NIST PQC Dilithium의 부채널 공격 및 대응 기법 동향 분석

유성환*, 한동국**

*,**국민대학교 (학부생, 교수)

Trends in Side-Channel Attacks and Countermeasures of NIST PQC Dilithium

Sung-Hwan Yoo*, Dong-Guk Han**

*,**Kookmin University(Undergraduate, Professor)

요 약

최근 양자 컴퓨팅 기술의 발전으로 인해 양자 내성 암호 (Post-quantum cryptography, PQC)에 대한 관심도가 높아지고 있다. 이에 따라, NIST는 2016년도부터 PQC 표준화 사업을 시작하였으며, 2025년 현재 KEM 1종(CRYSTALS-Kyber), 전자서명 3종(CRYSTALS-Dilithium, FALCON, SPHINCS+)을 선정하여 표준화를 진행하고 있다. 표준화에 따른 산업 활용도 측면에서 선정 알고리즘에 대한 안전성 분석은 지속적으로 연구되어야 하며, 부채널 관점에서의 안전성 평가 역시 필수적이다. 본 논문은 전자서명 부분의 표준으로 선정된 CRYSTALS-Dilithium에 대한 부채널 공격 및 대응 기법 연구에 대한 동향을 분석하고, 향후 Dilithium의 효율적인 대응 기법 적용 및 다른 알고리즘에 대한 부채널 분석 연구의 필요성을 제시한다.

I. 서론

최근 양자 컴퓨터 기술과 양자 알고리즘 기술의 발전으로 인해 현재 사용되고 있는 RSA와 ECC 등의 암호 알고리즘이 위협을 받고 있다. 양자 기술의 사용화 이후에도 안전한 암호 통신 체계를 구축하고자 양자 내성 암호(Post-Quantum Cryptograpy, PQC)에 대한 연구가 활발히 진행되고 있으며, 미국 국립표준기술연구소(NIST)에서는 2016년에 PQC 공모전을 시작했다. 2025년 5월 기준 PKE(Public Key Encryption)/KEM(Key Encapsulation Mechanism) 부분에서 CRYSTALS-Kyber(ML-KEM)를 표준으로 선정하였으며, 4라운드를 통해 HQC 알고리즘을 추가로 표준화 하고 있다. 전자서명 부분에서는 CRYSTALS-Dilithium(ML-DSA), Falcon(ML-DSA2), SPHINCS+(SLH-DSA)가 표준으로 선정되었다.

부채널 분석은 1996년 P.Kocker에 의해 처음 소개되었으며, 암호가 장비에서 동작할 때 누출되는 전력, 전자파 등의 부가적인 정보를 통하여 비밀정보를 누출하는 물리적 기법이다. 분석기법은 크게는 단순전력분석과 상관전력분석, 프로파일링 공격 기법이 존재하며, 대표적으로 SPA(Simple Power Analysis), DPA(Differential Power Analysis), Template Attack 등이 존재한다. 이러한 부채널 공격으로부터 비밀정보를 보호하기 위해 다양한 부채널 대응 기법이 연구되고 있으며, 대표적으로 암호 연산 시점을 랜덤하게 변경하는 하이딩 기법과 랜덤 값을 추가 연산 중간 값을 추측할 수 없도록 하는 마스크 기법이 존재한다.

PQC 공모전이 진행됨에 따라 각 알고리즘의 안전성 평가는 지속적으로 수행되어 왔으며 부채널 공격에 대한 안전성 검증 및 대응 기법 연구도 지속적으로 진행되어 왔다[1. 특히, 표준으로 선정된 알고리즘은 산업분야의 활용도 측면에서 공격 가능 지점과 그에 대한 대응 기법 연구에 대한 동향 정리는 필수이다.

본 논문에서는 전자서명에서 표준으로 선정된 CRYSTALS-Dilithium(ML-DSA)에 대한 부채널 공격 및 대응 기법 연구에 대한 동향을 소개한다. 논문의 구성은 다음과 같다. 2장에서는 Dilithium의 특징과 구조에 대해 설명한 후 3장에서는 Dilithium의 키 생성 과정에서 발생하는 부채널 공격과 대응 기법을 소개한다. 4장에서는 서명 과정에서 발생하는 부채널 공격과 대응 기법을 소개하고, 마지막으로는 Dilithium의 효율적인 대응 기법 적용 및 기타 표준 알고리즘에 대한 부채널 분석 연구 필요성을 제시한다.

II. 배경지식

CRYSTALS-Dilithium은 MLWE(Module Learning With Error)와 SIS(Short Integer Solution)를 기반으로 하는 격자 기반 전자서명 알고리즘이며, Uniform Smapling을 사용하여 상수 시간 연산을 보장한다[1]. 또한 다항식 환을 $Z_q[X]/(X^n+1)$과 같이 정의하며 $n=256$, $q=8{,}380{,}417$로 정의된다.

다음으로는 Dilithium의 키 생성, 서명 생성, 서명 검증 알고리즘을 의사코드 기반으로 설명한다. Fig.1은 키 생성 알고리즘의 의사코드이다.

Input :
Output : $pk=(\rho,t_1), sk=(\rho,K,tr,s_1,s_2,t_0)$
1. $\zeta \leftarrow \{0,1\}^{256}$
2. $(\rho,\rho',K)\in\{0,1\}^{256}\times\{0,1\}^{512}\times\{0,1\}^{256} := H(\zeta)$
3. $A\in R_q^{k\times l} :=$ ExpandA(ρ)
4. $(s_1,s_2)\in S_\eta^l\times S_\eta^l :=$ ExpandS(ρ')
5. $t := As_1+s_2$
6. $(t_1,t_0) :=$ Power2Round$_q(t,d)$
7. $tr\in\{0,1\}^{256} := H(\rho \,||\, t_1)$
8. return $(pk=(\rho,t_1), sk=(\rho,K,tr,s_1,s_2,t_0))$

Fig. 1. Key Generation Pseudocode

키 생성에서는 먼저 난수 생성기를 사용하여 임의의 난수 ζ를 생성 한 후, Keccak의 SHAKE 함수로 구성된 H함수를 통해 seed (ρ,ρ',K)를 생성한다. 생성된 seed(ρ,ρ',K) 중 ρ를 통해 행렬 A를 생성하며, ρ'은 비밀 값 s_1, s_2, 그리고 K는 서명 과정에서의 비밀 값 y를 생성할 때 사용된다. 따라서 난수 ζ는 비밀 값으로 유지되어야 한다. 이후 A와 s_1, s_2를 통하여 t를 생성한다.

Fig.2는 서명 알고리즘의 의사 코드이다.

Input : $sk=(\rho,K,tr,s_1,s_2,t_0), m$
Output : $\sigma=(\tilde{c},z,h)$
1. $A\in R_q^{k\times l} :=$ ExpandA(ρ)
2. $\mu\in\{0,1\}^{512} := H(tr \,||\, m)$
3. $\kappa := 0, (z,h) := \perp$
4. $\rho'\in\{0,1\}^{512} := H(K \,||\, \mu)$
5. **while** $(z,h)=\perp$ **do**
6. $\quad y\in S_{\gamma_1}^l :=$ ExpandMask$(\rho' \,||\, \kappa)$
7. $\quad w := Ay$
8. $\quad w_1 :=$ HighBits$(w, 2\gamma_2)$
9. $\quad \tilde{c}\in\{0,1\}^{256} := H(\mu \,||\, w_1)$
10. $\quad c\in B_\tau :=$ SampleBall$(\tilde{c})$
11. $\quad z := y+cs_1$
12. $\quad r_0 :=$ LowBits$_q(w-cs_2, 2\gamma_2)$
13. $\quad$ **if** $\|z\|_\infty\geq\gamma_1-\beta$ **or** $\|r_0\|_\infty\geq\gamma_2-\beta$, **then** $(z,h):=\perp$
14. $\quad$ **else**
15. $\quad\quad h :=$ MakeHint$_q(-ct_0, w-cs_2+ct_0, 2\gamma_2)$
16. $\quad\quad$ **if** $\|ct_0\|_\infty\geq\gamma_2$ **or** the # of 1's in h in greater then ω, then $(z,h)=\perp$
17. $\quad \kappa := \kappa+l$
18. return $\sigma=(\tilde{c},z,h)$

Fig. 2. Signature Generation Pseudocode

서명 생성 단계에서는 비밀키 값 중 K을 활용하여 y를 생성한다. 이후 공개 행렬 A와 곱셈을 수행하여 w를 생성한 후 반올림 연산을 통해 상위 비트 w_1을 생성한다. w_1은 서명 데이터 c를 생성하는데 활용되며, $y+cs_1$ 연산을 통해 z를 생성한다. 이후 Fiat-Shamir with Aborts 설계 방식을 활용하여 생성된 w, z, c의 유효성을 판단한 후 유효한 경우 MakeHint 함수를 활용하여 마지막 서명 데이터 h를 생성한다.

Fig.3은 서명 검증 알고리즘이다.

Input : $pk=(\rho,t_1), \sigma=(\tilde{c},z,h)$
Output : True or False
1. $A\in R_q^{k\times l} :=$ ExpandA(ρ)
2. $\mu\in\{0,1\}^{512} := H(H(\rho \,||\, t_1) \,||\, M)$
3. $c :=$ SampleInBall$(\tilde{c})$
4. $w_1' :=$ UseHint$_q(h, Az-ct_1\cdot 2^d, 2\gamma_2)$
5. return $[\![\|z\|_\infty<\gamma_1-\beta]\!]$ and $[\![\tilde{c}=H(\mu \,||\, w_1')]\!]$ and $[\![$#of 1's in h is $\leq\omega]\!]$

Fig. 3. Verify Signature Pseudocode

서명 검증 단계에서는 공개키 데이터와 메시지를 통해 공개 행렬 A와 μ를 생성한다. 이후 생성한 데이터와 서명 데이터, SampleInBall, UseHint 함수를 통해 c, w'_1을 생성하고, 마지막으로 z와 $\tilde{c}, w'_1$의 범위 확인을 통해 서명 검증을 마무리 한다.

III. 키 생성에서의 부채널 분석 동향

본 장에서는 Dilithium 키 생성 과정에서 발생하는 부채널 공격 및 이에 대한 대응 기법 동향을 소개한다.

키 생성 과정은 암호화 통신에서 키 생성 혹은 갱신에서만 사용되기에 수행 횟수가 매우 제한적이다. 전자서명에서 역시 이와 동일하며, 이미 생성된 키를 기반으로 서명 생성 및 검증을 수행한다. 따라서 키 생성 알고리즘을 타겟으로 수행되는 공격은 단일 수행 공격 특성을 띄고 있다. Fig.4는 키 생성 알고리즘에 공개 데이터와 비밀 데이터를 표기한 사진으로 공개 정보는 초록색, 비밀 정보는 빨간색으로 표기하였다.

Input :
Output : $pk = (\rho, t_1), sk = (\rho, K, tr, s_1, s_2, t_0)$
1. $\zeta \leftarrow \{0,1\}^{256}$
2. $(\rho, \rho', K) \in \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} := H(\zeta)$
3. $A \in R_q^{k \times l} := \text{ExpandA}(\rho)$
4. $(s_1, s_2) \in S_\eta^l \times S_\eta^l := \text{ExpandS}(\rho')$
5. $t := As_1 + s_2$
6. $(t_1, t_0) := \text{Power2Round}_q(t, d)$
7. $tr \in \{0,1\}^{256} := H(\rho \,||\, t_1)$
8. return $(pk = (\rho, t_1), sk = (\rho, K, tr, s_1, s_2, t_0))$

Fig. 4. Notation of Public and private values in Key Generation Pseudocode

3.1 H 함수에 대한 SASCA

Fig.4의 line 2에서는 ζ값과 H 함수를 활용하여 seed(ρ, ρ', K)을 생성한다. 공식 문헌에[1] 따르면 H함수는 Keccak 기반의 확장 출력 함수인 SHAKE-256함수를 사용한다. 생성된 seed(ρ, ρ', K) 값 중 ρ' 값을 ExpansS의 seed로 사용되어 비밀 값 s_1, s_2를 생성한다. H 함수와 ExpandS 함수의 코드는 공개 함수이고, ζ와 ρ' 값의 연관성으로 인해 ζ값 노출 시 비밀 값인 s_1, s_2가 공격자에게 노출되게 된다. 따라서 ζ값은 비밀 정보로 유지되어야 한다.

하지만 2020년 Matthias J. Kannwischer 등은 Keccak 함수를 대상으로 단일 부채널 공격을 통해 seed 값을 도출하는 공격 기법을 제안했다[2]. 본 공격은 Keccak 함수의 sponge 구조에서 사용되는 Keccak-f 함수를 대상으로 수행되며, Soft Analysis Side Channel Attack (SASCA) 공격 기법을 같이 사용하였다. SASCA 공격 기법은 암호의 원시 연산 구현을 팩터 그래프로 모델링 한 후 측정한 전력 값을 기반으로 Belief Propagation 알고리즘을 사용하여 확률적으로 비밀 정보를 추론하는 공격 기법이다.

본 문단에서는 공격 지점에 대해 설명한다. Keccak 함수는 sponge 구조를 따르며 해당 구조는 Keccak-f 함수와 XOR연산을 기반으로 수행된다. Fig.5는 sponge 구조를 도식화한 그림이며 absorbing과 squeezing 절차로 구분된다.

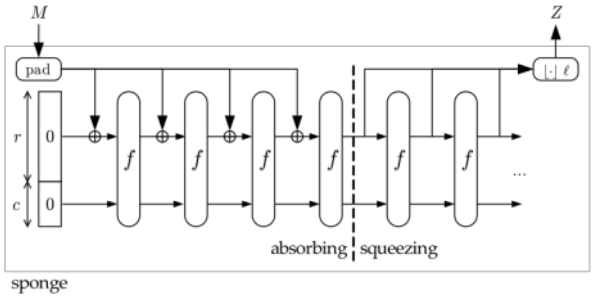

Fig. 5. Keccak sponge structure

Fig.6은 Keccak-f 함수 내부의 5가지 연산 $(\theta, \rho, \pi, \chi, \iota)$ 중 θ, ρ, π, χ을 나타낸다.

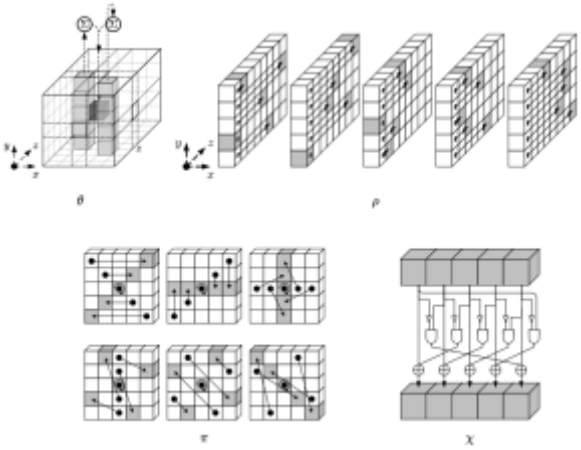

Fig. 6. Function θ, ρ, π, χ

공격자는 내부 함수 수행 결과 값을 RAM에 저장하는 시점에 HW(Hamming Weight) 기반 누설 정보를 측정한다. Table.1은 공격자가 전력 정보를 측정하는 위치에 대한 정보를 나타낸다.

위치	RAM에 저장되는 데이터
θ 입력	라운드 시작 시점의 전체 state
θ 출력, 패리티 평면	θ 계산 결과, θ 중간 XOR 결과(패리티 평면)
$\pi \circ \rho$ 입출력	state를 위치 이동 및 bit rotation 결과 값
χ 입출력	state를 논리 연산으로 처리한 값

Table. 1. Attack location information

측정된 정보는 팩터 그래프의 팩터 노드로 입력된 후 Belief Propagation 알고리즘을 통해 비밀 정보를 추론한다.

3.2 H 함수에 대한 대응 기법

키 생성에서 사용되는 H 함수의 경우 단일 수행 공격에 대응되어야 하기에 통상적으로 셔플링 및 하이딩 기법이 사용된다. 하지만 이에 대한 구체적인 대응 기법 연구가 미흡하여 Keccak을 대상으로 하는 부채널 공격의 대응 기법 연구가 필요하다.

3.3 NTT 곱셈 함수에 대한 탬플릿 기반 SASCA

Fig.4의 line 5에서는 비밀 값 s_1과 공개 행렬 A와의 NTT 곱셈 연산을 수행한다. 2017년 Primas, R 등은 SASCA 공격 방법을 활용하여 단일 파형 기반의 템플릿 공격을 통해 NTT 곱셈 연산에서의 비밀키 복구를 성공하였다[3]. 해당 논문은 Kyber의 INTT 연산을 대상으로 공격을 수행하였지만 Dilithium의 키 생성 과정에서의 NTT 곱셈 역시 본 공격과 동일한 환경이기에 충분히 공격 가능하다.

본 공격은 NTT 곱셈 중 단일 Cooley-Turkey(CT) 버터플라이 곱셈 연산을 대상으로 수행되었다. Fig.7의 (a)는 해당 곱셈 연산을 도식화한 그림이다.

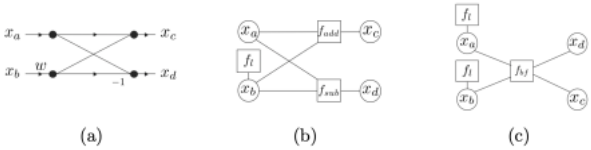

Fig. 7 CT butterfly show in (a), Factor Graph of CT butterfly in (b), optimized Factor Graph in (c).

공격자는 해당 곱셈 연산을 기반으로 팩터 그래프를 모델링 하며 Fig.7의 (b)는 CT 버터플라이 곱셈 연산을 기반으로 모델링된 팩터 그래프이다. 이후 해당 그래프를 통해 사전에 필요한 탬플릿을 생성한다. (b)의 팩터 그래프에서는 약 100만개의 탬플릿이 필요하며 이를 위해 1억개의 파형을 수집한다. 본 논문에서는 해당 프로파일링 단계에서 복잡도가 크기에 HW 기반 탬플릿 사용과 전력 수집 지점을 연산 입/출력 시 메모리 접근 지점으로 변경하여 복잡도 문제를 해결하였다. Fig.7의 (c)는 논문에서 제안된 팩터 그래프로 기존 100만개의 필요 탬플릿 수를 213개로 줄여 프로파일링 복잡도를 해결하였다.

프로파일링 단계 이후 측정한 전력 값과 사전 탬플릿 정보를 기반으로 계산된 후보 값의 확률을 팩터 노드로 입력한다. 이후 Belief Propagation 알고리즘을 통해 가장 높은 확률 값을 가진 후보 값을 비밀 정보를 추론한다.

3.4 NTT 곱셈 함수에 대한 대응 기법

NTT 곱셈 연산 지점은 키 생성 알고리즘 단계에서 수행하기에 단일 수행 공격에 대한 대응기법이 적용되어야 한다. 따라서 셔플링 및 하이딩과 같은 대응기법이 필요하다.

2020년 Prasana Ravi 등은 NTT 곱셈 연산에 대한 셔플링 대응 기법으로 3가지 방법을 제안했다[4]. Table. 2는 3가지 셔플링에 대한 정보를 나타낸다.

구분	Coarse-Full	Coarse-In-Group	Fine
보안성	매우 높음	높음	중간
성능	느림	중간	빠름
엔트로피	$(n/2)!$	$((n/2m)!)^m$	2^{2n}

Table. 2. Countermeasure Method Information
(n : input vector size, m : butterfly group count)

NTT 각 stage의 버터플라이 연산은 독립적으로 계산될 수 있다. Coarse-Full 셔플링 기법은 이러한 특징을 활용하여 각 stage에서 발생하는 $n/2$개의 버터플라이 연산 순서를 무작위로 셔플링한다. 셔플 알고리즘은 이전 연구[3,5]에서도 부채널 대응 기법으로 사용된 Knuth-Yates 알고리즘을 사용한다[6]. 또한 [7]에서는

LFSR을 사용한 셔플링 방법을 제안했지만, 생성 가능한 순열 개수가 제한되기에 대규모 공격에 취약하다. Coarse-In-Group 셔플링 기법은 각 stage에서 버터플라이 연산을 그룹별로 구분한 뒤 그룹 내에서 셔플링을 수행한다. 해당 방법은 Coarse-Full보다는 적은 엔트로피를 가지지만 현실적은 무작위 공격은 불가능하다. 또한 CT 버터플라이 특성상 그룹 내의 버터플라이는 동일한 twiddle factor를 사용하기에 그룹내 연산 시 1회만 메모리에서 로드하면 된다는 장점이 있다.

Fine 셔플링 기법은 기존 NTT 단일 수행 공격 발생 지점인 메모리 Load 및 Store 버터플라이의 입력 값이 메모리에 Load되는 과정에서 부채널 공격이 수행되었다. 따라서 Fine 셔플링 기법은 메모리 Load 및 Store 순서를 무작위로 셔플링 하여 공격에 대응한다. 하지만 이는 기존 Primas 등이 제안한 공격이 유효하기에 다른 대응 기법과 조합하여 사용된다.

IV. 서명 생성에서의 부채널 분석 동향

본 장에서는 Dilithium 서명 생성 과정에서 발생하는 부채널 공격 및 이에 대한 대응 기법 동향을 소개한다.

서명 생성 과정은 키 생성 과정과 다르게 고정된 비밀 키 값을 기반으로 수행되기에 다중 수행 공격도 가능하다. Fig.8은 서명 생성 알고리즘에 공개 데이터와 비밀 데이터를 표기한 사진으로 공개 정보는 초록색, 비밀 정보는 빨간색으로 표기하였다.

4.2 Fiat-Shamir with abort에 대한 탬플릿 기반 SASCA

Fig.8의 line 13에서는 서명 생성 이후 특정 조건을 만족하지 않는 경우 서명을 다시 생성하는 과정이 존재한다. 이는 암호학적 안전성을 위해 필요한 과정이며 이를 Dilithium에서는 Fiat-Shamir with Aborts 구조라고 부른다. 2025년 Zheng Liu 등은 Fiat-Shamir with Aborts 과정에서 생성되는 거부된 서명 값과

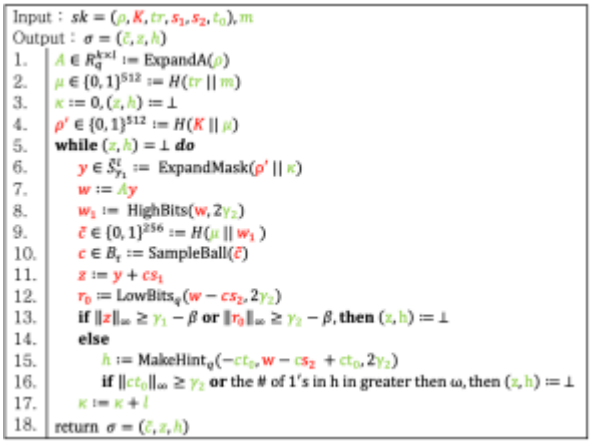

Fig. 8. Notation of Public and private values in Signature Generation Pseudocode

유효한 서명 값을 사용하여 개인 키를 복구하는 방법을 제안하였다[8]. Fig.9는 해당 논문에서 사용한 탬플릿 기반 SASCA 기법을 도식화한 사진이다.

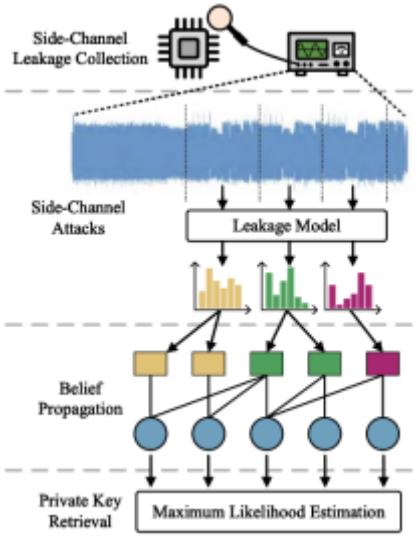

Fig. 9. Process of Template based SASCA

공격자는 공격에 사용할 중간 값을 선정한 후 중간 값들의 관계성을 기반하는 팩터 그래프를 생성한다. 이후 프로파일링 단계에서 팩터 그래프에 필요한 탬플릿을 생성한 뒤 실제 공격 시, 단일 전력 측정과 Belief Propagation 알고리즘을 통해 개인 키를 복구한다.

공격에 사용되는 중간 값으로는 유효한 서명 생성에서 연산되는 $y, c \cdot s_1$와 거부된 서명 생성

에서 연산되는 $\bar{c}, \bar{c} \cdot s_1$이 사용된다. Fig.10은 중간 값들의 연산 관계를 기반으로 모델링한 팩터 그래프이다.

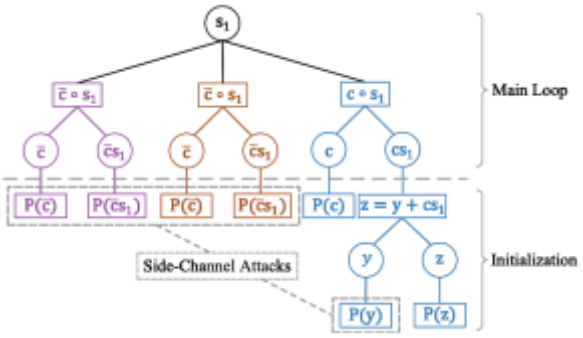

Fig. 10. Factor Graph

Initialization 단계의 c와 z 값은 유효한 서명 값 중 일부이기에 공격자가 사전에 알고있는 정보이다. 따라서 $P(c)$와 $P(z)$ 값은 0 또는 1로 주어진다. 그 외의 확률 값은 프로파일링 단계에서 생성된 탬플릿을 통해 도출하고, 각 확률 값과 Belief Propagation 알고리즘을 통해 가장 높은 확률 값을 가진 후보 값을 비밀 키로 추론한다.

4.3 NTT 곱셈 함수에 대한 CPA

Fig.8의 line 11, 12에서는 비밀 값 c와 s_1, s_2와의 NTT 곱셈 연산을 수행한다. 서명 생성에서는 다중 전력 수집이 가능하기에 CPA 혹은 DPA와 같은 통계 기반 부채널 분석 공격이 가능하다. 2021년 Zhaohui Chen 등은 NTT를 대상으로 수행되는 고전적인 CPA 공격 기법을 개선한 효율적인 공격 기법을 제안하였다[9]. Fig.11은 NTT 곱셈 과정 중 Mongomery modulus 곱셈을 수행하는 의사코드이다.

```
Input : t, c, s
Output : t
1.  unsigned int i;
2.  for(i = 0; i < 256; ++i){
3.      int64_t a;
4.      a = (int64_t )c[i] · s[i];      Point1
5.      t = (int32_t )a · qinv;
6.      t[i] = (a − (int64_t)t · q) ≫ 32;      Point2
7.  }
```

Fig. 11. Mongomery modulus multiplication pseudocode

고전적인 CPA는 Fig.11의 Point2 지점만을 대상으로 공격을 수행하였다. 하지만 Dilithium 키 특성상 256차 다항식 형태이기에 모든 후보 키를 대상으로 수행하기에는 많은 시간이 소요된다. 이러한 문제점을 개선하고자 Zhaohui Chen 등은 Point2를 대상으로 CPA를 수행하기 전 Point1을 통한 CPA로 후보키 범위를 제한한다. Point1을 통한 CPA는 NTT 곱셈 결과 값 64bit 중 하위 L bit를 대상으로 CPA를 수행하며 후보키를 선택한다. 이후 해당 후보키를 대상으로 Point2를 대상으로 CPA 공격을 수행하며 후보키 개수가 감소된 만큼 공격 속도는 빨라진다.

실제 수행 결과 후보키 개수(n_{add}), CPA 파형 수(No. Traces) 그리고 하위 L bit 길이(p)에 따라 수행 시간이 달라진다. Fig.12는 각 변수 값에 따른 공격 속도를 나타낸다.

Scheme	No. Traces	n_{cdd}	p	Hit Rate[1]	Acceleration
Conservative	157	NA	NA	NA	1.0
Hybrid *a*	1500	80	15	77.73%	3.19
Hybrid *b*	3000	40	14	93.75%	6.56
Hybrid *c*	5000	24	13	92.97%	6.34
Hybrid *d*		8		91.02%	6.63
Hybrid e	10000	12	12	96.10%	**7.77**
Hybrid *f*		16		**96.48%**	6.83
Hybrid *g*	15000	8	12	94.53%	7.20
Hybrid *h*	20000	6	11	94.53%	7.06

Fig. 12. CPA Attack runtime

하위 12bit를 대상으로 파형 10,000개의 CPA를 수행한 결과 12개의 후보키로 제한되었을 때 (Hybrid e) 속도가 가장 많이 향상되었다. 하지만 본 공격 방법은 Point 1 CPA 공격에서 실제 키 값이 후보로 선택되지 않은 경우 공격에 실패한다는 한계점을 가지고 있다.

4.4 서명 생성의 다중 수행 공격에 대한 대응 기법

서명 생성의 경우 키 생성과 다르게 다중 수행 공격이 가능하기에 셔플링과 하이딩 대응 기법 보다 마스킹과 같은 대응기법이 적용되어야 한다.

2023년 Jean Sébastien Coron 등은 서명 생성 과정에서 적용 가능한 마스킹 기법을 새롭게 제안하였다[10]. Dilithium 연산 특성상 산술 마스킹과 부호 마스킹이 함께 적용되기에 마스

킹을 변환하는 함수와 마스킹이 적용된 modular shift 연산을 수행하는 ShiftMod 가젯 그리고 마스킹 share 값의 modular를 변환하는 ModSwitch 가젯을 새롭게 제안하였다. 마스킹 변환 함수는 ShiftMod와 ModSwitch 가젯을 활용하여 Fig.8의 line 6에서 변수 y를 생성할 때 활용된다. 또한 서명 생성 과정에서 상위 비트와 하위 비트를 나누는 Decompose 함수에서도 두 가젯을 모두 활용하여 기존 방법보다 효율적인 함수를 제안한다.

2024년 Jean Sébastien Coron 등은 기존 [11]에서 제안한 마스킹 변환 함수보다 공간 복잡도를 개선한 새로운 알고리즘을 제안하였으며 [8]에서 제안한 Fiat-Shamir with Aborts 대응 기법보다 시간 복잡도에서 개선된 알고리즘을 제안하였다. 기존 마스킹 변환 함수는 변환 과정 중 q에 대한 modular 연산을 한 번에 수행하기 위해 큰 레지스터를 사용하였다. 하지만 해당 논문에서는 2^k에 대한 작은 modular을 먼저 수행 한 후 ShiftMod 가젯을 통해 q에 대한 modular로 변환 한다. 이로 인해 연산 수행 레지스터 크기가 64bit에서 32bit로 감소하여 공간 복잡도 측면에서 효율적인 알고리즘을 제시하였다. 또한 기존 [12]에서 제안된 Fiat-Shamir Abort 대응 기법은 유효한 서명 검증 과정 중 대소 비교를 위해 마스킹 변환을 수행하였다. 하지만 마스킹 변환 과정에서의 시간 복잡도 오버헤드가 크기에 해당 논문에서는 ShfitMod 가젯을 활용하여 대소 비교를 수행하는 알고리즘을 새롭게 제안하였다. 실제 연산 시간 측정 결과 Cycle counts에서 약 40% 정도 더 빠른 연산 수행을 보이는 것을 확인하였다.

V. 결론

본 논문은 NIST PQC 전자서명에서 표준으로 선정된 CRYSTALS-Dilithium에 대한 부채널 공격 및 대응 기법을 소개한다. 현재까지 다양한 함수를 대상으로 부채널 공격이 연구되었으며 이에 대한 대응 기법 역시 많은 연구가 진행되었다. 대응 기법 특성상 많은 오버헤드를 유발하기에 키 생성 과정에서는 마스킹보다는 단일 공격의 대응할 수 있는 셔플링과 하이딩 기법을 적용하고, 서명 생성 과정에서는 마스킹과 같은 대응 기법이 필요하다. 표준으로 선정된 만큼 산업 분야에서도 많은 활용이 진행될 것이며 안전성 측면에서 부채널 공격의 검증 및 대응 기법 연구에 지속적인 관심이 필요하다.

[참고문헌]

[1] Shi Bai, Léo Ducas, Eike Kiltz, et al, CRYSTALS-Dilithium, NIST PQC August, 2022.

[2] Matthias J. Kannwischer, Peter Pessl and Robert Primas, Signle-Trace Attacks on Keccak, TCHES, April, 2020.

[3] Robert Primas, Peter Pessl and Stefan Mangard, Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption, CHES, Agust, 2017.

[4] Prasanna Ravi, Romain Poussier, Shivam Bhasin and Anupam Chattopadhyay, On Configurable SCA Countermeasures Against Single Trace Attacks for the NTT, IACR, December, 2020.

[5] Petter Pessl and Rober Primas, More Practical Single-Trace Attacks on the Number Theoretic Transfrom, Latincrypt, July, 2019.

[6] Sujoy Sinha Roy, Oscar Reparaz, Frederik vercauteren and Ingrid Verbauwhede, Compact and Side Channel Secure Discrete Gaussian Sampling, IACR ePrint, October, 2014.

[7] Timo Zijlstra, Karim Bigou and Arnaud Tisserand, FPGA Implementation and Comparison of Protections against SCAs for RLWE, International Conference on Cryptology in India, October 2019.

[8] Zheng Liu, An Wang, Congming Wei,

Yaoling Ding, Jingqi Zhang, Annyu Liu and Liehuang Zhu, Release the power of Rejected Signatures: An Efficient Side-Channel Attack on Dilithium, Cryptology ePrint, April, 2025.

[9] Zhaohui Chen, Emre Karabulut, Aydin Aysu, Yuan Ma and Jisu Jing, An Efficient Non-Profiled Side-Channel Attack on the CRYSTALS-Dilithium Post-Quauntum Signature, IEEE, October, 2021.

[10] Jean Sébastien Coron, François Gérard, Matthias Trannoy and Rina Zeitoun, Improved Gadgets for the High-Order Masking of Dilithium, TCHES, November, 2023.

[11] Jean Sébastien Coron, François Gérard, Tancrède Lepoint, Matthias Trannoy and Rina Zeitoun, Improved High-Order Masked Generation of Masking Vector and Rejection Sampling in Dilithium, TCHES, November, 2024.

[12] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Méelissa Rossi and Mehdi Tibouchi, Masking the GLP Lattice-Based Signatrue Scheme at Any Order, EUROCRYPT, March, 2018.