_q^v$를 랜덤하게 선택한 후 $F^{(k)}$에 대입하여 o개의 해를 갖는 o개의 연립 일차 방정식을 얻는다. 연립 일차 방정식의 계수의 행렬을 R이라 하고, R을 다음과 같은 블록 행렬로 나타낸 후 BMI(Block matric inversion) method[7]를 사용하여 R^{-1}를 구하지 않고 $R^{-1} \cdot \alpha$를 직접 계산한다.

$$R = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} I & O \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & O \\ 0 & D - CA^{-1}B \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

$$= L \circ D_{S_c} \circ U$$

$$R^{-1} = U^{-1} \circ D_{S_c}^{-1} \circ L^{-1}$$

$$R^{-1} \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_0 \end{pmatrix} =$$

$$\begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & O \\ 0 & [D - CA^{-1}B]^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_0 \end{pmatrix}$$

$R^{-1} \cdot \alpha$를 계산하여 구한 연립 일차 방정식의 해의 집합 s_o와 랜덤하게 선택한 v개의 Vinegar 집합 s_v가 최종해 $s = (s_v, s_o)$가 된다. $T^{-1}(s) = \text{z}$를 계산하여 서명 값 $\sigma = (\text{z}, r)$을 구한다. 세부적인 MQ-Sign 서명 생성 알고리즘은 [표 6]와 같다.

〔표 6〕 MQ-Sign 서명 생성 알고리즘

Require: message M, prive key $(F, InvT)$, length of the salt l.
Ensure: signature $\sigma = (\text{z}, r) \in F_q^n \times \{0,1\}^l$ such that $P(\text{z}) = H(M \| r)$

repeat
  $y_1,...,y_v \leftarrow_R F_q$
  $\hat{f}^{(v+1)},...,\hat{f}^{(n)} \leftarrow f^{(v+1)}(y_1,...,y_v),...,f^{(n)}(y,...y_v)$
  $(A, A_{S_c}, B, C, D, C_V) \leftarrow Aff^{-1}(\hat{f}^{(v+1)},...,\hat{f}^{(n)})$
until $\text{IsInvertible}(A, A_{S_c}) = \text{TRUE}$
$InvR = (A^{-1}, A_{S_c}^{-1})$
$r \leftarrow \{0,1\}^l$
$\text{x} \leftarrow H(M \| r)$
$(y_{v+1},...,y_n) \leftarrow \text{BMI}(R, H(M \| r) - C_V)$
$\text{z} = InvT \cdot \text{y}$
$\sigma \leftarrow (\text{z}, r)$
Return σ

상기 과정을 통해 생성된 MQ-Sign 서명에 대한 검증은 공개키 P의 변수에 σ값을 대입하여 계산한 값 $P(\text{z})$와 동일하게 유도되는지를 확인하도록 수행된다. MQ-Sign의 서명 검증 알고리즘은 [표 7]과 같다.

〔표 7〕 MQ-Sign 서명 검증 알고리즘

Require: message M,
  signature $\sigma = (z, r) \in F_q^n \times 0,1^l$
Ensure: boolean value TRUE or FALSE

$h \leftarrow H(M \| r)$
$h' \leftarrow P(z)$
if $h' = h$ **then**
  return TRUE
else
  return FALSE
end if

2.3. 아이소제니 기반 전자서명 - FIBS

FIBS[8]는 아이소제니를 기반으로 하는 전자서명 알고리즘으로 해시 기반 전자서명 생성 방식인 XMSS[9] (eXtended Merkle Signature Scheme)과 WOTS+[10] (Winternitz One-time Signature)에서 양자컴퓨터에 내성을 갖는 CGL 해시함수[11]를 사용하도록 설계된 전자서명 알고리즘이다. CGL 해시함수는 SHA와 같은 기존 해시함수와는 다르게 아이소제니를 기반으로 설계되어 양자컴퓨터에 내성을 갖고 있다. XMSS와 WOTS+를 사용한 전자서명의 안전성은 내부 해시함수의 의존하므로 FIBS는 내부에 사용된 CGL 해시함수의 안전성에 따라 양자내성 안전성을 제공할 수 있다.

CGL 해시함수는 타원곡선 연산을 수행함에 따라 기존 해시 기반 전자서명 알고리즘과 일부 양자내성 전자서명 알고리즘들에 대비하여 속도가 느린 한계가 있다. 하지만, 키 사이즈가 작고 최적화 구현의 가능성이 있어 자원제약 환경에서 유리한 암호이다.

FIBS는 키 생성 과정, 서명 생성 과정, 검증 과정 등 총 세 가지 과정으로 구성되어 있다. 먼저 키 생성 과정에서는 서명 생성에 사용할 공개키/개인키 쌍을 생성하는 과정이다. 공개키와 개인키는 다음과 같이 각각 두 가지 요소들로 구성된다. 세부적인 키 생성 알고리즘은 [표 8]과 같다.

- 공개키 Pk = (root,seed)
  - Pk.root: 최상위 레벨 트리의 루트
  - Pk.seed: 무작위로 생성된 n-바이트 시드 값
- 개인키 Sk = (seed,prf)
  - Sk.seed: 무작위로 생성된 n-바이트 시드 값
  - Sk.prf: n-바이트 의사난수 함수 키

〔표 8〕 FIBS 키 생성 알고리즘

Require: –
Ensure: Pk.root, Pk.seed, Sk.seed, Sk.prf

1: SK.seed $\xleftarrow{\$} \{0,1\}^n$
2: SK.prf $\xleftarrow{\$} \{0,1\}^n$
3: PK.seed $\xleftarrow{\$} \{0,1\}^n$
4: **for** $i = 1,\ldots,2^{h/d}$ **do**
5: WOTS.Sk[i], WOTS.Pk[i] ← WOTS_KeyGen(SK.seed, PK.seed, ADRS)
6: **end for**
7: PK.root ← XMSS_PkGen(SK.seed, PK.seed, ADRS)
8: return Pk.root, Pk.seed, Sk.seed, Sk.prf

FIBS의 서명은 개인키 중 의사난수 함수 키 Sk.prf와 메시지를 의사난수생성기의 입력으로 사용하여 난수 R을 생성한 후에, R과 공개키(Pk.root,Pk.seed)를 사용하여 CGL 해시함수 수행을 통해 WOTS+ 서명을 생성한다. 이를 반복하여 FORS 서명을 생성한다. 세부적인 FIBS 서명 알고리즘은 [표 9]와 같다.

〔표 9〕 FIBS 서명 알고리즘

Require: M, Sk.seed, Sk.prf, Pk.seed, Pk.root
Ensure: FIBS.Sig

1: OptRand $\xleftarrow{\$} \{0,1\}^n$
2: $R \leftarrow \mathrm{PRF}_{msg}$(SK.prf, OptRand, M)
3: FIBS.Sig = FIBS.Sig $\| R$
4: Digest ← H_{msg}(R, Pk.seed, Pk.root, M)
5: md, ind.tree, ind.leaf ← Split_md(Digest)
6: FORS.Sig ← FORS_Sign(md, Sk.seed, Pk.seed, ADRS)
7: FIBS.Sig = FIBS.Sig ‖ FORS.Sig
8: FORS.Sig ← FORS_PkFromSig(FORS.Sig, M, Pk.seed, ADRS)
9: Sign.tmp ← XMSS_Sign(FORS.Sig, Sk.seed, ind.leaf, Pk.seed, ADRS)
10: HT.Sig ← HT.Sig ‖ Sign.tmp
11: root ← XMSS_PkFromSig(ind.leaf, Sig.tmp, M, Pk.seed, ADRS)
12: **for** $j = 1,\ldots,d-1$ **do**
13: Sig.tmp = XMSS_Sign(root, Sk.seed, ind.leaf, Pk.seed, ADRS)
14: HT.Sig = HT.Sig ‖ Sig.tmp
15: **if** $j < d-1$ **then**
16: root = XMSS_PkFromSig(ind.leaf, Sig.tmp, root, Pk.seed, ADRS)
17: **end if**
18: **end for**
19: FIBS.Sig = FIBS.Sig ‖ HT.Sig
20: return FIBS.Sig

상기 과정을 통해 생성된 FIBS 서명에 대한 검증은 CGL 해시함수의 결과 값과 인덱스를 다시 계산하고 이를 기반으로 Pk.root가 동일하게 유도되는지를 확인하

도록 수행된다. FIBS의 서명 검증 알고리즘은 [표 10]과 같다.

〔표 10〕 FIBS 검증 알고리즘

Require: M, FIBS.Sig, Pk
Ensure: Accept or Reject

$R \leftarrow$ Get R from FIBS.Sig
FORS.Sig $\leftarrow$ Get FORS.Sig from FIBS.Sig
HT.Sig $\leftarrow$ Get HT.Sig from FIBS.Sig
Sig.tmp $\leftarrow$ Get Sig.tmp from HT.Sig
Digest $\leftarrow H_{msg}(R, Pk.seed, Pk.root, M)$
md, ind.tree, ind.leaf $\leftarrow$ Split_md(Digest)
FORS.Pk $\leftarrow$ FORS_PkFromSig(FORS.Sig, md, Pk.seed, ADRS)
node $\leftarrow$ XMSS_PkFromSig(ind.leaf, Sig.tmp, M, Pk.seed, ADRS)
for $j = 1, \ldots, d-1$ **do**
    node $\leftarrow$ XMSS_PkFromSig(ind.leaf, Sig.tmp, node, Pk.seed, ADRS)
end for
if node = Pk.root **then**
    return Accept
else
    return Reject
end if

2.4. 영지식 기반 전자서명 - AIMer

AIMer[12]는 MPCinH[13]라고 하는 영지식 증명 프레임워크에 AIM이라는 일방향 함수를 적용한 전자서명 알고리즘이다. MPCinH는 NP 관계에 있는 연산들을 다자간연산을 통해 효율적으로 영지식 증명 프로토콜을 설계할 수 있도록 하며, 다자간연산에서 파티들이 실제로 상호작용하는 것이 아니라 암호학적인 기법을 사용하여 가상으로 다자간연산을 수행하는 방식을 말한다. AIMer에서는 앞서 언급한 AIM이라는 일방향 함수를 BN++ proof system[14]을 통해 대화형 영지식 증명 시스템 형태를 구성한다. 또한, 이를 Fiat-Shamir transfomation[15]을 사용하여 비대화형 영지식 증명의 형태로 변환함으로써 생성된 proof를 기반으로 전자서명을 수행할 수 있도록 설계하였다. 이와 같이 설계된 AIMer는 AIM의 일방향성을 기반으로 양자내성을 제공한다.

AIMer에서 사용된 AIM은 multi-target one-wayness를 제공할 수 있도록 설계된 tweakable 일방향 함수다. 입력/출력 크기가 n이고 $(\ell+1)$-튜플 $(e_1,\ldots,e_\ell,e_*)$가 주어졌을 때, $\mathrm{AIM}:\mathbb{F}_{2^n}\times\mathbb{F}_{2^n}\rightarrow\mathbb{F}_{2^n}$은 다음과 같이 정의된다.

$$\mathrm{AIM}(\mathrm{iv},\mathrm{pt}) = \mathrm{Mer}[e_*] \circ \mathrm{Lin}[\mathrm{iv}] \circ \mathrm{Mer}[e_1,\ldots,e_\ell](\mathrm{pt}) \oplus \mathrm{pt}$$

여기서, $\mathrm{Mer}[e](x)$ $(x \in \mathbb{F}_{2^n})$는 지수가 메르센 수(mersenne number) 2^e-1인 지수승 연산 x^{2^e-1}로 정의되며 비선형 연산인 S-box 역할을 수행한다.

또한, $\mathrm{Lin}[\mathrm{iv}](x)$는 선형 연산으로 $n\times\ell n$ 행렬 $A_{\mathrm{iv}} = [A_{\mathrm{iv},1}|\ldots|A_{\mathrm{iv},\ell}] \in (\mathbb{F}_2^{n\times n})^\ell$에 대한 곱셈 연산과 벡터 $b_{\mathrm{iv}} \in \mathbb{F}_2^n$에 대한 덧셈 연산으로 구성되어 있다. 행렬 곱셈 연산은 행렬이 선형방정식으로 표현하여 $Ax = \Sigma_{1\le i\le\ell} L_{iv,i}(x_i)$와 같이 구성되며, 이 연산 결과에 b_{iv}를 더한다. 즉, 입력이 $x = (x_1,\ldots,x_\ell) \in (\mathbb{F}_{2^n})^\ell$로 주어졌을 때, $\mathrm{Lin}[\mathrm{iv}](x)$은 $\Sigma_{1\le i\le\ell} L_{\mathrm{iv},i}(x_i) \oplus b_{\mathrm{iv}}$과 같이 정의된다. 여기서, A_{iv}와 b_{iv}는 iv를 입력으로 한 XOF (eXtendable-Output Function) 연산을 통해 생성한다.

AIM은 128, 192, 256-비트 보안 강도를 제공할 수 있도록 세 가지 인스턴스(AIM-Ⅰ, AIM-Ⅲ, AIM-Ⅴ)로 구성되어 있다. 각 인스턴스에 대한 파라미터 셋은 [표 11]과 같이 구성되어 있다.

〔표 11〕 AIM 인스턴스 및 파라미터 명세

Instances	λ	n	l	e_1	e_2	e_3	e_*
AIM-Ⅰ	128	128	2	3	27	-	5
AIM-Ⅲ	192	192	2	5	29	-	7
AIM-Ⅴ	256	256	3	3	53	7	5

AIMer은 키 생성, 서명 생성, 검증 알고리즘으로 구성된다. 키 생성 단계에서는 AIM의 파라미터들을 입력으로 공개키와 개인키를 생성한다. 생성된 공개키는 $(\mathrm{iv},\mathrm{ct})$로 각각은 AIM에 사용되는 초기벡터와 메시지 pt에 대한 연산 결과인 $\mathrm{ct} = \mathrm{AIM}(\mathrm{iv},\mathrm{pt})$를 의미한다. 또한, ct에 대응되는 메시지 pt는 개인키가 된다.

키생성 과정을 통해 생성된 키 쌍 $(\mathrm{pt},(\mathrm{iv},\mathrm{ct}))$는 서

명 알고리즘의 입력되어 메시지 m에 대한 서명 σ를 생성한다. 서명 생성 과정은 MPCitH 프레임워크에서 영지식을 구성하기 위한 과정으로 5가지 단계로 구성되어 있다. 서명 생성 과정의 세부적인 과정은 [표 12]에서 확인할 수 있으며, 각 단계의 설명은 다음과 같다. 1단계에서는 초기 설정 과정으로 개별 파티들이 암호학적 프로토콜에 적용할 입력값과 실행 결과들을 구성하는 단계이다. 여기서, AIM에서 사용되는 메르센 수 지수 연산의 결과와 개인키를 기반으로 BN++ proof system에서 사용할 다양한 입력 값을 구성한다. 이 때, 추후 Fiat-Shamir transformation을 통한 비대화형 영지식으로의 변환을 수행하기 위해 입력값들에 임의성을 추가하는 과정들(Commit, Expand, ExpandTape, Sample)이 포함되어 있다. 이와 같은 과정을 통해 생성된 입력값들은 σ_1으로 묶여 커밋된다.

2단계에서는 1단계를 통해 커밋된 σ_1, 메시지 m, 공개키 $(\mathrm{iv},\mathrm{ct})$를 해시함수 H_1에 입력하여 해시값 h_1을 계산하는 과정으로, Fiat-Shamir transform에서 해시함수를 random oracle로 사용하여 비대화형 영지식 증명의 proof를 구성하는 과정에 해당한다.

3단계에서는 1단계에서 구성한 입력값들을 BN++ proof system에 적용하여 다자간연산의 일치성을 확인하는 단계로, 증명자는 각 파티들이 개별적으로 계산한 결과의 총합이 0이 되도록 구성한 후 파티들에서 계산 결과를 요구하고 응답을 검증하는 방식으로 구성된다. 이와 같은 방식은 triple check protocol (이하, TCP)이라고 하며 세부적인 동작 과정은 다음과 같다. N개의 파티가 유한체 $\mathbb{F}$ 상의 C개의 곱셈 triple $(x_j,y_j,z_j=x_j\cdot y_j)_{j=1}^{C}$과 연관된 곱셈 연산 결과를 공유하고 확인함으로써 x_j와 y_j를 알고 있는지를 검증한다. 이를 위해 helping values라고 하는 보조 입력값 $(((a_j,b_j)_{j=1}^{C},c))$이 주어진다. 여기서 $a_j\leftarrow\mathbb{F}$, $b_j=y_j$, $c=\Sigma_{j=1}^{C}a_j\cdot b_j$이다. 상기 입력값들을 사용하여 다음과 같은 상호작용을 수행한다.

1. 검증자는 임의의 챌린지(challenges) $\epsilon_1,\cdots,\epsilon_C\in\mathbb{F}$를 준비한다.
2. 각 파티는 개별적으로 $\alpha_j^{(i)}=\epsilon_j\cdot x_j^{(i)}+a_j^{(i)}$, $i\in[N]$를 계산하여 $\alpha_1^{(i)},\cdots,\alpha_C^{(i)}$를 준비하며 이를 브로드캐스팅하여 모든 파티에게 공유한다.
3. 각 파티는 공유된 α_j를 사용하여 개별적으로 $v^{(i)}=\Sigma_{j=1}^{C}\epsilon_j\cdot z_j^{(i)}-\Sigma_{j=1}^{C}\alpha_j\cdot b_j^{(i)}+c^{(i)},\quad i\in[N]$를 계산하고 v를 브로드캐스팅하여 모든 파티에게 공유한다.
4. 검증자와 모든 파티들은 공유된 v가 0인지를 확인하여 연산 결과를 검증한다.

상기 설명한 TCP는 검증자와 증명자가 상호작용하는 대화형이지만 MPCitH 방식으로 설계되는 AIMer에서는 이를 가상의 파티들이 존재한다고 가정하고 시뮬레이션 형태로 수행한다. TCP에서 사용되는 입력값들은 1단계에서 구성한 입력값들로 개인키 pt와 관련된 값들이다. 즉, 3단계에서는 TCP를 통해 개인키 정보를 영지식 증명을 통해 확인할 수 있도록 변환하는 과정으로 생각할 수 있다. 이 과정에서 계산된 α와 v는 salt와 함께 σ_2로 묶여 추후 Fiat-Shamir transformation에 사용된다.

4단계에서는 비대화형 영지식 proof 생성을 위해 2단계에서 계산한 h_1과 3단계에서 계산한 σ_2를 해시함수 H_2에 입력하여 해시값 $h2$를 생성한다. 마지막으로 5단계에서는 σ_1과 σ_2를 포함하여 앞 단계들에서 요구되는 입력값들을 계산하기 위해 필요한 값들을 묶어 최종적인 서명 σ를 생성하고 이를 반환한다.

서명 검증 과정에서는 공개키 $(\mathrm{iv},\mathrm{ct})$와 메시지 m을 사용하여 서명 σ를 올바르게 다시 유도할 수 있는지를 확인하는 과정으로 구성되어 있다. 서명 검증 과정은 입력된 σ를 파싱한 후 TCP에서 사용할 입력값들을 유도한 후, h_1의 계산, 유도된 입력값들을 사용한 TCP 수행과 h_2의 계산을 수행하여 같은 σ 값이 유도되는지를 확인하도록 구성되어 있다. 이에 대한 세부적인 동작과정은 [표 13]과 같다.

상기 설명한 키 생성, 서명 생성, 서명 검증 과정에서 사용되는 XOF와 해시함수 Commit, H_1, H_2와 의사난수생성기 Expand, ExpandTape는 SHAKE[16]가 사용되었다.

〔표 12〕 AIMer 서명 생성 알고리즘

Require: pt, $(\mathrm{iv},\mathrm{ct})$, m **Ensure:** σ
Phase 1: Commiting to the seeds and the execution views of the parties.
1: Sample a random salt $\mathrm{salt}\xleftarrow{\$}\{0,1\}^{2\lambda}$

2: Compute the first ℓ S-boxes' output $t_1,\ldots,t_\ell$

3: Derive the binary matrix $A_{\mathrm{iv}} \in (\mathbb{F}_2^{n\times n})^{\ell}$ and the vector $b_{\mathrm{iv}} \in \mathbb{F}_2^{n}$ from the initial vector iv.

4: **for** *each parallel execution* $k \in [\tau]$ **do**

5: $\mathrm{seed}_k \xleftarrow{\$} \{0,1\}^{2\lambda}$

6: Compute parties' seeds $\mathrm{seed}_k^{(1)},\ldots,\mathrm{seed}_k^{(N)}$ as leaves of binary tree from seed_k

7: **for** *each party* $i \in [N]$ **do**

8: $\mathrm{com}_k^{(i)} \xleftarrow{\$} \mathrm{Commit}(\mathrm{salt}, k, i, \mathrm{seed}_k^{(i)})$

9: $\mathrm{tape}_k^{(i)} \leftarrow \mathrm{ExpandTape}(\mathrm{salt}, k, i, \mathrm{seed}_k^{(i)})$

10: $\mathrm{pt}_k^{(i)} \leftarrow \mathrm{Sample}(\mathrm{tape}_k^{(i)})$

11: $\Delta \mathrm{pt}_k \leftarrow \mathrm{pt} - \Sigma_i \mathrm{pt}_k^{(i)}$, $\mathrm{pt}_k^{(1)} \leftarrow \mathrm{pt}_k^{(1)} + \Delta \mathrm{pt}_k$

12: **for** *each S-box with index* j **do**

13: **if** $j \le \ell$ **then**

14: For each party i, $t_{k,j}^{(i)} \leftarrow \mathrm{Sample}(\mathrm{tape}_k^{(i)})$

15: $\Delta t_{k,j} = t_j - \Sigma_i t_{k,j}^{(i)}$, $t_{k,j}^{(1)} \leftarrow t_{k,j}^{(1)} + \Delta t_{k,j}$

16: For each party i, $x_{k,j}^{(i)} = t_{k,j}^{(i)}$ and $z_{k,j}^{(i)} = (\mathrm{pt}_k^{(i)})^{2^r j}$

17: **if** $j = \ell+1$ **then**

18: For $i=1$, $x_{k,j}^{(i)} = A_{\mathrm{iv}} \cdot t_{k,*}^{(i)} + b_{\mathrm{iv}}$, where $t_{k,*}^{(i)} = [t_{k,1}^{(i)} | \ldots | t_{k,\ell}^{(i)}]$ is the output shares of the first ℓ S-boxes

19: For each party $i \in [N]\backslash\{1\}$, $x_{k,j}^{(i)} = A_{\mathrm{iv}} \cdot t_{k,*}^{(i)}$

20: For each party i, $z_{k,j}^{(i)} = (x_{k,j}^{(i)})^{2^r *} + \mathrm{ct} \cdot x_{k,j}^{(i)}$

21: For each party i, $a_k^{(i)} \leftarrow \mathrm{Sample}(\mathrm{tape}_k^{(i)})$

22: $a_k = \Sigma_{i=1}^{n} a_k^{(i)}$

23: $c_k = a_k \cdot \mathrm{pt}$

24: For each party i, $c_k^{(i)} \leftarrow \mathrm{Sample}(\mathrm{tape}_k^{(i)})$

25: $\Delta c_k = c_k - \Sigma_i c_k^{(i)}$, $c_k^{(1)} \leftarrow c_k^{(1)} + \Delta c_k$

26: $\sigma_1 \leftarrow (salt, ((com_k^{(i)})_{i\in[N]}, \Delta \mathrm{pt}_k, \Delta c_k, (\Delta t_{k,j})_{j\in[l]})_{k\in[\tau]})$

Phase 2: Challenging the checking protocol.

27: $h_1 \leftarrow H_1(m, \mathrm{iv}, \mathrm{ct}, \sigma_1)$

28: $((\epsilon_{k,j})_{j\in[\ell+1]})_{k\in[\tau]} \leftarrow \mathrm{Expand}(h_1)$ where $\epsilon_{k,j} \in \mathbb{F}_{2^n}$

Phase 3: Commit to the simulation of the checking protocol.

29: **for** *each repetition* k **do**

30: Simulate the triple check protocol in the BN++ proof system for all parties with challenge $\epsilon_{k,j}$. The inputs are $((x_{k,j}^{(i)}, \mathrm{pt}_k^{(i)}, z_{k,j}^{(i)})_{j\in[\ell+1]}$, where $b_k^{(i)} = \mathrm{pt}_k^{(i)}$, and let $\alpha_k^{(i)}$ and $v_k^{(i)}$ be the broadcast values

31: $\sigma_2 \leftarrow (\mathrm{salt}, ((\alpha_k^{(i)}, v_k^{(i)})_{i\in[N]})_{k\in[\tau]})$

Phase 4: Challenging the views of the MPC protocol.

32: $h_2 \leftarrow H_2(h_1, \sigma_1)$

33: $(\bar{i}_k)_{k\in[\tau]} \leftarrow \mathrm{Expand}(h_2)$ where $\bar{i}_k \in [N]$

Phase 5: Opening the views of the MPC and checking protocol.

34: **for** *each repetition* k **do**

35: $\mathrm{seeds}_k \leftarrow \{$ $\lceil \log_2(N) \rceil$ nodes to compute $\mathrm{seed}_k^{(i)}$ for $i \in [N]\backslash\{\bar{i}_k\}\}$

36: $\sigma \leftarrow (\mathrm{salt}, h_1, h_2, (\mathrm{seeds}_k, \mathrm{com}_k^{(\bar{i}_k)}, \Delta \mathrm{pt}_k, \Delta ct_k, (\Delta t_{k,j})_{j\in[\ell]}, \alpha_k^{(\bar{i}_k)})_{k\in[\tau]})$

〔표 13〕 AIMer 서명 검증 알고리즘

Require: $(\mathrm{iv}, \mathrm{ct})$, m, σ
Ensure: Accept or Reject

1: Parse σ as $(\mathrm{salt}, h_1, h_2, (\mathrm{seeds}_k, \mathrm{com}_k^{(\bar{i}_k)}, \Delta \mathrm{pt}_k, \Delta ct_k, (\Delta t_{k,j})_{j\in[\ell]}, \alpha_k^{(\bar{i}_k)})_{k\in[\tau]})$

2: Derive the binary matrix $A_{\mathrm{iv}} \in (\mathbb{F}_2^{n\times n})^{\ell}$ and the vector $b_{\mathrm{iv}} \in \mathbb{F}_2^{n}$ from the initial vector iv.

3: $((\epsilon_{k,j})_{j\in[\ell+1]})_{k\in[\tau]} \leftarrow \mathrm{Expand}(h_1)$ and $(\bar{i}_k)_{k\in[\tau]} \leftarrow \mathrm{Expand}(h_2)$

4: **for** *each parallel execution* $k \in [\tau]$ **do**

5: Uses seeds_k to recompute $\mathrm{seed}_k^{(i)}$ for $i \in [N]\backslash\{\bar{i}_k\}$

6: **for** *each party* $i \in [N]\backslash\{\bar{i}_k\}$ **do**

7: $\mathrm{com}_k^{(i)} \xleftarrow{\$} \mathrm{Commit}(\mathrm{salt}, k, i, \mathrm{seed}_k^{(i)})$

8: $\mathrm{tape}_k^{(i)} \leftarrow \mathrm{ExpandTape}(\mathrm{salt}, k, i, \mathrm{seed}_k^{(i)})$

9: $\mathrm{pt}_k^{(i)} \leftarrow \mathrm{Sample}(\mathrm{tape}_k^{(i)})$

10: **if** $i = 1$ **then**

11: $\mathrm{pt}_k^{(i)} \leftarrow \mathrm{pt}_k^{(i)} + \Delta \mathrm{pt}_k$

12: **for** *each S-box with index* j **do**

13: **if** $j \le \ell$ **then**

14: $t_{k,j}^{(i)} \leftarrow \mathrm{Sample}(\mathrm{tape}_k^{(i)})$

15: **if** $i = 1$ **then**

16: $t_{k,j}^{(1)} \leftarrow t_{k,j}^{(1)} + \Delta t_{k,j}$

17: $x_{k,j}^{(i)} = t_{k,j}^{(i)}$ and $z_{k,j}^{(i)} = (\mathrm{pt}_k^{(i)})^{2^e j}$

18: **if** $j = \ell + 1$ **then**

19: **if** $i = 1$ **then**

20: $x_{k,j}^{(i)} = A_{\mathrm{iv}} \cdot t_{k,*}^{(i)} + b_{\mathrm{iv}}$, where $t_{k,*}^{(i)} = [t_{k,1}^{(i)} | ... | t_{k,\ell}^{(i)}]$ is the output shares of the first ℓ S-boxes

21: **else**

22: $x_{k,j}^{(i)} = A_{\mathrm{iv}} \cdot t_{k,*}^{(i)}$

23: $z_{k,j}^{(i)} = (x_{k,j}^{(i)})^{2^{e*}} + \mathrm{ct} \cdot x_{k,j}^{(i)}$

24: $a_k^{(i)} \leftarrow \mathrm{Sample}(\mathrm{tape}_k^{(i)})$ and $c_k^{(i)} \leftarrow \mathrm{Sample}(\mathrm{tape}_k^{(i)})$

25: **if** $i = 1$ **then**

26: $c_k^{(i)} = c_k^{(i)} + \Delta c_k$

27: $\sigma_1 \leftarrow (salt, ((com_k^{(i)})_{i \in [N]}, \Delta \mathrm{pt}_k, \Delta c_k, (\Delta t_{k,j})_{j \in [l]})_{k \in [\tau]})$

28: $h_1' \leftarrow H_1(m, \mathrm{iv}, \mathrm{ct}, \sigma_1)$

29: **for** *each parallel execution* $k \in [\tau]$ **do**

30: **for** *each party* $i \in [N] \backslash \{\bar{i}_k\}$ **do**

31: Simulate the triple check procotol in the BN++ proof system for all parties with challenge $\epsilon_{k,j}$. The inputs are $((x_{k,j}^{(i)}, \mathrm{pt}_k^{(i)}, z_{k,j}^{(i)})_{j \in [\ell+1]}$, where $b_k^{(i)} = \mathrm{pt}_k^{(i)}$, and let $\alpha_k^{(i)}$ and $v_k^{(i)}$ be the broadcast values

32: $v_k^{(\bar{i}_k)} = 0 - \Sigma_{i \neq \bar{i}_k} v_k^{(i)}$

33: $\sigma_2 \leftarrow (\mathrm{salt}, ((\alpha_k^{(i)}, v_k^{(i)})_{i \in [N]})_{k \in [\tau]})$

34: $h_2' \leftarrow H_2(h_1, \sigma_1)$

35: Output Accept if $h_1 = h_1'$ and $h_2 = h_2'$

36: Ohterwise, output Reject

Ⅲ. 암호 알고리즘 별 안전성 및 효율성

3.1. IPCC 안전성 및 효율성

IPCC는 80-비트 안전성을 제공할 수 있도록 설계되었으며, 하나의 파리미터 집합만 정의되어있다. IPCC의 안전성은 키복구 전수조사 공격과 평문 복구 공격에 대한 안전성이 제시되었다. IPCC는 각 그래프마다 공개키, 개인키 쌍이 존재하므로 각각에 대해 키 복구 공격을 해야하며, 키 전수조사 공격은 그래프가 2개일 때, 더 큰 정점의 개수를 가진 그래프를 공격하는 복잡도에 의존한다. IPCC에서는 정점의 개수가 200일 경우 $O(2^{158})$으로 키 복구 공격에 대해 안전함을 보였다. 또한, 평문 복구 공격에 대해서는 그래프 2개의 정점의 총 개수가 400이고 암호문 최대 차수 k가 8일 경우 약 162-비트의 안전성을 가짐을 보였다.

IPCC의 공개키와 개인키는 각각 4,800, 400바이트이다. IPCC의 성능은 키 생성, 암호화, 복호화의 단계별 실행시간으로 평가되었다. 성능 측정은 Intel Core-i7-9700K CPU@3.6HZ processor, 16GB RAM 상에서 수행되었으며, 키 생성, 암호화 복호화의 실행시간은 각각 0.028ms, 2.411ms, 3.363ms으로 측정되었다.

3.2. MQ-Sign 안전성 및 효율성

MQ-Sign은 256-비트 안전성을 제공할 수 있도록 설계되었으며, 방정식의 개수 o는 짝수이며, Direct 공격 분석에 기반하여 128-, 192-, 256-비트의 안전도에서 $0 \geq 46,72,96$이 되도록 한다. Vinegar 변수 v는 교차 공격에 대한 저항성 보장하기 위해 $v \geq 1.5o$를 만족해야 하면, 128-, 192-, 256-비트의 안전도에서 $v = 72, 112, 148$로 선택한다. 이를 위해 MQ-Sign의 설계단계에서 Direct Attack[17], UOV-Reconciliation

〔표 14〕 MQ-Sign 최적화 구현 성능 (AVX2)

Scheme	Security Category	I	Ⅲ	Ⅴ
MQ-Sign-SS	KeyGen	6,046,385	25,506,351	62,972,759
	Sign	154,645	378,634	471,085
	Verify	71,267	232,377	401,412
MQ-Sign-RS	KeyGen	9,026,556	38,892,327	95,016,980
	Sign	174,790	447,656	626,373
	Verify	71,267	232,377	401,412
MQ-Sign-SR	KeyGen	10,222,889	43,634,459	104,441,512
	Sign	166,987	417,445	630,000
	Verify	71,267	232,377	401,412
MQ-Sign-RR	KeyGen	13,493,778	56,071,342	138,481,524
	Sign	184,761	491,738	708,415
	Verify	71,267	232,377	401,412

* the perfomance is evaluated by the number of CPU cycles.

Attack[18, 19], Kipnis-Shamir Attack[6], Intersection Attack[20], Implementation Attack[21] 등 algebraic cryptanalysis에 대한 안전성과 UOV 서명의 위조불가능성 (existential unforgeability against adaptive chosen-message attacks: EUF-CMA)[22]에 대한 안전성을 제시함과 더불어, 양자 환경에서의 적용가능한 Grover's algorithms을 이용한 공격 기법에 대한 안전성을 제시하였다.

MQ-Sign은 성능은 키 생성에서 비밀 키의 다른 선택에 대하여 각 단계별로 평가되었으며, AVX2 플랫폼에서 구현되었으며, 서명 및 검증(또는 키 생성)의 각 결과는 GNU GCC 버전 9.4.0 컴파일러를 사용하여 C 프로그래밍 언어로 100,000회(또는 10,000회) 측정한 평균이다. [표 14]는 키 생성에서는 비밀 키의 다른 선택으로 인해 서로 다른 결과가 나타날 수 있지만, 검증에 대해서는 동일한 결과가 나타나는 걸 보여준다.

3.3. FIBS 안전성 및 효율성

FIBS는 128-비트 안전성을 제공할 수 있도록 설계되어 FIBS-128로 하나의 인스턴스만 정의되어있다. FIBS의 안전성은 내부에서 사용된 CGL 해시함수를 기반으로 한다. 이에 대한 안전성을 PQ-DM-SPR(Post Qauntum Distinct function Multitarget Second Preimage Resistance)과 PQ-itsr(Post Quantum Indistinguishability from Target Subset Resistance)에 대하여 저항성이 있음을 보임으로써 제시하였다. 또한, FIBS에서 선택된 파라미터들이 시작 타원곡선의 엔도모르피즘 링(Endomorphism Ring)이 알려진 경우에 수행될 수 있는 KLTP 알고리즘[23]에 저항성이 있음을 보였다.

FIBS의 성능은 FIBS-128의 개인키/공개키 및 서명 크기와 키 생성, 서명 생성 및 검증 과정의 실행속도를 통해 제시되었다. FIBS-128의 개인키/공개키의 크기는 각각 32, 64바이트이며 서명은 17,088바이트이다. 키 생성, 서명 생성 및 검증의 단계별 실행시간은 Intel Core i9-10980XE, Ubuntu 20.04.5 LTS 상에서 측정되었으며 각 단계별로 121.66s, 2837.04s, 172.37s로 나타났다.

3.4. AIMer 안전성 및 효율성

AIMer는 128-, 192-, 256-비트의 안전성을 제공할 수 있도록 3가지 인스턴스(AIMer-Ⅰ, AIMer-Ⅲ, AIMer-Ⅴ)로 구성되어 있다. 각 인스턴스는 AIM-Ⅰ, AIM-Ⅲ, AIM-Ⅴ를 기반으로 한다. AIMer의 안전성은 내부 함수인 AIM의 multi-target one-wayness에 기반한다. 이를 위해 AIM의 설계단계에서 Gröber basis attack[24], XL attack[25] 등 algebraic cryptanalysis에 대한 안전성과 differential cryptanalysis와 같은 전통적인 공격 기법에 대한 안전성을 제시함과 더불어, Grover's algorithms과 같은 양자 환경에서의 적용가능한 공격들에 대한 안전성을 제시하였다.

AIMer의 성능은 개인키/공개키 및 서명의 크기와 서명 생성 및 검증의 실행속도로 제시되었다. 이는 [표 15]에서 확인할 수 있다. 먼저, 개인키/공개키의 크기는 인스턴스 별로 각각 (16, 24, 32) 바이트, (32, 48, 64) 바이트이며 생성된 서명의 크기는 파라미터에 따라 3,840~25,152바이트이다. AIMer의 실행속도는 AVX2를 통한 최적화된 성능으로 제시되었으며 128GB 메모리가 Intel Xeon E5-1650 v3 @ 3.50 GHz 상에서 측정되었다. AIMer-Ⅰ에서는 파라미터에 따라 서명 생성 0.82~29.62ms, 서명 검증 0.78~29.17ms로 나타났으며, AIMer-Ⅲ에서는 서명 생성 1.57~58.70ms, 서명 검증 1.48~58.10ms, AIMer-Ⅴ에서는 서명 생성 2.87~98.49ms, 2.78~98.64ms로 나타났다.

〔표 15〕 AIMer 최적화 구현 성능 (AVX2)

Instances	N	τ	Sign (ms)	Verify (ms)	Size (B)
AIMer-Ⅰ	16	33	0.82	0.78	5,904
	57	23	1.82	1.77	4,800
	256	17	5.96	5.90	4,176
	1,615	13	29.62	29.17	3,840
AIMer-Ⅲ	16	49	1.57	1.48	13,080
	64	33	3.86	3.62	10,440
	256	25	10.57	10.42	9,144
	1,621	19	58.70	58.10	8,352
AIMer-Ⅴ	16	65	2.87	2.78	25,152
	62	44	6.60	6.54	19,904
	256	33	19.21	19.19	17,088
	1,623	25	98.49	98.64	15,392

〔표 16〕 NIST PQC 및 기존 영지식 증명 기반 전자서명 대비 AIMer 성능 비교

Scheme	$\|pk\|$ (B)	$\|sk\|$ (B)	Sign (ms)	Verify (ms)
AIMer- I	32	5,904	0.82	0.78
Falcon-512	897	690	0.27	0.04
DIlithium2	1,312	2,420	0.10	0.03
SPHINCS$^+$-128s*	32	7,856	315.74	0.35
SPHINCS$^+$-128f*	32	17,088	16.32	0.97
Picnic-L1-full	32	30,925	1.16	0.91
Picnic3-L1	32	12,463	5.83	4.24
Banquet	32	19,776	7.09	5.24
Limbo-AES128	32	21,520	2.70	2.00
Rainier	32	8,544	0.97	0.89
BN++Rain$_3$	32	6,432	0.83	0.77

*: -SHAKE-simple

또한, AIMer의 성능은 NIST PQC 전자서명 알고리즘들(CRYSTALS-Dilithium, Falcon, SPHINCS+)과 기존 영지식 증명 기반 전자서명 알고리즘들(Picnic, Limbo, Banquet, Rainier, BN++Rain)에 대한 비교를 수행하였으며 이는 [표 16]에서 확인할 수 있다. 성능 비교 결과, 기존 NIST PQC 전자서명 알고리즘 대비 공개키의 크기가 작거나 서명 생성 및 검증 속도가 빠른 결과를 보였다. 또한, 기존 영지식 증명 기반 전자서명과 대비해서는 서명의 크기가 현저히 작은 것으로 나타났다.

Ⅳ. 결 론

본 논문에서는 KpqC 공모전 1라운드 암호 중 그래프, 다변수, 아이소제니, 영지식 기반 암호 알고리즘들에 대해 소개하고 설계원리, 동작과정 및 안전성/효율성에 대해 살펴보았다. 이들 중 그래프 기반 암호인 IPCC는 암호화 알고리즘이며 그 외 다변수 기반 암호 MQ-Sign, 아이소제니 기반 암호 FIBS, 영지식 기반 암호 AIMer는 전자서명 알고리즘이다. 각 알고리즘들은 기반 문제들로부터 암호를 설계하는 과정에서 나타날 수 있는 공격들에 대한 저항성과 기존 NIST PQC Standardization에서 선정된 암호들과의 성능 비교가 수행되었으며 기존 양자내성암호 대비 키 사이즈 측면이나 실행 속도 측면에서 개선된 결과를 보이고 있다. 하지만, 지속적인 안전성 평가가 수행되어야 할 것으로 보이며, KpqC 공모전에 제안된 암호 알고리즘과의 성능 비교 연구와 추가적인 성능 개선 연구가 수행될 필요가 있다.

참 고 문 헌

[1] NIST PQC Standardization, https://csrc.nist.gov/projects/post-quantum-cryptography.

[2] KpqC 공모전, https://kpqc.or.kr/.

[3] J. Ryu, Y. Kim, S. Yoon, J.S. Kang, and Y. Yeom, “IPCC-Improved Perfect Code Cryptosystems”, *KpqC Round 1*, 2022, https://www.kpqc.or.kr/images/pdf/IPCC.pdf.

[4] M. Fellows, and N. Koblitz, “Kid krypto”. In *Advances in Cryptology—CRYPTO’92, 1992 Proceedings 12*, pp. 371-389. 1993.

[5] K.A. Shim, J. Kim, and Y. An, “MQ-Sign: A New Post-Quantum Signature Scheme based on Multivariate Quadratic Equations: Shorter and Faster”, *KpqC Round 1*, 2022, https://www.kpqc.or.kr/images/pdf/MQ-Sign.pdf.

[6] A. Kipnis, J. Patarin, and L. Goubin,“Unbalanced Oil and Vinegar signature schemes”, *Advances in Cryptology, CRYPTO'99, LNCS 1592*, pp. 206-222, 1999.

[7] K.A. Shim, S. Lee, N. Koo, “Efficient implementations of Rainbow and UOV using AVX2”, *IACR Trans. Cryptogr.Hardw Embed. Syst.vol. 2022*, no. 1,pp. 245-269, 2022.

[8] S. Kim, Y. Lee, and K. Yoon, “FIBS: Fast Isogeny Based Digital Signature”, KpqC Round 1, 2022, https://www.kpqc.or.kr/images/pdf/FIBS.pdf.

[9] J. Buchmann, E. Dahmen, and A. Hülsing, “XMSS-a practical forward secure signature scheme based on minimal security assumptions”. In *Post-Quantum Cryptography, PQCrypto 2011, Proceedings 4*, pp. 117-129, 2021.

[10] A. Hülsing, “W-OTS+ – shorter signatures for hash-based signature schemes”. In *Progress in Cryptology –AFRICACRYPT 2013, Proceedings 6*, pp. 173-188, 2013.

[11] D.X. Charles, K.E. Lauter, E.Z. Goren, "Cryptographic hash functions from exp-ander graphs", *Journal of Cryptology*, 22(1), pp. 93–113, 2009

[12] S. Kim, J. Ha, M. Son, B. Lee, D. Moon, J. Lee, ..., and J. Lee, "The AIMer Signature Scheme", *KpqC Round 1*, https://www.kpqc.or.kr/images/pdf/AIMer.pdf.

[13] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge from secure multiparty computation". In *Proceedings of the 39th annual ACM symposium on Theory of computing*, pp. 21-30, 2007.

[14] D. Kales, and G. Zaverucha, "Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures", *Cryptology ePrint Archive*, 2022.

[15] J. Don, S. Fehr, and C. Majenz, "The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more", In *Advances in Cryptology –CRYPTO 2020, Proceedings, Part III,* pp. 602-631, 2020.

[16] M. Dworkin, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, 2015, https://doi.org/10.6028/NIST.FIPS.202.

[17] L. Bettale, J.-C. Faugere and L. Perret, "Hybrid approach for solving multivariate systems over finite fields", *Journal of Mathmatic al Cryptology*, vol. 3, pp. 177-197, 2009.

[18] C. Wolf and B. Preneel, "Large superfluous keys in multivariate quadratic asymmetric systems", *Proc.of the International Conference on Practice and Theory of Public-Key Cryptography, LNCS 3386*, pp.275-287, 2005.

[19] E. Thomae, "About the security of multivariate quadratic public key schemes", *Dissertation Thesis, RUB*, June 2013.

[20] W. Beullens, "Improved cryptanalysis on UOV and Rainbow", *Advances in Cryptology, EUROCRYPT 2021, Part I, LNCS 12696*, pp. 348-373, 2021.

[21] A. Park, K. Shim, N. Koo, D. Han, "Side-Channel Attacks on Post-Quantum Signature Schemes based on Multivariate Quadratic Equations: Rainbow and UOV", *IACR Trans. Cryptogr. Hardw. Embed. Syst*. 2018(3), pp. 500-523, 2018.

[22] K. Sakumoto, T. Shirai, H. Hiwatari, "On provable security of UOV and HFE signature schemes against chosen-message attack", *Proc. of the International Conference on Post-Quantum Cryptography, LNCS 7071*, pp. 68-82, 2011.

[23] S.D. Galbraith, C. Petit, and J. Silva, "Identification protocols and signature schemes based on supersingular isogeny problems", *Journal of Cryptology*, 33(1), pp. 130-175, 2020.

[24] J. Buchmann, A. Pyshkin, and R.P. Weinmann, "Block Ciphers Sensitive to Gröbner Basis Attacks". In *CT-RSA (Vol. 3860*, pp. 313-331, 2006.

[25] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, "Efficient algorithms for solving overdefined systems of multivariate polynomial equations", In *Advances in Cryptology –EUROCRYPT 2000, Proceedings*, pp. 392-407, 2000.

〈저 자 소 개〉

석 병 진 (Byoungjin Seok)

종신회원

2017년 8월 : 서울과학기술대학교 컴퓨터공학과 졸업

2019년 2월 : 서울과학기술대학교 컴퓨터공학과 석사

2023년 2월 : 서울과학기술대학교 컴퓨터공학과 박사

2023년 3월~현재 : 서울과학기술대학교 전기정보기술연구소 선임연구원

<관심분야> 정보보호, 암호학, 디지털포렌식

엄 혜 진 (Hyejin Eom)

학생회원

2001년 2월 : 한양대학교 자연과학부 수학전공 학사
2003년 2월 : 한양대학교 자연과학부 통계학 석사
2021년 9월~현재 : 서울과학기술대학교 컴퓨터공학과 박사과정
<관심분야> 정보보호, 암호학, 양자내성암호

조 민 정 (Minjeong Cho)

학생회원

2018년 2월 : 서울과학기술대학교 컴퓨터공학과 학사
2020년 2월 : 서울과학기술대학교 컴퓨터공학과 석사
2020년 3월~현재 : 서울과학기술대학교 컴퓨터공학과 박사과정
<관심분야> 인증암호, 양자내성암호, 블록체인

이 창 훈 (Changhoon Lee)

종신회원

2001년 2월 : 한양대학교 자연과학부 수학전공 학사
2003년 2월 : 고려대학교 정보보호대학원 석사
2008년 2월 : 고려대학교 정보경영전문대학원 정보보호전공 박사
2008년 4월~2008년 12월 : 고려대학교 정보보호연구원 연구교수
2009년 3월~2012년 2월 : 한신대학교 컴퓨터공학부 조교수
2012년 4월~2015년 3월 : 서울과학기술대학교 컴퓨터공학과 조교수
2015년 4월~2019년 3월 : 서울과학기술대학교 컴퓨터공학과 부교수
2019년 4월~현재 : 서울과학기술대학교 컴퓨터공학과 교수
<관심분야> 정보보호, 암호학, 디지털포렌식, 사이버보안