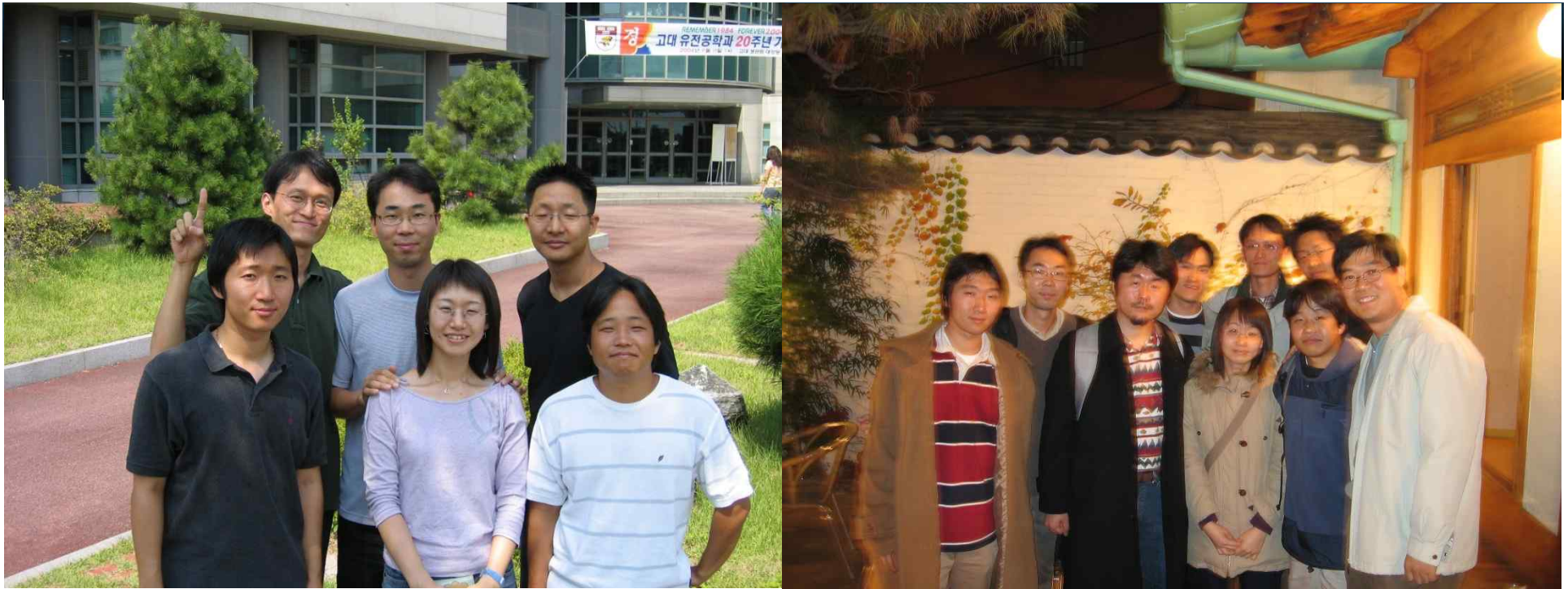


PQC 알고리즘 부채널 분석 및 대응 기술 동향

2022년 9월 30일 (금)
국민대학교 정보보안암호수학과
한동국 교수 (christa@kookmin.ac.kr)

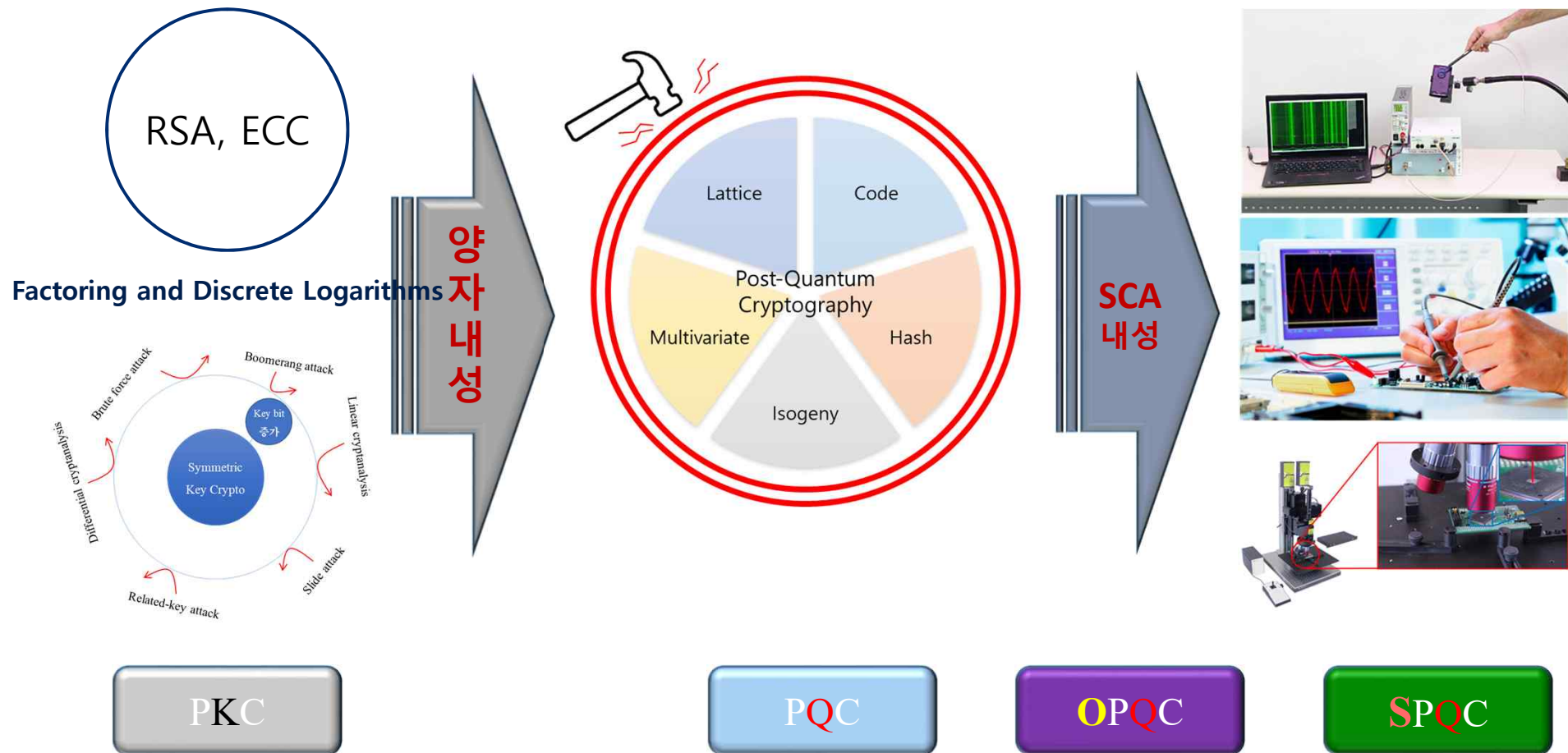
PQC 알고리즘 부채널 분석 및 대응 기술 동향

2022년 9월 30일 (금)
국민대학교 정보보안암호수학과
한동국 교수 (christa@kookmin.ac.kr)



- ❖ 2002.03 - 2005.02 고려대학교 정보보호대학원 공학박사
- ❖ 2004.06 - 2005.06 일본 큐슈대학교 교환학생
- ❖ 2005.04 - 2006.04 일본 Future Univ.-Hakodate Post.Doc.
- ❖ 2006.06 - 2009.02 ETRI 정보보호본부 선임연구원
- ❖ 2009.03 - 2021.현재 국민대학교 정보보안암호수학과 교수

From PKC to PQC



1 부채널 분석 관점에서의 PKC & PQC

2 PQC 알고리즘에 대한 부채널 분석 동향

3 상수시간 알고리즘은 항상 안전한가?

4 최적화 구현과 부채널 분석 취약점의 상관관계

- ✓ 1 부채널 분석 관점에서의 PKC & PQC
- 2 PQC 알고리즘에 대한 부채널 분석 동향
- 3 상수시간 알고리즘은 항상 안전한가?
- 4 최적화 구현과 부채널 분석 취약점의 상관관계

Side Channel Analysis for Cryptographic Device

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 388-397, 1999.
© Springer-Verlag Berlin Heidelberg 1999

Differential Power Analysis

Paul Kocher, Joshua Jaffe, and Benjamin Jun



graphY Research, Inc.
ket Street, Suite 1088
cisco, CA 94102, USA.
www.cryptography.com
osh,ben}@cryptography.com.

signers frequently assume that secrets will
reliable computing environments. Unfortun-
microchips leak information about the op-
er examines specific methods for analyz-
measurements to find secret keys from tamper
russ approaches for building cryptosystems
that can operate securely in existing hardware that leaks information.

Keywords: differential power analysis, DPA, SPA, cryptanalysis, DES



부채널 분석 (SCA, Side Channel Analysis)

❖ 암호 알고리즘이 동작할 때 의도치 않게 발생하는 **부채널 정보**를 활용하여 **비밀 값(Key)**을

분석하는 방법



전자 주민등록증



전자 여권



전자 공무원증



Smart Car



USIM



금융IC카드

Non-Invasive Attacks


 비침입

- ❖ 디바이스에서 암호 알고리즘 동작 시 발생하는 **부가적인 정보**(소비전력, 전자기 신호 등)를 수집·분석하여 비밀 정보를 획득하는 분석기법

➤ Ex) 단순 전력 분석, 상관 전력 분석



Acoustic Side Channel Attack

DUT



- Simple Acoustic Channel Attack

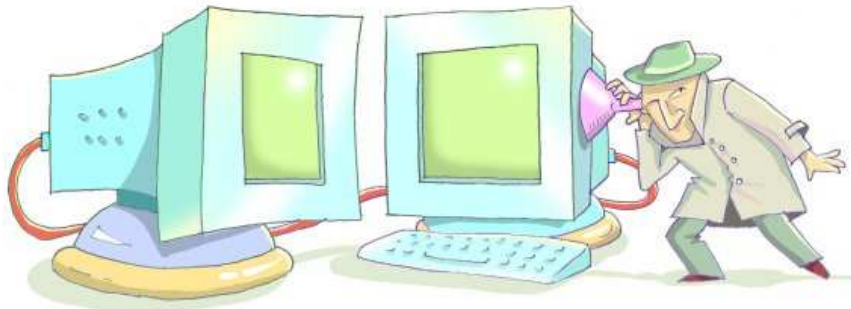


- Differential Acoustic Channel Attack

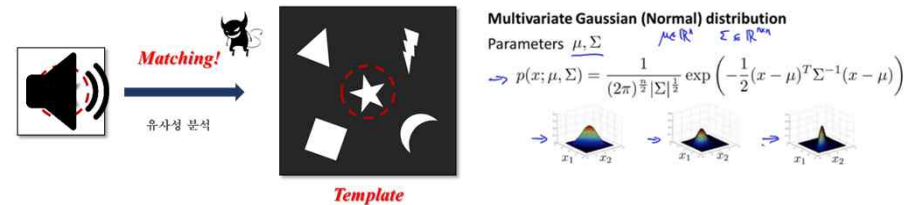


Profiling DUT

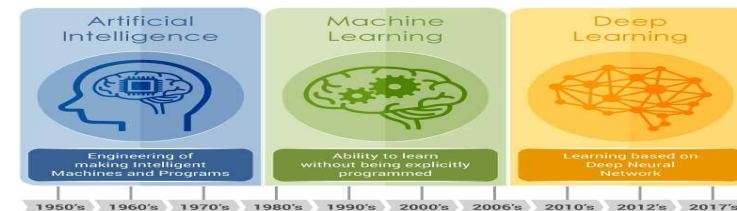
DUT



- Template Acoustic Channel Attack



- ML-based Acoustic Channel Attack



Public Key Cryptosystems

❖ RSA

- p, q : primes
- $n = p \times q$
- $ed \equiv 1 \pmod{(p-1)(q-1)}$
- ✓ e, n : public key, d : secret key, (factoring, n : 1024 bits)
- ✓ M : message, $M \in \{0, 1, 2, \dots, n-1\}$



Adi Shamir Ron Rivest Leonard Adleman

Encryption: $C \equiv M^e \pmod{n}$

e : small ($2^{16} + 1$), FAST.

Decryption: $M \equiv C^d \pmod{n}$

d : large ($d > n^{\frac{1}{2}}$), SLOW.

■ SCA on RSA: Non-Invasive Attack

❖ 조건문 사용에 따른 Operation 실행 여부 패턴 확인

1996 [Crypto] Timing Attacks

1999 [Crypto] Simple Power Analysis

Algorithm. Left to Right Binary Method

Input $M, N, k = (k_{n-1}, k_{n-2}, \dots, k_0)_2$

Output $M^k \bmod N$

Step 1. $R = 1$

Step 2. For $i = n - 1$ down to 0 do

2.1. $R = R \times R \bmod N$

2.2. If $k_i = 1$ then $R = R \times M \bmod N$

Step3. Return R



Mathematical proof

SCA on RSA: Non-Invasive Attack

❖ 조건문 사용에 따른 Operation 실행 여부 패턴 확인

1996 [Crypto] Timing Attacks

1999 [Crypto] Simple Power Analysis

Countermeasure

→ 조건문 제거

Mathematical proof

Algorithm. Left to Right Binary Method

Input $M, N, k = (k_{n-1}, k_{n-2}, \dots, k_0)_2$

Output $M^k \bmod N$

Step 1. $R = 1$

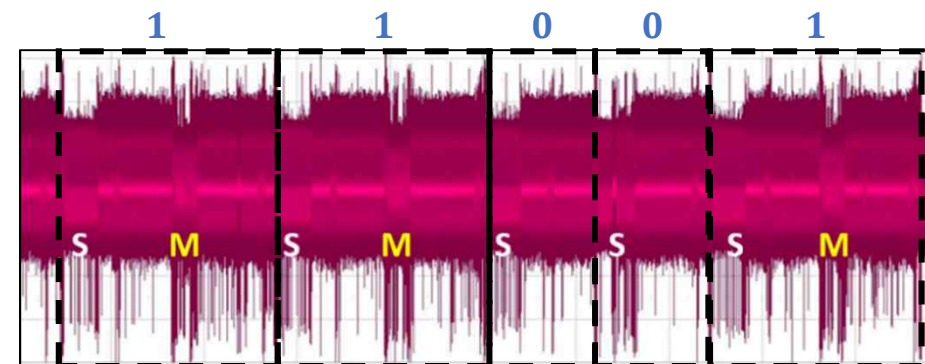
Step 2. For $i = n - 1$ down to 0 do

2.1. $R = R \times R \bmod N$

2.2. If $k_i = 1$ then $R = R \times M \bmod N$

Step3. Return R

곱셈 연산은 연속적으로 발생할 수 없음



S : Square, M : Multiplication

SCA on RSA: Non-Invasive Attack

❖ 중간값(intermediate values)에 따른 통계적 분석

1996 [Crypto] Timing Attacks

1999 [Crypto] Simple Power Analysis

[CHES] Differential Power Analysis

Countermeasure

➔ Randomization Method 적용

SPA Countermeasure

Mathematical proof

Algorithm. Left to Right Binary Method

Input $M, N, k = (k_{n-1}, k_{n-2}, \dots, k_0)_2$

Output $M^k \bmod N$

Step 1. $R_0 = 1$

Step 2. For $i = n - 1$ down to 0 do

2.1. $R_0 = R_0 \times R_0 \bmod N$

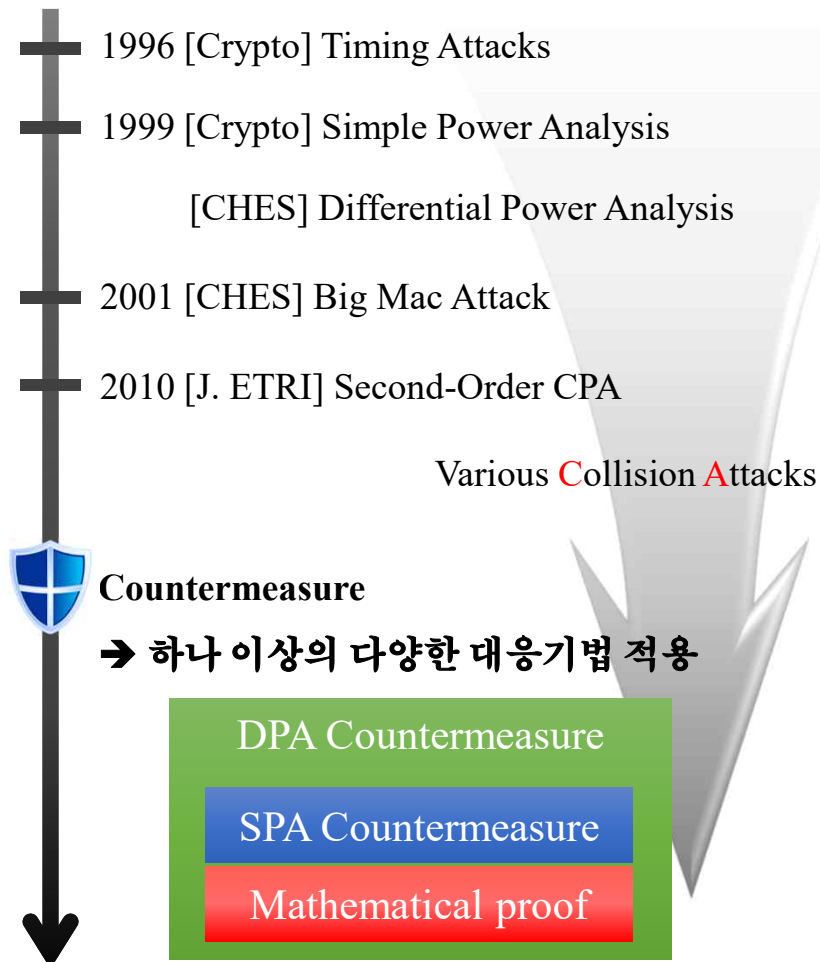
2.2. $R_{1-k_i} = R_0 \times M \bmod N$

Step3. Return R_0



SCA on RSA: Non-Invasive Attack

❖ 데이터 간의 상호 관계



Algorithm. Left to Right Binary Method

Input $M, N, k = (k_{n-1}, k_{n-2}, \dots, k_0)_2$

Output $M^k \bmod N$

Step 1. $R_0 = 1$

Step 2. For $i = n - 1$ down to 0 do

2.1. $R_0 = R_0 \times R_0 \bmod N$

2.2. $R_{1-k_i} = R_0 \times M \bmod N$

Step 3. Return R_0

+ data / exponent blinding



SCA on RSA: Non-Invasive Attack

❖ 키 비트 확인 단계

- 1996 [Crypto] Timing Attacks
- 1999 [Crypto] Simple Power Analysis
- [CHES] Differential Power Analysis
- 2001 [CHES] Big Mac Attack
- 2010 [J. ETRI] Second-Order CPA
- 2017 [ISPEC] Key Bit-dependent Attack



DPA Countermeasure

DPA Countermeasure

SPA Countermeasure

Mathematical proof

Algorithm. Left to Right Binary Method

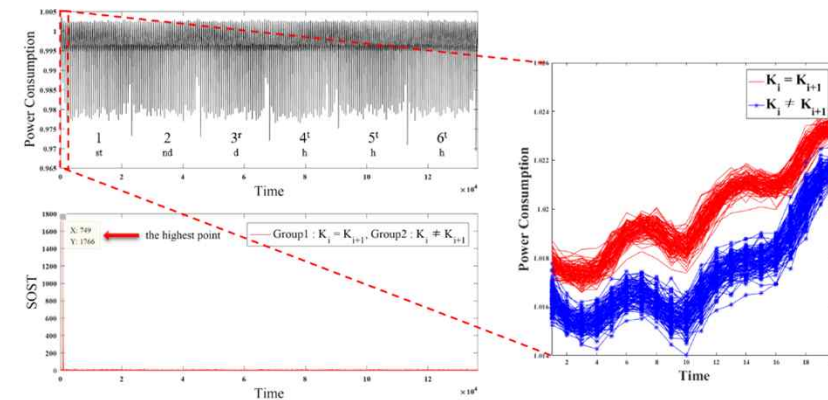
Input	$M, N, k = (k_{n-1}, k_{n-2}, \dots, k_0)_2$
Output	$M^k \bmod N$

Step 1. $R_0 = 1$ Step 2. **For $i = n - 1$ down to 0 do**

Countermeasure

Step3. Return R_0

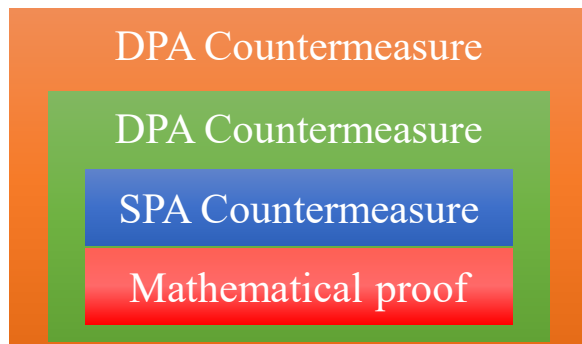
키 비트 값이 추출되어 변수에 저장



SCA on RSA: Non-Invasive Attack

❖ 키 비트 확인 단계

- 1996 [Crypto] Timing Attacks
- 1999 [Crypto] Simple Power Analysis
- [CHES] Differential Power Analysis
- 2001 [CHES] Big Mac Attack
- 2010 [J. ETRI] Second-Order CPA
- 2017 [ISPEC] Key Bit-dependent Attack
- 2018 [USENIX] KBA on Mobile phone



Algorithm. Left to Right Binary Method

Input $M, N, k = (k_{n-1}, k_{n-2}, \dots, k_0)_2$

Output $M^k \bmod N$

Step 1. $R_0 = 1$

Step 2. For $i = n - 1$ down to 0 do

Countermeasure

Step3. Return R_0

키 비트 값이 추출되어 변수에 저장

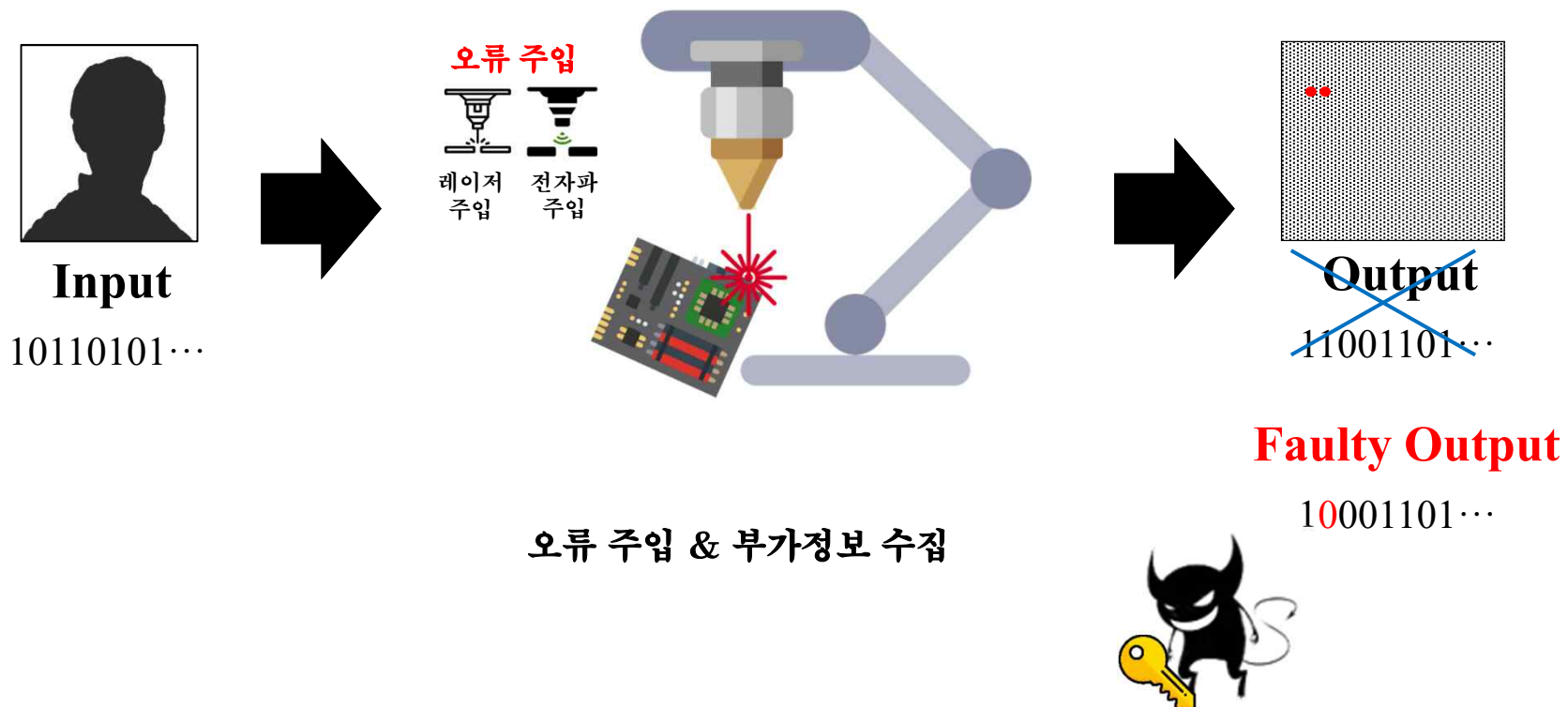


Semi-Invasive Attacks

준침입

- ❖ 디바이스에서 암호 알고리즘 동작 시 오류(레이저, 전자파 등)를 주입하여 변경된 출력 값과 부가적인 정보(수행 시간, 소비전력, 전자기 신호 등)를 수집·분석하여 비밀 정보를 획득하는 분석기법

➢ Ex) 오류 주입 공격



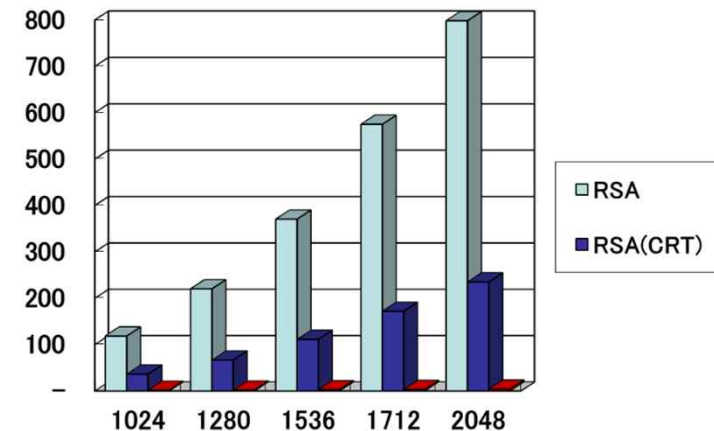
■ RSA-CRT

Decryption: $M \equiv C^d \pmod{n}$

❖ CRT based Decryption

- $n = p \times q$
- $M \equiv C_p \equiv C^d \pmod{(q-1)} \pmod{p}$
- $M \equiv C_q \equiv C^d \pmod{(q-1)} \pmod{q}$

✓ Gauss



$$M \equiv \left(C_p \cdot q \cdot (q^{-1} \pmod{p}) + C_q \cdot p \cdot (p^{-1} \pmod{q}) \right) \pmod{n}$$

✓ Garner

$$M \equiv C_q + \left((C_p - C_q) \cdot (q^{-1} \pmod{p}) \pmod{p} \right) \cdot q$$

Approx **4 times** faster than doing directly

SCA on RSA: Semi-Invasive Attack

❖ RSA-CRT 오류 주입 분석

1997 [EUROCRYPT]

Differential Fault Analysis

Algorithm. RSA-CRT using Garner's reconstruction

Input	Message M , $sk = (p, q, d_p, d_q, i_q)$ where $d_p = d \ (m \bmod (p-1))$, $d_q = d \ (m \bmod (q-1))$, $i_q = q^{-1} \ (m \bmod p)$
Output	Signature of M : $S = M^d \ (m \bmod N)$
Step 1. $S_p = M^{d_p} \ (m \bmod p)$, $S_q = M^{d_q} \ (m \bmod q)$ $\quad \quad \quad \underline{\hspace{1.5cm}}$	
Step 2. $S = CRT(S_p, S_q) = S_q + q \left(i_q (S_p - S_q) \ (m \bmod p) \right)$	
Step3. Return S	

SCA on RSA: Semi-Invasive Attack

❖ RSA-CRT 오류 주입 분석

1997 [EUROCRYPT]

Differential Fault Analysis

정확한 서명 (S) 과 오류서명 (\tilde{S}) 이용

Algorithm. RSA-CRT using Garner's reconstruction

Input	Message M , $sk = (p, q, d_p, d_q, i_q)$ where $d_p = d \ (m \bmod (p-1))$, $d_q = d \ (m \bmod (q-1))$, $i_q = q^{-1} \ (m \bmod p)$
Output	Signature of M : $S = M^d \ (m \bmod N)$
Step 1. $\tilde{S}_p = M^{d_p} \ (m \bmod p)$, $S_q = M^{d_q} \ (m \bmod q)$ Step 2. $\tilde{S} = CRT(\tilde{S}_p, S_q) = S_q + q \left(i_q (\tilde{S}_p - S_q) \ (m \bmod p) \right)$ Step 3. Return \tilde{S}	

$$q = \gcd \left((S - \tilde{S}) \ (m \bmod N), N \right)$$

SCA on RSA: Semi-Invasive Attack

❖ RSA-CRT 오류 주입 분석

1997 [EUROCRYPT]

Differential Fault Analysis

1999 [J. Cryptol]

Differential Fault Analysis

메시지 (M) 과 오류서명 (\tilde{S}) 이용

Countermeasure

→ 서명 출력 전 검증하는 연산 추가
추가 지수승 연산 필요

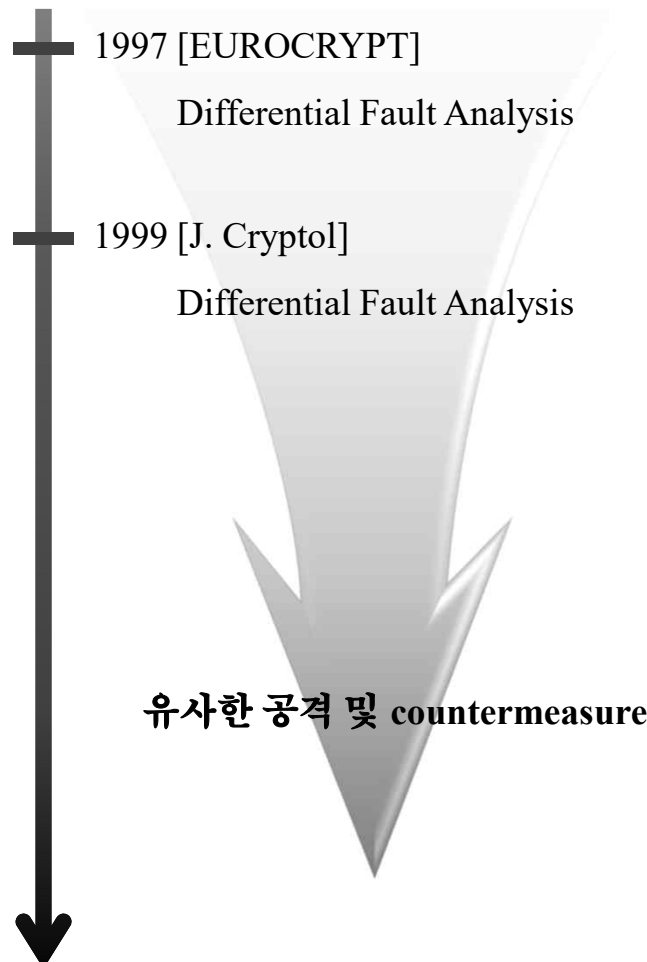
Algorithm. RSA-CRT using Garner's reconstruction

Input	Message M , $sk = (p, q, d_p, d_q, i_q)$ where $d_p = d \pmod{p-1}$, $d_q = d \pmod{q-1}$, $i_q = q^{-1} \pmod{p}$
Output	Signature of M : $S = M^d \pmod{N}$
Step 1. $\tilde{S}_p = M^{d_p} \pmod{p}$, $S_q = M^{d_q} \pmod{q}$ ----- Step 2. $\tilde{S} = CRT(\tilde{S}_p, S_q) = S_q + q \left(i_q (\tilde{S}_p - S_q) \pmod{p} \right)$ Step 3. Return \tilde{S}	

$$q = \text{gcd} \left((\tilde{S}^e - M) \pmod{N}, N \right)$$

SCA on RSA: Semi-Invasive Attack

❖ RSA-CRT 오류 주입 분석



Algorithm. RSA-CRT using Garner's reconstruction

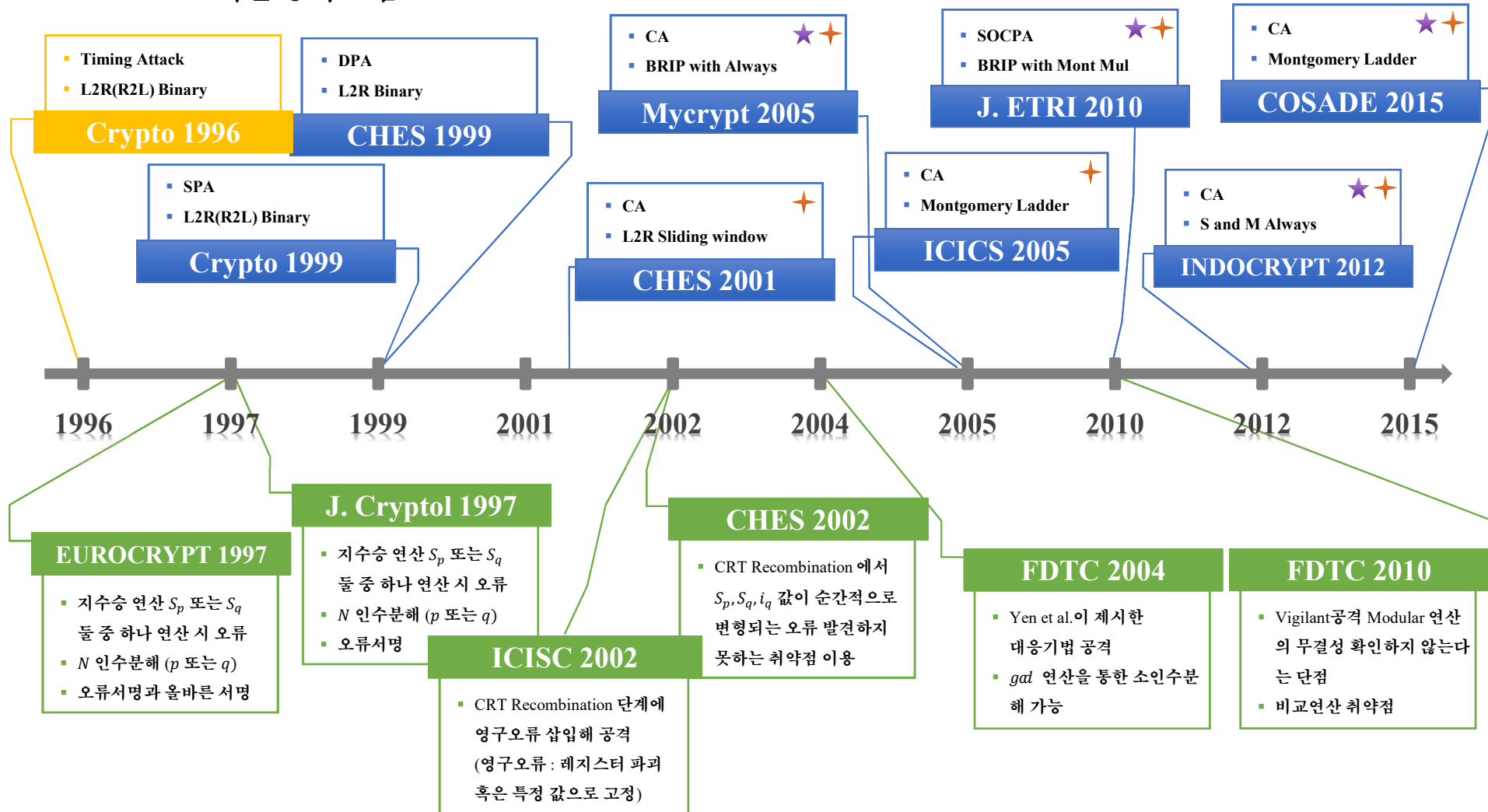
Input	Message M , $sk = (p, q, d_p, d_q, i_q)$ where $d_p = d \pmod{p-1}$, $d_q = d \pmod{q-1}$, $i_q = q^{-1} \pmod{p}$
Output	Signature of M : $S = M^d \pmod{N}$
Step 1. $\tilde{S}_p = M^{d_p} \pmod{p}$, $S_q = M^{d_q} \pmod{q}$ Step 2. $\tilde{S} = CRT(\tilde{S}_p, S_q) = S_q + q \left(i_q (\tilde{S}_p - S_q) \pmod{p} \right)$ Step 3. Return \tilde{S}	

$$q = \text{gcd} \left((\tilde{S}^e - M) \pmod{N}, N \right)$$

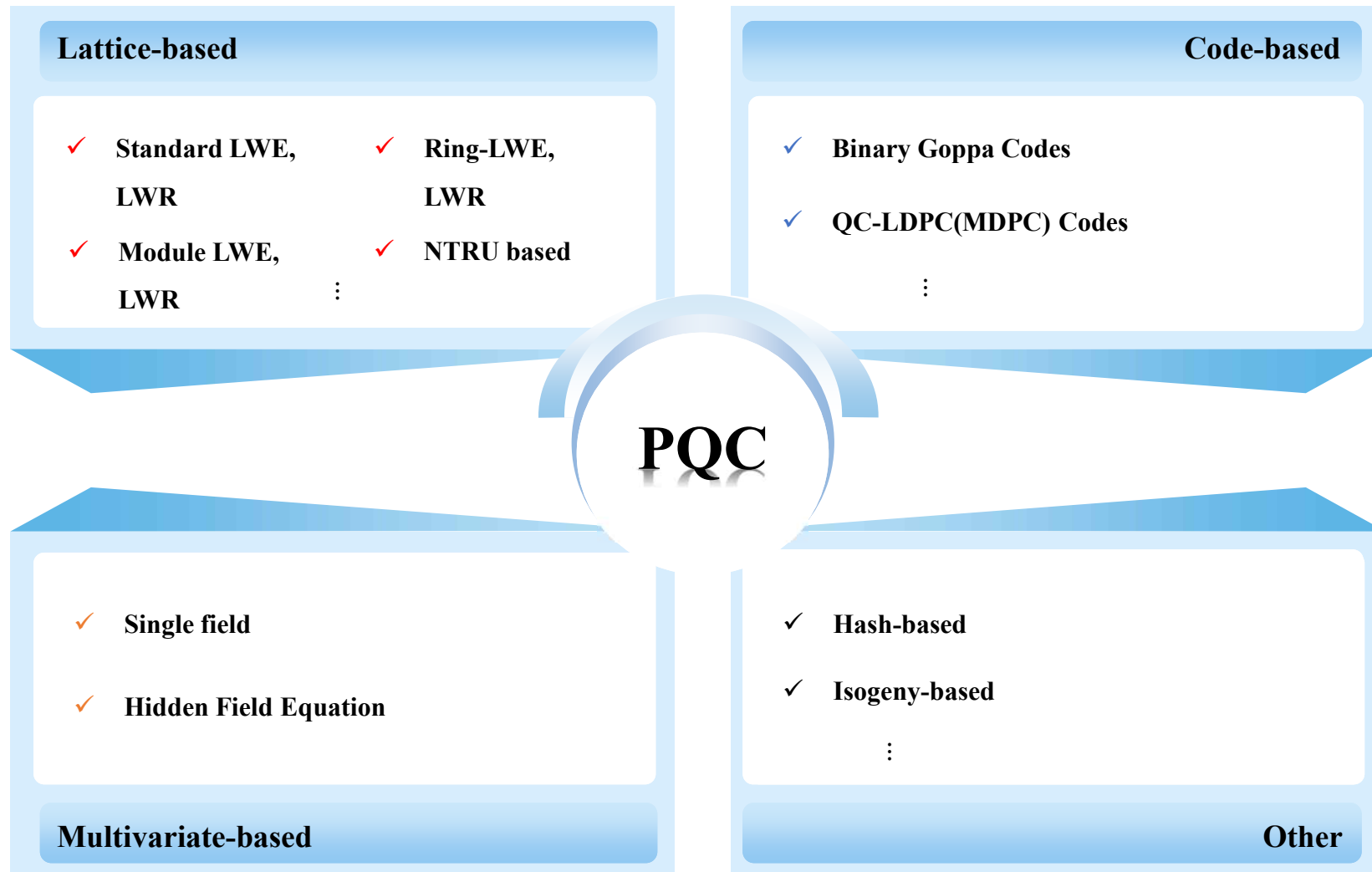
Side Channel Analysis on PKC

★ SPA CM
 ☆ DPA CM

❖ RSA 기준 공격 흐름

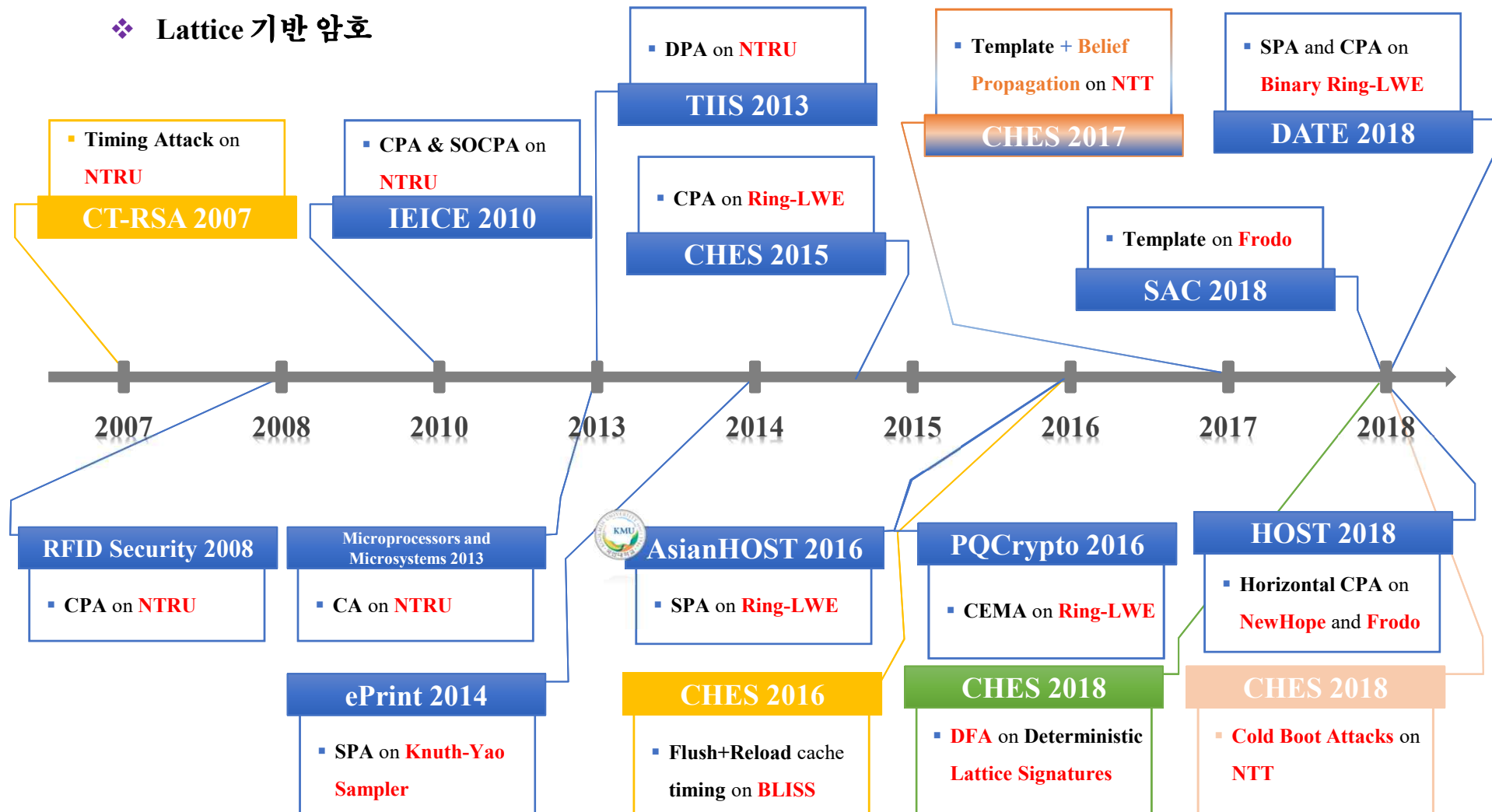


■ Hard problem on Post-Quantum Cryptography



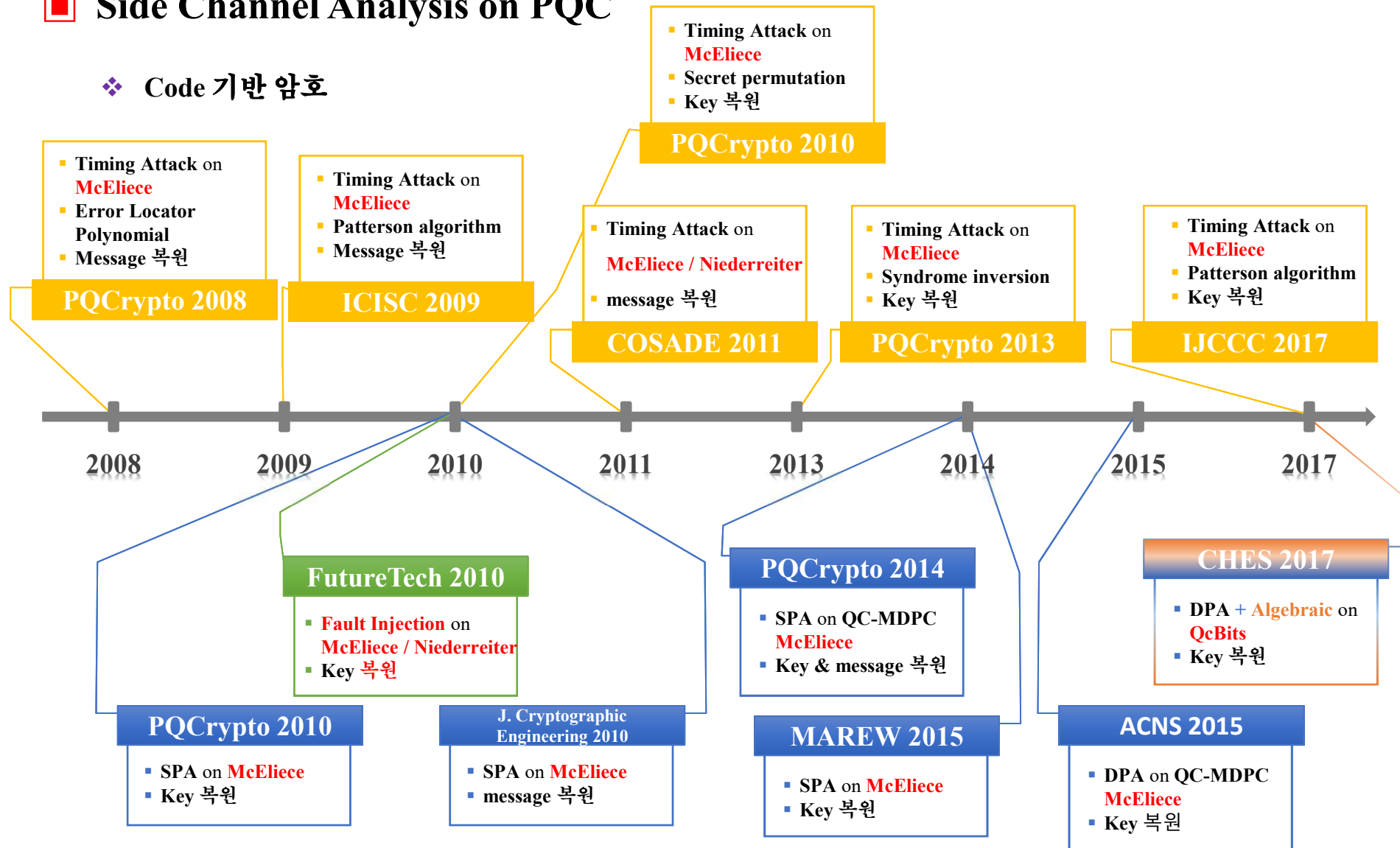
Side Channel Analysis on PQC

❖ Lattice 기반 암호



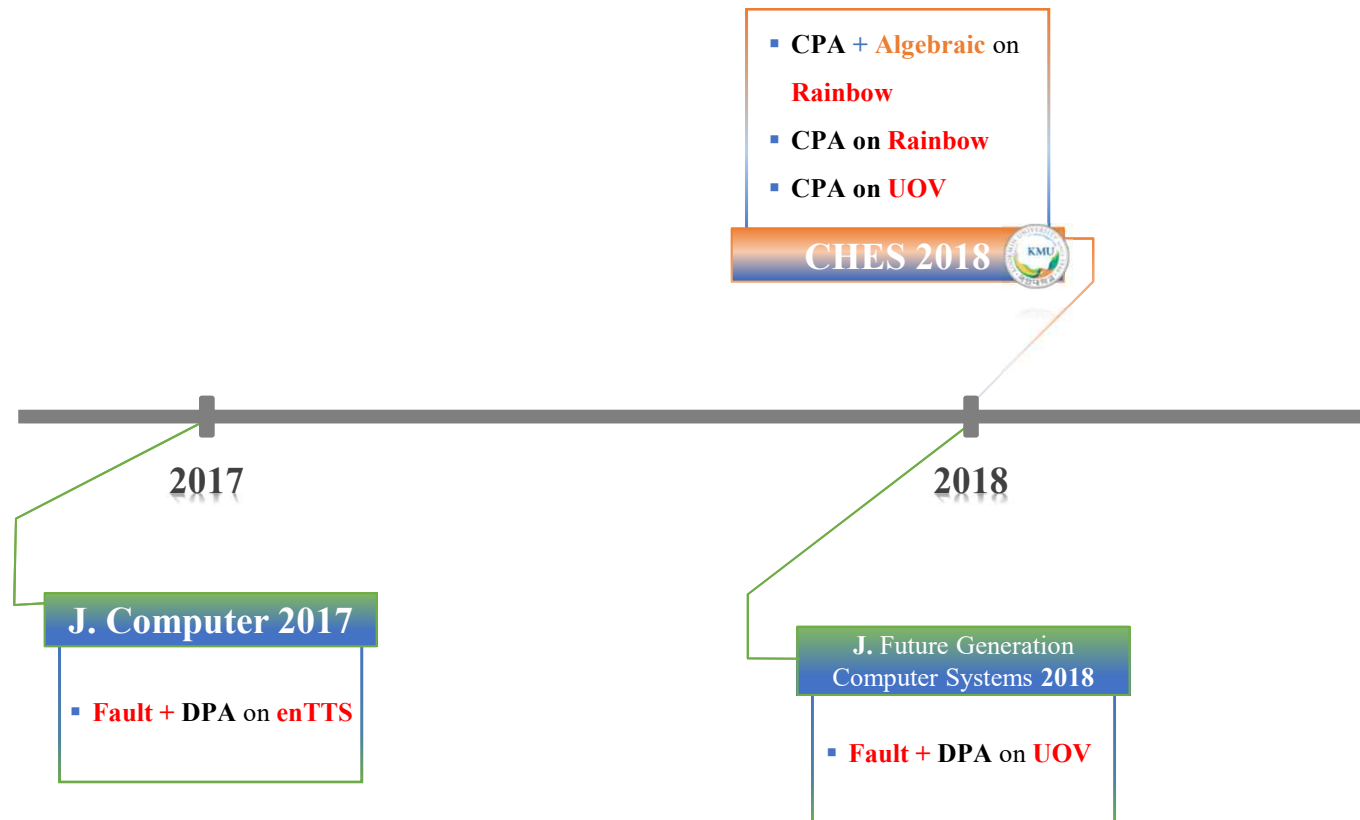
Side Channel Analysis on PQC

❖ Code 기반 암호



■ Side Channel Analysis on PQC

❖ Multivariate 기반 암호



SCA on PQC - 곱 연산 부채널 분석 취약점

q	n	\bar{n}
2^{15}	640	8

128-bit security

❖ 행렬 곱 (1): 행렬 \times 행렬

Lattice: Standard LWE

FrodoKEM

1. Generate

$$seed \leftarrow_R \{0, 1\}^{128}$$

$$A \in \mathbb{Z}_q^{n \times n} \leftarrow \text{Gen}(seed)$$

$$S, E \leftarrow_R \chi(\mathbb{Z}_q^{n \times \bar{n}})$$

2. Compute

$$\star B \leftarrow AS + E$$

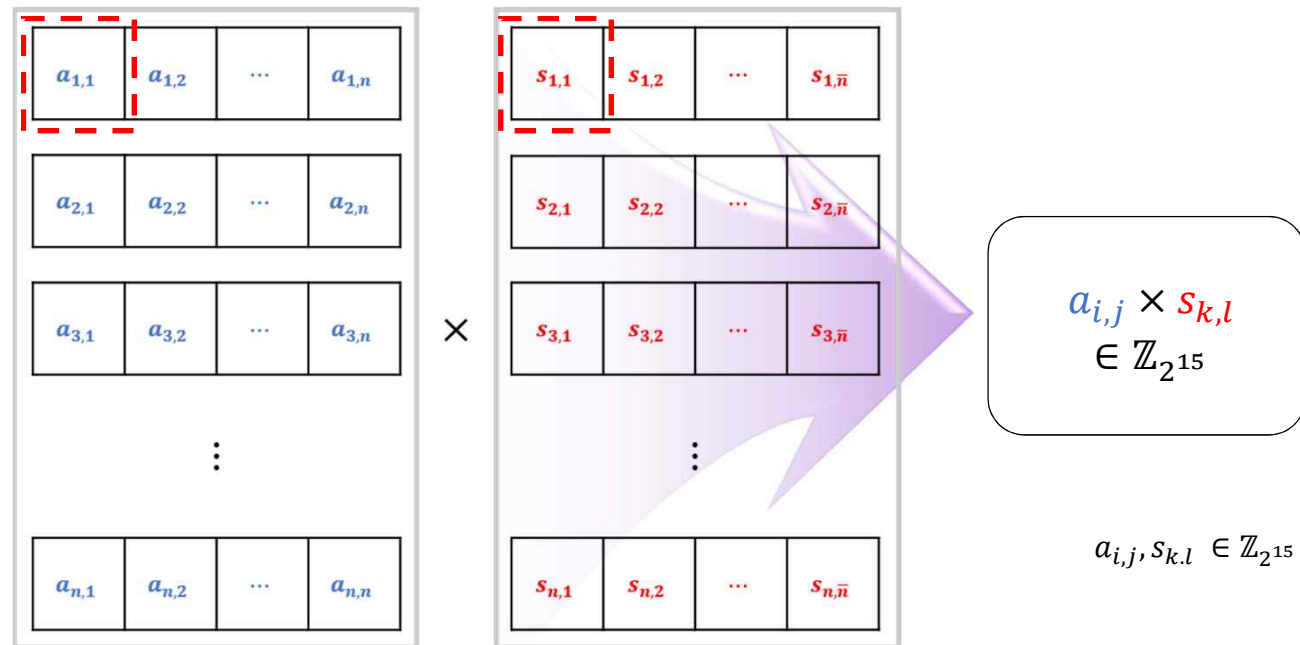
3. Send $(B, seed)$ to Bob

4. Receive (B', C) from Bob

5. Compute

$$K \leftarrow \text{Rec}(B', S, C)$$

Output : K



SCA on PQC - 곱 연산 부채널 분석 취약점

q	m
2^8	48

128-bit security

❖ 행렬 곱 (2): 행렬 \times 벡터

Multivariate: small field

Rainbow Sig.

message $X \in \mathbb{F}_q^m$.Secret key: $S \in \mathbb{F}_q^{m \times m}$, central map \mathcal{F} , $T \in \mathbb{F}_q^{n \times n}$

1. Compute

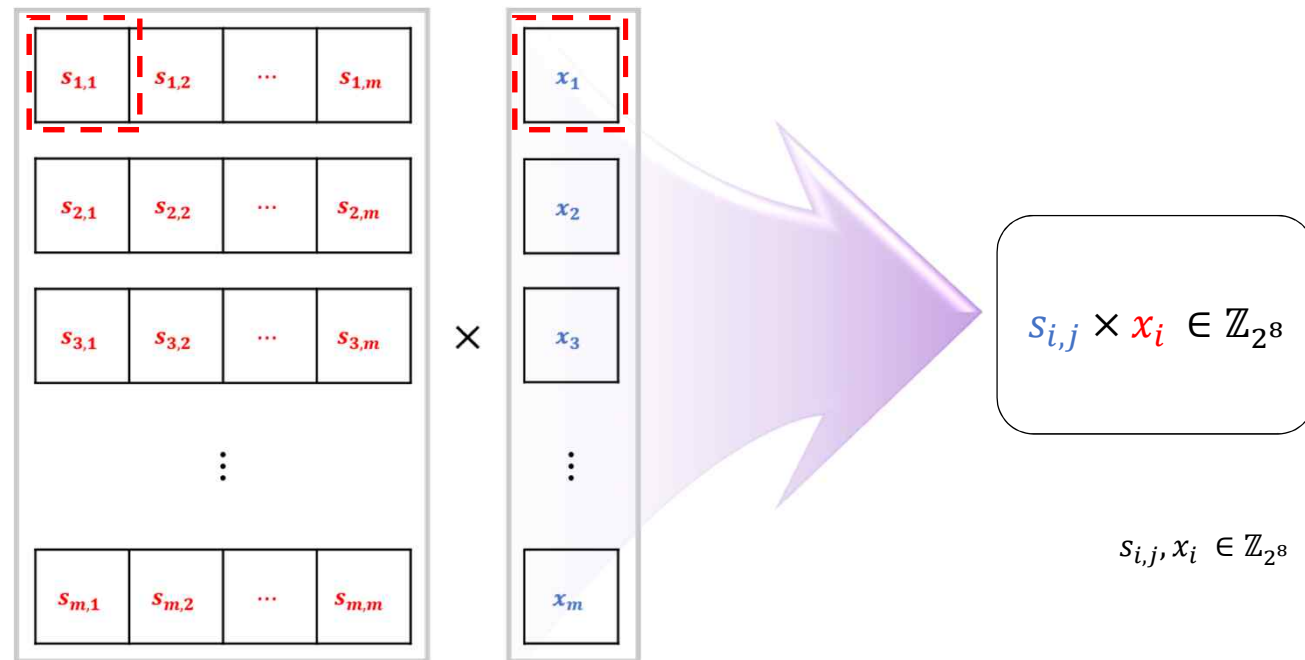
$$\star \quad X' \leftarrow S^{-1}X$$

2. Compute the inversion of the central map \mathcal{F}

$$X' \leftarrow \mathcal{F}^{-1}(X')$$

3. Compute

$$Y \leftarrow T^{-1}X''$$

Output : Y 

SCA on PQC - 곱 연산 부채널 분석 취약점

 m

184

128-bit security

❖ 행렬 곱 (3): 행렬 \times 벡터 (Binary)

Multivariate: Hidden Field Equation

Gui Sig.

message $X \in \mathbb{F}_2^m$.Secret key: $S \in \mathbb{F}_2^{m \times m}$, central map \mathcal{F} , $T \in \mathbb{F}_2^{(n+v) \times (n+v)}$

1. Compute

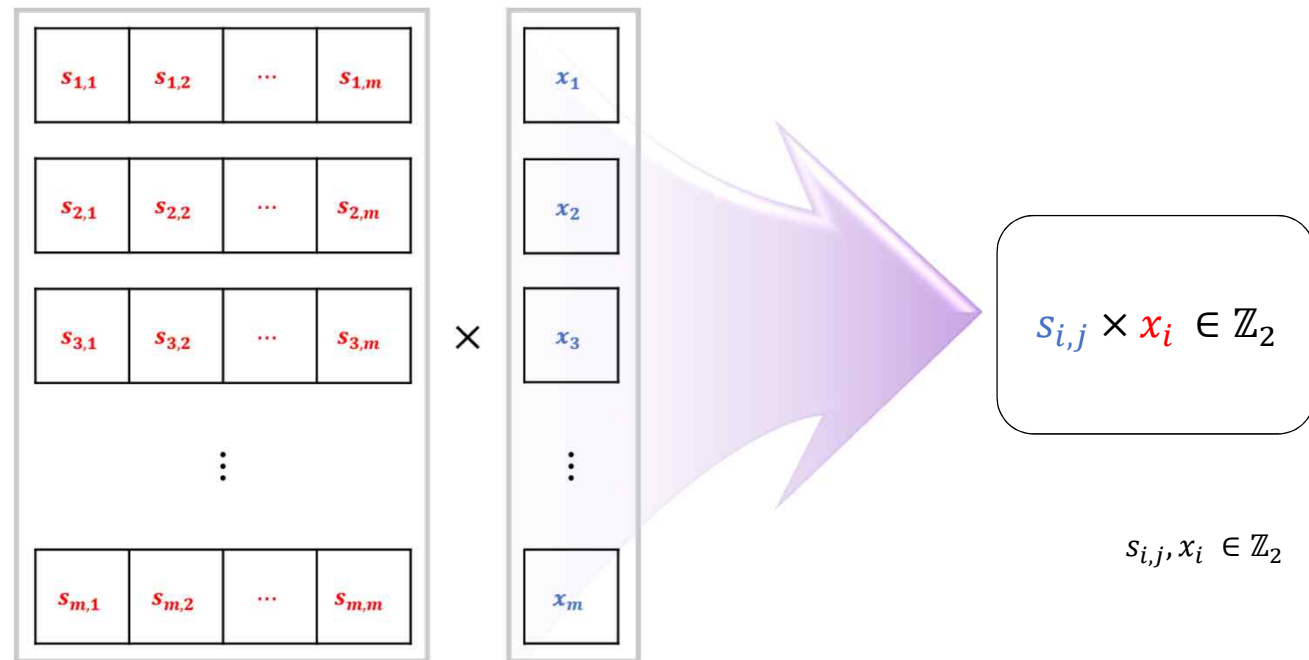
$$\star \quad X' \leftarrow S^{-1}X$$

2. Compute the inversion of the central map \mathcal{F}

$$X' \leftarrow \mathcal{F}^{-1}(X')$$

3. Compute

$$Y \leftarrow T^{-1}X''$$

Output : Y 

$$s_{i,j}, x_i \in \mathbb{Z}_2$$

SCA on PQC - 곱 연산 부채널 분석 취약점

m

184

128-bit security

❖ 행렬 곱 (3): 행렬 \times 벡터 (Binary)

Multivariate: Hidden Field Equation

Gui Sig.

message $X \in \mathbb{F}_2^m$.

Secret key: $S \in \mathbb{F}_2^{m \times m}$, central map \mathcal{F} ,

$T \in \mathbb{F}_2^{(n+v) \times (n+v)}$

1. Compute

$$\star X' \leftarrow S^{-1}X$$

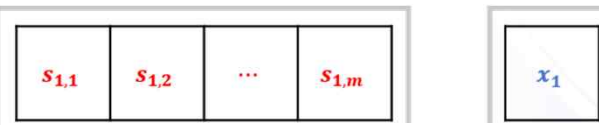
2. Compute the inversion of the central map \mathcal{F}

$$X' \leftarrow \mathcal{F}^{-1}(X')$$

3. Compute

$$Y \leftarrow T^{-1}X''$$

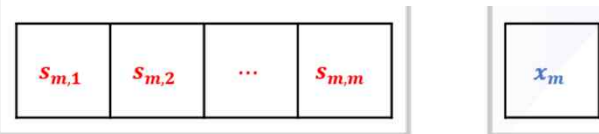
Output : Y



✓ Binary 곱의 경우 일반적인 부채널 분석 적용 어려움

✓ 구현 방법에 따라 부채널 분석 적용 가능

➔ Ex) [CHES 2017] QcBits(QC-MDPC 기반 McEliece)의 DPA 공격 존재



$$s_{i,j}, x_i \in \mathbb{Z}_2$$

SCA on PQC - 곱 연산 부채널 분석 취약점

❖ 다항식 곱 (1)

Lattice: NTRU lattice

NTRU Dec.

Ciphertext $c \in R_q$

Secret key $f = 1 + pfm \bmod q \in R_q$

F : d_F 개의 1을 갖는 $N-1$ 차 이진 다항식

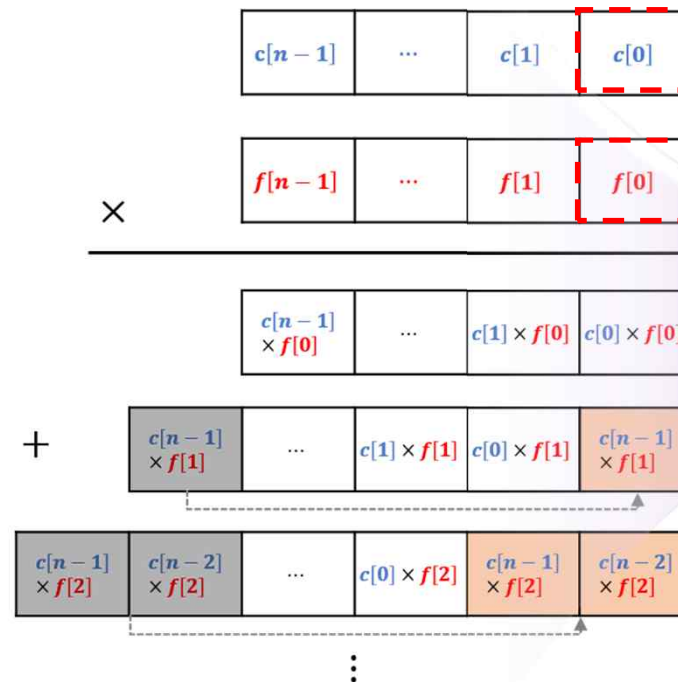
1. Compute

$$\star a \leftarrow c * f \bmod q$$

2. Compute

$$m = a \bmod p$$

Output : m



$$c[i] \times f[j] \in \mathbb{Z}_{2^{11}}$$

$$c[i] \in \mathbb{Z}_{2^{11}} \\ f[j] \in \{-1, 0, 1\}$$

q	n	p
2^{11}	443	3

128-bit security

$$R_q = \mathbb{Z}_q[X] / \langle x^n - 1 \rangle$$

SCA on PQC - 곱 연산 부채널 분석 취약점

❖ 다항식 곱 (2)

Lattice: Ring-LWE with NTT

RLWE Dec.

Ciphertext $(\tilde{c}_1, \tilde{c}_2) \in R_q \times R_q$

Secret key $\tilde{r}_2 \in R_q$

1. Compute

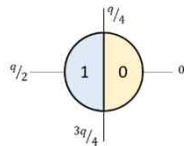
$$\star X \leftarrow \tilde{c}_1 \cdot \tilde{r}_2 + \tilde{c}_2$$

2. Inverse NTT operation

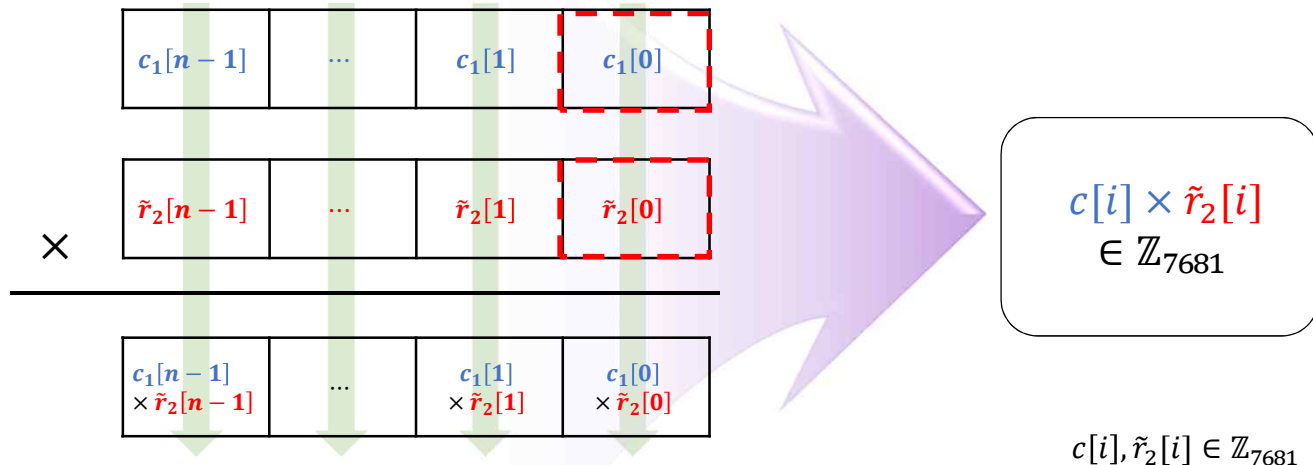
$$X = \text{NTT}(X)$$

3. Decode the message

$$m = \mathbf{h}(X)$$



Output : m

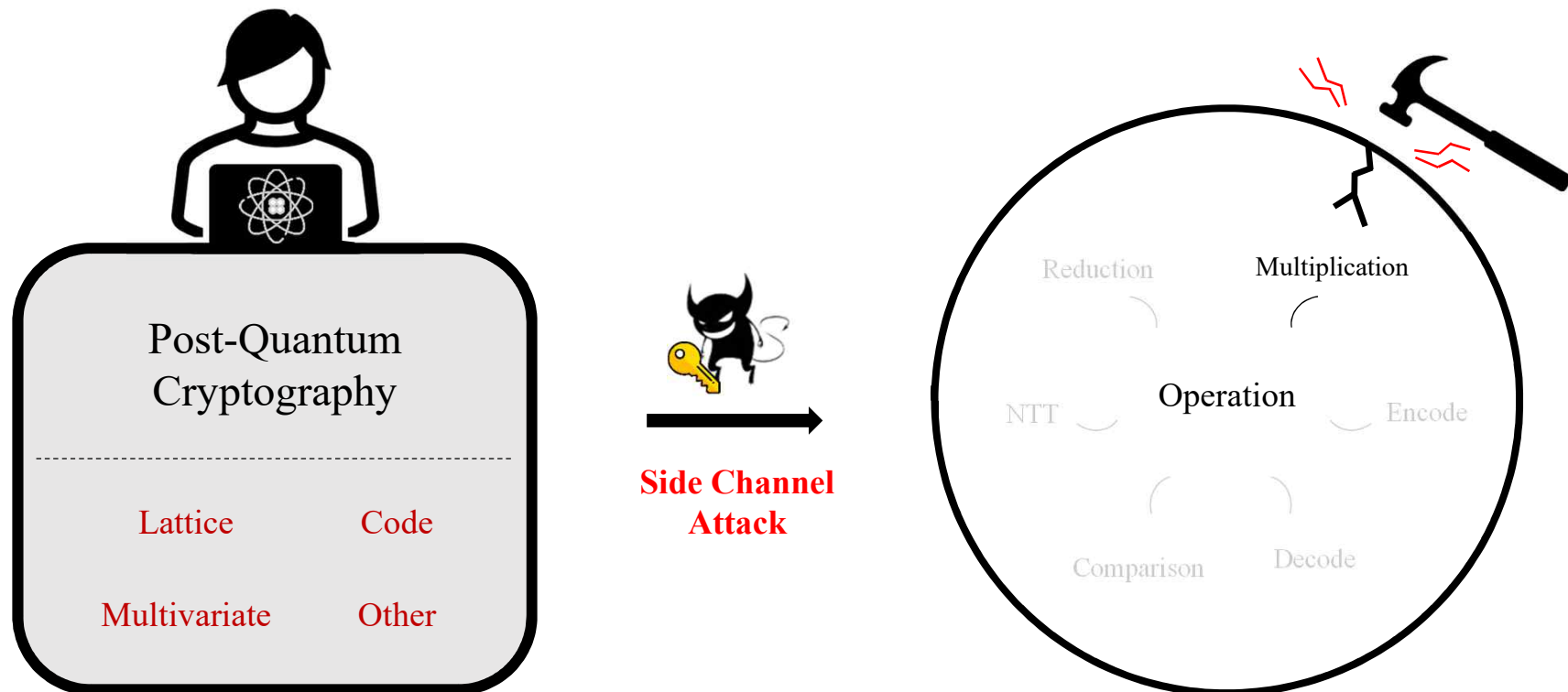


n	q
256	7681

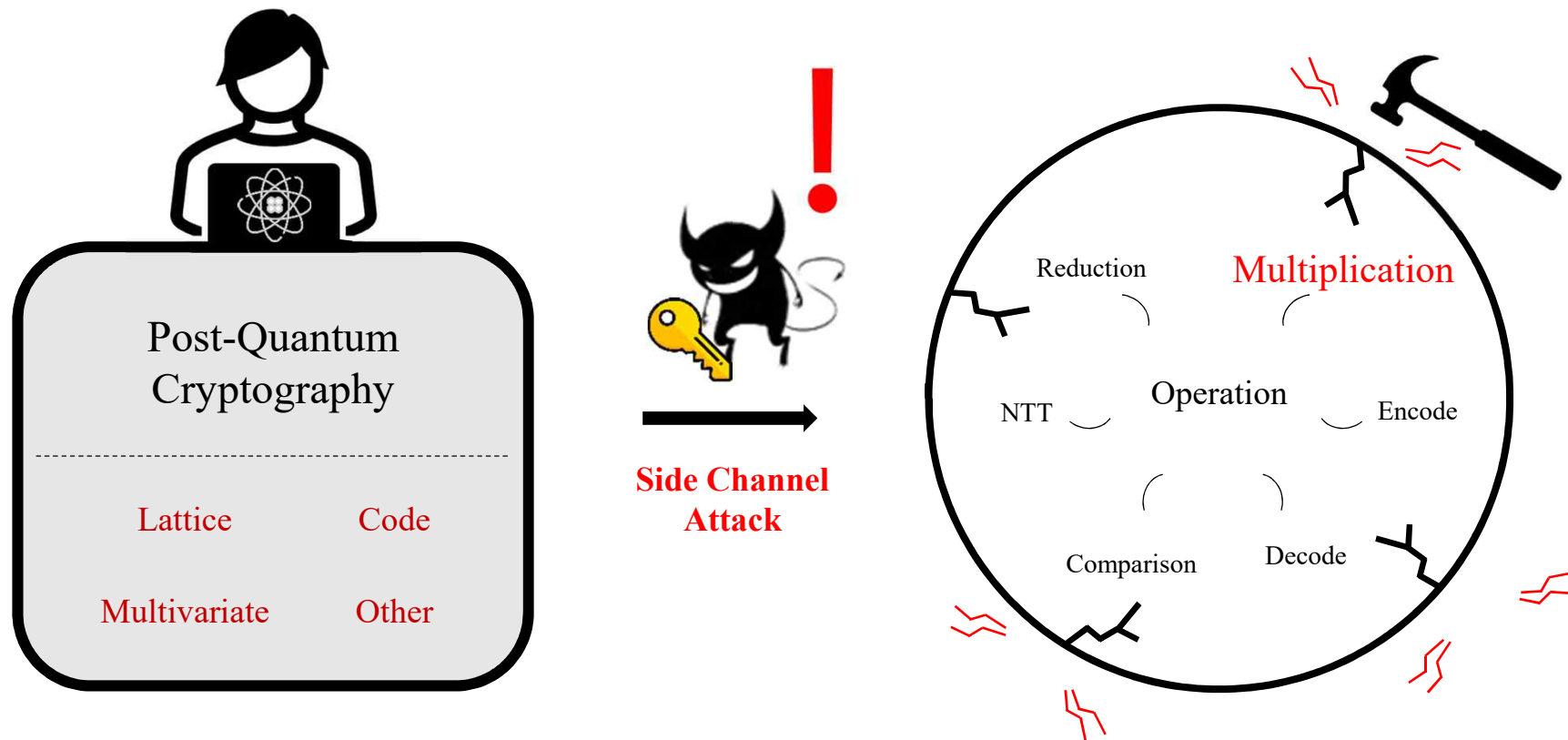
128-bit security

$$R_q = \mathbb{Z}_q[X]/\langle X^N + 1 \rangle$$

■ SCA on PQC



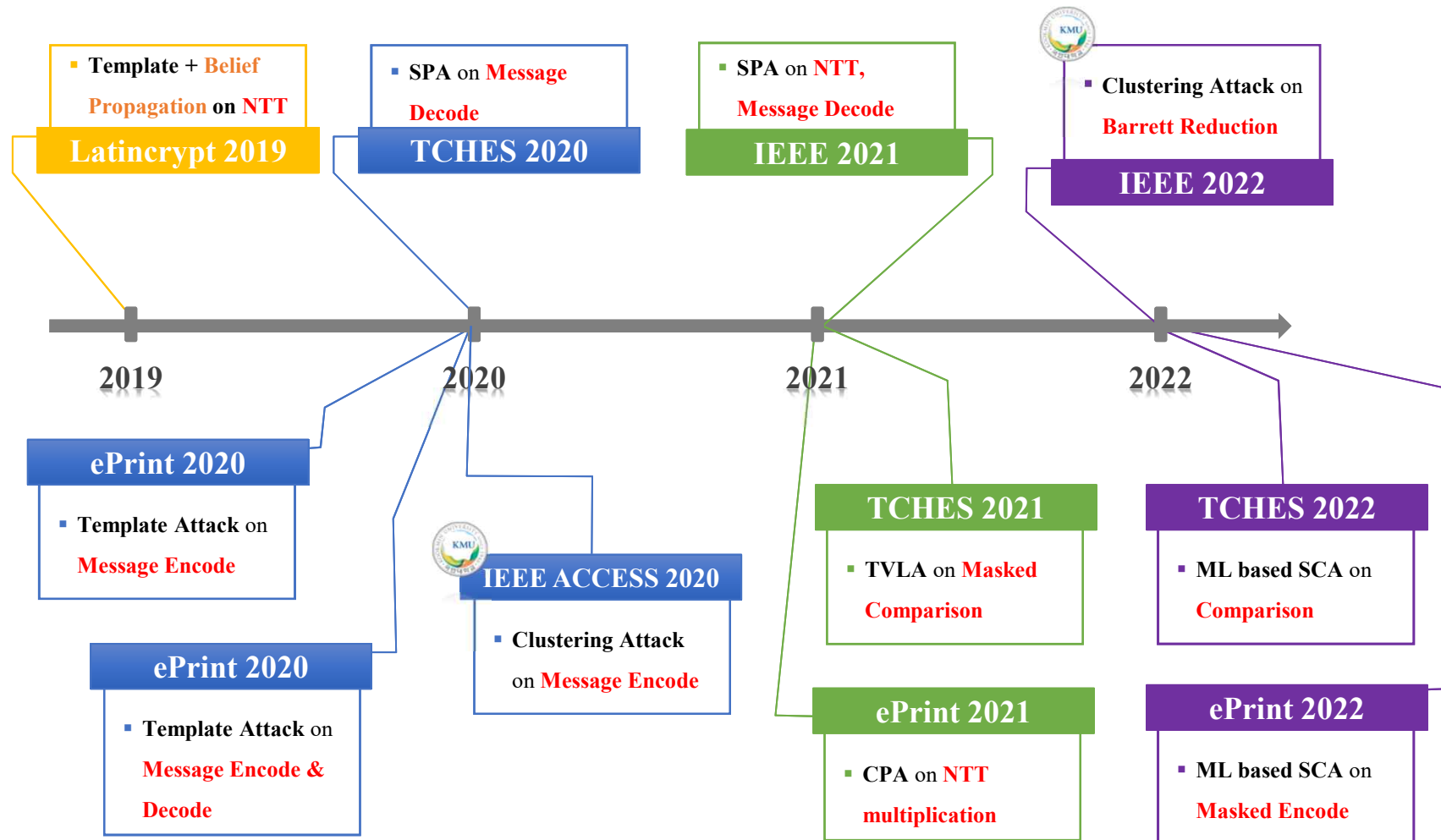
■ SCA on PQC



SCA on PQC

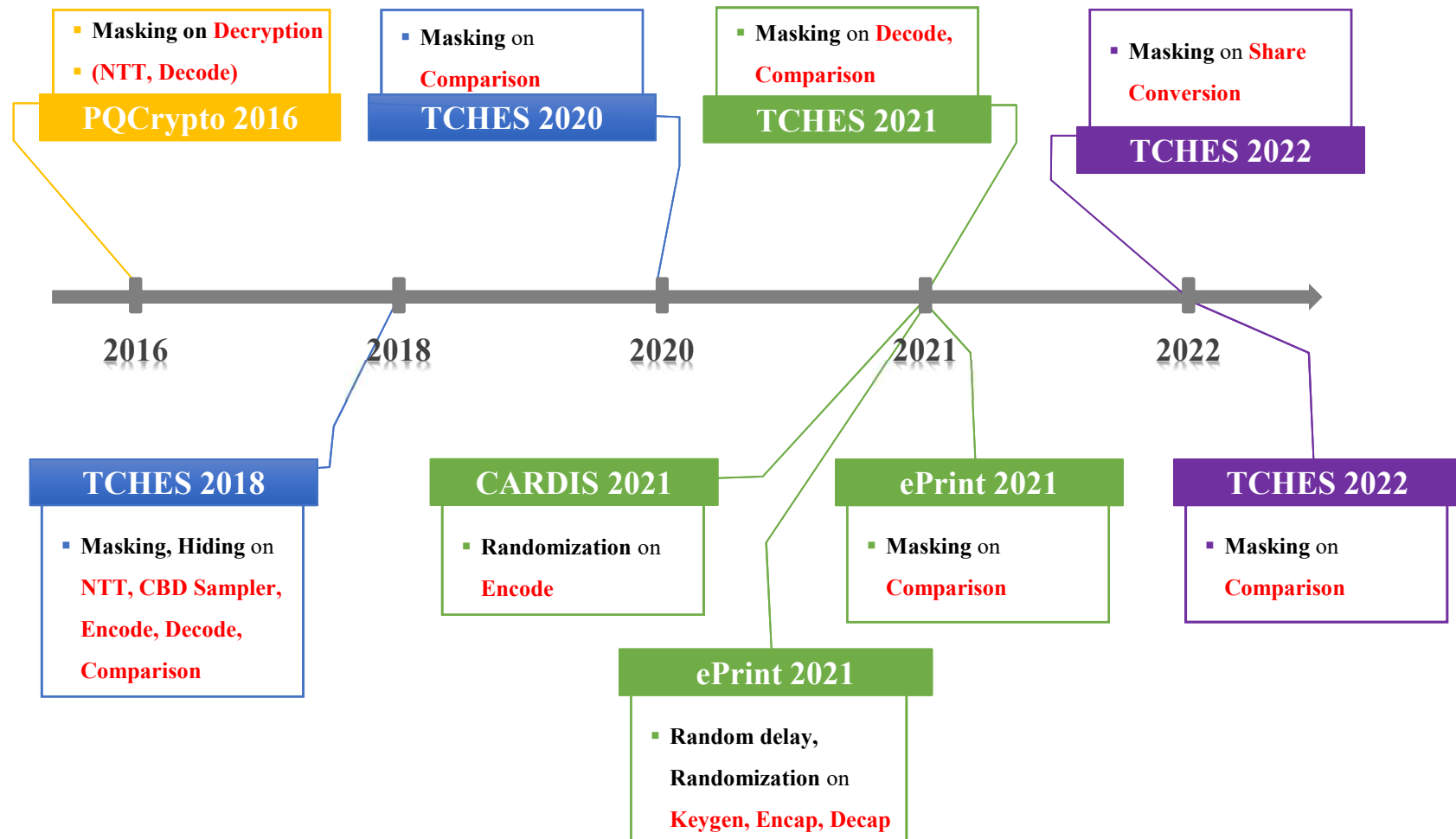
CRYSTALS-KYBER

취약성 연산 위치	NTT	Multiplication	Encode	Decode	Comparison	Reduction
동향 개수	2	1	4	3	1	1



SCA Countermeasure on PQC

❖ CRYSTALS-KYBER



- 1 부채널 분석 관점에서의 PKC & PQC
- ✓ 2 PQC 알고리즘에 대한 부채널 분석 동향
- 3 상수시간 알고리즘은 항상 안전한가?
- 4 최적화 구현과 부채널 분석 취약점의 상관관계

NIST PQC 공모전

Round 2	Signatures	PKE/KEM	Overall
Lattice	3	9	12
Code	-	7	7
Multi-variate	4	-	4
Symmetric/Hash	2	-	2
Other	-	1	1
Total	9	17	26

[2019] 2 라운드 시작

Standardization (Candidate)

Round 4	Signatures	PKE/KEM	Overall
Lattice	2 (-)	1 (-)	3 (-)
Code	- (-)	- (3)	- (3)
Multi-variate	- (-)	- (-)	- (-)
Symmetric/Hash	1 (-)	- (-)	1 (-)
Isogeny	- (-)	- (1)	- (1)
Total	3	1 (4)	4 (4)



[2022] 4 라운드 시작

2016

현재

[2017] 1 라운드 시작

[2020] 3 라운드 시작 Finalists (Alternate)

Round 1	Signatures	PKE/KEM	Overall
Lattice	5	21	26
Code	2	17	19
Multi-variate	7	2	9
Symmetric/Hash	3	-	3
Other	2	5	7
Total	19	45	64

Round 3	Signatures	PKE/KEM	Overall
Lattice	2 (-)	3 (2)	5 (2)
Code	-	1 (2)	1 (2)
Multi-variate	1 (1)	- (-)	1 (1)
Symmetric/Hash	- (2)	- (-)	- (2)
Other	-	- (1)	- (1)
Total	3 (3)	4 (5)	7 (8)

■ 격자 기반 PQC 부채널 분석 동향 [1]

논문	공격 기법	대상 연산	구현 코드	실험 환경	공격 파형 수	지표	성공률
[RBR+20]	Template Attack	Decode	Round 2 Ref.	STM32F407VG MCU	256 (Single Bit) / 32 (Single Byte)	비밀 메시지 전체	100 %

❖ PKE Decryption의 메시지 Decode를 대상으로 템플릿 공격을 통해 비밀 메시지 분석

➤ 공격 대상 – NewHope, **KYBER**, SABER, Round5, LAC (Round 3 구현에도 적용가능)

KYBER.CPAPKE.Dec()	
Input	$sk = s, c = (c_1, c_2)$
Output	Message m'
<div>1: $u' \leftarrow \text{Decompress}_q(c_1, d_u)$</div> <div>2: $v' \leftarrow \text{Decompress}_q(c_2, d_v)$</div> <div>3: $M \leftarrow v' - u' \times s$</div> <div>4: $m' \leftarrow \text{Decode}(M)$</div> <div>5: return m'</div>	

```
void poly_tomsg(uint8_t msg[KYBER_INDCPA_MSGBYTES], poly *a)
{
    unsigned int i, j;
    uint16_t t;

    poly_csubq(a);

    for(i=0; i<KYBER_N/8; i++) {
        msg[i] = 0;
        for(j=0; j<8; j++) {
            t = (((uint16_t)a->coeffs[8*i+j] << 1) + KYBER_Q/2)/KYBER_Q & 1;
            msg[i] |= t << j;
        }
    }
}
```

✓ KYBER Round 3 Reference c code Decode 함수

■ 분석 방법 [1]

❖ Decode 함수의 구현 취약성

```

void poly_tomsg(uint8_t msg[KYBER_INDCPA_MSGBYTES], poly *a)
{
    :
    for(i=0; i<KYBER_N/8; i++) {
        msg[i] = 0;
        for(j=0; j<8; j++) {
            t = (((uint16_t)a->coeffs[8*i+j] << 1) + KYBER_Q/2)/KYBER_Q & 1;
            msg[i] |= t << j;
        }
    }
}
  
```

➤ Single Bit Leakage Attack) 메시지 **msg**가 업데이트 될 때 HW 차이 발생

✓ Ex)

msg[0] = 0



<i>j</i>	0	...	7
msg[0]	0	0	0



Power ≈

HW(msg[0]) = 0

msg[0] = 1



<i>j</i>	0	...	7
msg[0]	1	1	1



Power ≈

HW(msg[0]) = 1

■ 분석 방법 [1]

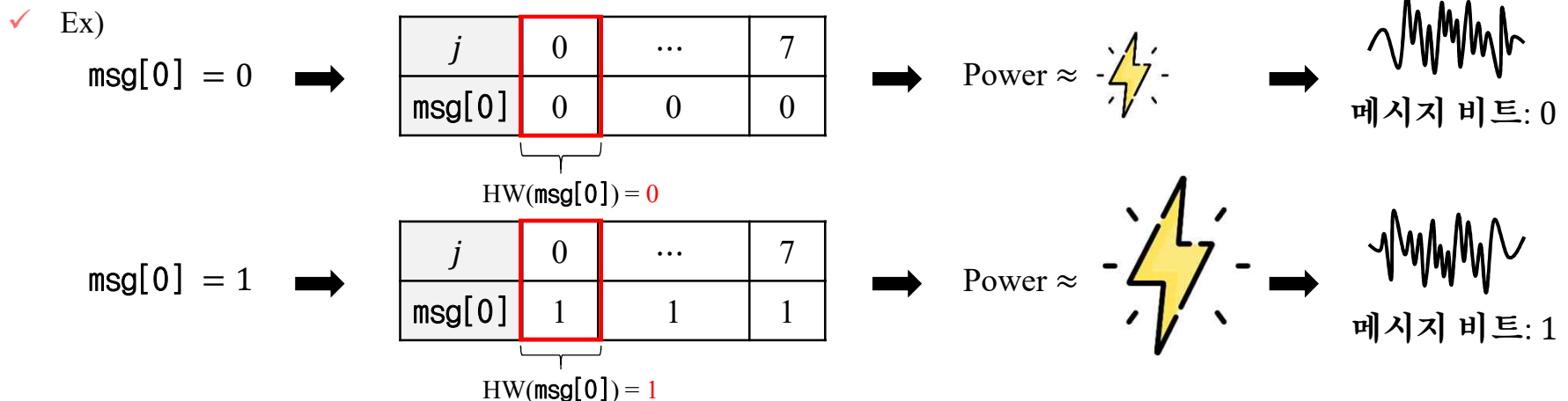
❖ Decode 함수의 구현 취약성

```

void poly_tomsg(uint8_t msg[KYBER_INDCPA_MSGBYTES], poly *a)
{
    :
    for(i=0; i<KYBER_N/8; i++) {
        msg[i] = 0;
        for(j=0; j<8; j++) {
            t = (((uint16_t)a->coeffs[8*i+j] << 1) + KYBER_Q/2)/KYBER_Q & 1;
            msg[i] |= t << j;
        }
    }
}
  
```

➤ Single Bit Leakage Attack) 메시지 **msg**가 업데이트 될 때 HW 차이 발생

템플릿 생성



■ 분석 결과 [1]

❖ Single Bit Leakage Attack (공격 과정에서 한 비트 씩 복구)



- 템플릿 생성 과정: 2*50개 파형 사용
- 공격 과정: 256개 파형 사용 (m : 256비트)
- 성공률: 98.5~100%

❖ Single Byte Leakage Attack (공격 과정에서 한 바이트 씩 복구)

- 템플릿 생성 과정: 256*50개 파형 사용
- 공격 과정: 32개 파형 사용 (m : 32바이트)
- 성공률: 35~100%

❖ 다른 격자 기반 PKE/KEM에도 동일한 취약성이 발생하여 적용 가능

```
void POLmsg2BS(uint8_t bytes[SABER_KEYBYTES], const uint16_t data[SABER_N])
{
    :
    for (j = 0; j < SABER_KEYBYTES; j++)
    {
        for (i = 0; i < 8; i++)
        {
            bytes[j] = bytes[j] | ((data[j * 8 + i] & 0x01) << i);
        }
    }
}
```

✓ SABER Round 3 Reference c code Decode 함수



■ 격자 기반 PQC 부채널 분석 동향 [2]

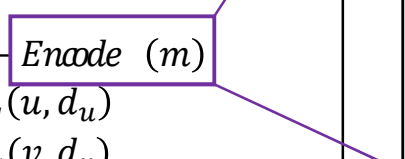
논문	공격 기법	대상 연산	구현 코드	실험 환경	공격 파형 수	지표	성공률
[SKL+20]	Clustering	Encode	Round 3 Ref.	STM32F3 MCU	1	비밀 메시지 전체	100 %

❖ KEM Encapsulation의 메시지 Encode를 공격하여 비밀 메시지 복구

➤ **KYBER**, SABER, FrodoKEM, NTRU, NTRU LPrime, Streamlined NTRU Prime

❖ 각 KEM의 메시지 Encode 타입에 따라 클러스터링 및 머신러닝 기반 분석 적용

KYBER.CPAPKE.Enc()	
Input	$pk = (A, b), m; r$
Output	c
1: $(t, e_1, e_2) \leftarrow \$B_{\eta_1}^k \times B_{\eta_2}^k \times B_{\eta_2}$ 2: $u \leftarrow t \times A + e_1$ 3: $v \leftarrow t \times b + e_2 + \text{Encode}(m)$ 4: $c_1 \leftarrow \text{Compress}_q(u, d_u)$ 5: $c_2 \leftarrow \text{Compress}_q(v, d_v)$ 6: return $c = (c_1, c_2)$	



```

void poly_frommsg(poly *r, const uint8_t msg[KYBER_INDCPA_MSGBYTES])
{
    unsigned int i, j;
    int16_t mask;

    for(i=0; i<KYBER_N/8; i++) {
        for(j=0; j<8; j++) {
            mask = -(int16_t)((msg[i] >> j)&1);
            r->coeffs[8*i+j] = mask & ((KYBER_Q+1)/2);
        }
    }
}
          
```

✓ KYBER Round 3 Reference c code Encode 함수

■ 분석 방법 [2]

❖ Encode 함수의 구현 취약성

➤ 메시지 비트에 따른 **mask** 값에 의해 부채널 분석 취약성 발생

```
void poly_frommsg(poly *r, const uint8_t msg[KYBER_INDCPA_MSGBYTES])
{
    unsigned int i, j;
    int16_t mask;

    for(i=0; i<KYBER_N/8; i++) {
        for(j=0; j<8; j++) {
            mask = -(int16_t)((msg[i] >> j)&1);
            r->coeffs[8*i+j] = mask & ((KYBER_Q+1)/2);
        }
    }
}
```

$$\frac{((msg[i] \gg j) \& 1) = 0}{\text{메시지 비트}} \quad \Rightarrow \quad \text{mask} = 0x0000 \quad \Rightarrow \quad \text{Power} \approx \text{⚡}$$

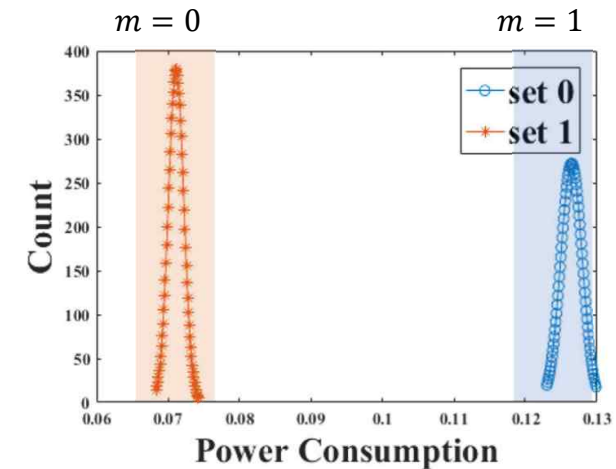
$$\frac{((msg[i] \gg j) \& 1) = 1}{\text{메시지 비트}} \quad \Rightarrow \quad \text{mask} = 0xffff \quad \Rightarrow \quad \text{Power} \approx \text{⚡}$$

■ 분석 결과 [2]

❖ KYBER Encode 함수에 대한 단일 파형 분석

- $m = 0, m = 1$ 일 때 Encode 함수에 대한 소비 전력 파형 수집
- **mask**값을 계산할 때 전력 차이가 발생

✓ 단일 파형으로 100%로 비밀 메시지 값 복구



❖ SABER

- 머신러닝 기반 프로파일링 공격을 적용한 메시지 복구 실험 결과

Optimization	First PoIs				Second PoIs
Level	1-bit value	2-bit value	8-bit value	Combined	1-bit value
-O0	100%	-	-	-	100%
-O1	99.97%	99.89%	99.83%	99.97%	100%
-O2	99.90%	99.56%	99.53%	99.87%	100%
-O3	95.87%	83.12%	86.96%	96.71%	100%
-Os	99.99%	99.96%	99.96%	99.98%	100%



▣ 격자 기반 PQC 부채널 분석 동향 [3]

논문	공격 기법	대상 연산	구현 코드	실험 환경	공격 파형 수	지표	성공률
[HLK+21]	ML-based SCA	NTT	Round 3 Ref.	STM32F3 MCU	1	비밀키 전체	76~100 %

- ❖ Dilithium 서명 생성의 NTT 연산 및 키 생성의 NTT, 샘플링, 덧셈, 라운딩, 패킹을 대상으로 비밀키 분석
- ❖ 머신러닝 기반 프로파일링 부채널 분석 적용

Dilithium Signature generation	
Input	Message m , $sk = (\rho, K, \tau, s_1, s_2, t_0)$
Output	Signature $\sigma = (z, c, h)$
1: $A \in \mathcal{R}_q^{k \times l} \leftarrow \text{ExpandA}(\rho)$ 2: $\mu \in \{0,1\}^{384} \leftarrow \text{CRH}(\tau m)$ 3: $\kappa \leftarrow 0$ 4: $\rho' \in \{0,1\}^{384} \leftarrow \text{CRH}(K \mu)$ 5: $\hat{s}_1 \leftarrow \text{NTT}(s_1), \hat{s}_2 \leftarrow \text{NTT}(s_2), \hat{t}_0 \leftarrow \text{NTT}(t_0)$ 6: $y \in \widetilde{\mathcal{S}}_{Y_1}^l \leftarrow \text{ExpandMask}(\rho', \kappa++)$: :	

```

void ntt(int32_t a[N]) {
    unsigned int len, start, j, k;
    int32_t zeta, t;

    k = 0;
    for(len = 128; len > 0; len >>= 1) {
        for(start = 0; start < N; start = j + len) {
            zeta = zetas[++k];
            for(j = start; j < start + len; ++j) {
                t = montgomery_reduce((int64_t)zeta * a[j + len]);
                a[j + len] = a[j] - t;
                a[j] = a[j] + t;
            }
        }
    }
}

```










✓ Dilithium Round 3 Reference c code NTT 함수

■ 분석 방법 [3]

❖ NTT 연산의 부채널 분석 취약성

- 입력된 비밀키 계수 s 에 따라 t 의 값이 바뀜
- s 와 전력 파형을 입력으로 신경망 학습

```
void ntt(int32_t a[N]) {
    :
    for(len = 128; len > 0; len >>= 1) {
        for(start = 0; start < N; start = j + len) {
            zeta = zetas[++k];
            for(j = start; j < start + len; ++j) {
                t = montgomery_reduce(((int64_t)zeta * a[j + len]));
                a[j + len] = a[j] - t;
                a[j] = a[j] + t;
            }
        }
    }
}
```

s	t	Power(t) \approx
-4	0xFFDA47FA	
-3	0x0023A5FC	
-2	0xFFED23FD	
-1	0x003681FF	
0	0x00000000	
1	0xFFC97E01	
2	0x0012DC03	
3	0xFFDC5A04	
4	0x0025B806	



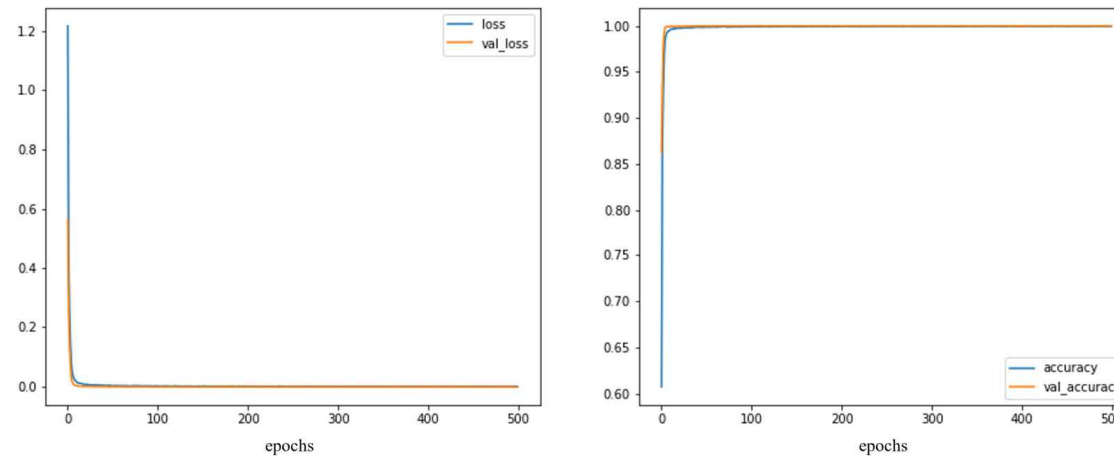
input (s , Power)



딥러닝 신경망

■ 분석 결과 [3]

❖ NTT 변환 분석 결과 (서명 생성 : s_1, s_2 , 키 생성 : s_1)



Optimization Level	Stage m : $s_{1,i,j}$, $1 \leq m < 8$, $0 \leq i < L$, $0 < j < N$	Stage 8: $s_{1,i,0}$, $0 \leq i < L$
-O3	100%	100%
-Os	100%	100%

❖ NTT 외 연산 분석 결과 (키 생성 : s_2)

Optimization Level	Sampling	Addition	Rounding	Packing
-O3	94.03	98.38	89.49	76.04
-Os	95.38	98.47	91.21	75.67

➤ 75~100% 확률로 비밀키 복구 가능



■ 격자 기반 PQC 부채널 분석 동향 [4]

논문	공격 기법	대상 연산	구현 코드	실험 환경	공격 파형 수	지표	성공률
[SPH22]	Clustering	Barrett Reduction	Round 3 Ref. / pqm4	STM32F3 MCU	6 ~ 12	비밀키 전체	100 %

❖ KYBER PKE Decryption의 Barrett Reduction을 대상으로 선택 암호문 공격으로 비밀키 복구

❖ 6~10개의 파형을 사용해 비밀키 복구 가능

KYBER.CPAPKE.Dec()	
Input	$sk = s, c = (c_1, c_2)$
Output	Message m'
1: $u' \leftarrow \text{Decompress}_q(c_1, d_u)$ 2: $v' \leftarrow \text{Decompress}_q(c_2, d_v)$ 3: $M \leftarrow v' - u' \times s$ 4: $m' \leftarrow \text{Decode}(M)$ 5: return m'	

```

int16_t barrett_reduce(int16_t a)
{
    int16_t t;
    const int16_t v = ((1U << 26) + KYBER_Q/2)/KYBER_Q;

    t = (int32_t)v*a >> 26;
    t *= KYBER_Q;
    return a - t;
}
  
```

✓ KYBER Round 3 Reference c code Barrett Reduction 함수

■ 분석 방법 [4]

❖ Barrett Reduction 함수의 구현 취약성


```

int16_t barrett_reduce(int16_t a)
{
    int16_t t;
    const int16_t v = ((1U << 26) + KYBER_Q/2)/KYBER_Q;


    t = (int32_t)v * a >> 26;
    t *= KYBER_Q;
    return a - t;
}
  
```

➤ 입력값 a 에 따른 t 값에 의해 전력 차이 발생

a	t	HW(t)
$a \in [3329, 6656]$	3329	4
$a \in [0, 3328]$	0	0
$a \in [-3328, -1]$	-3329	16

➔ Power \approx 

➔ Power \approx 

➔ Power \approx  HW 차가 큰
두 값을 가지도록
선택 암호문 구성

■ 격자 기반 PQC 부채널 분석 동향 [5]

논문	공격 기법	대상 연산	구현 코드	실험 환경	공격 파형 수	지표	성공률
[SH18]	SPA	Gaussian Sampler (CDT sampler)	Round 2 Ref.	ATmega128	1	비밀키 전체	99 %

❖ 격자 기반 암호의 CDT Sampler를 공격하여 비밀키 복구

➤ FrodoKEM, Lizard

FrodoPKE.KeyGen()	
Input	None
Output	pk, sk
1: $\text{see } d_A \leftarrow \$ U(\{0,1\}^{en_{seed_A}})$ 2: $\text{see } d_{SE} \leftarrow \$ U(\{0,1\}^{en_{seed_{SE}}})$ 3: $A \leftarrow \text{Frodo.Gen}(\text{see } d_A)$ 4: $S, E \leftarrow \text{Frodo.SampleMatrix}(\text{see } d_{SE})$ 6: $B \leftarrow AS + E$ 7: $(pk, sk) \leftarrow ((\text{see } d_A, B), S)$ 8: return (pk, sk)	

```

void frodo_sample_n(uint16_t *s, const size_t n)
{
    unsigned int i, j;

    for (i = 0; i < n; ++i) {
        uint16_t sample = 0;
        uint16_t prnd = s[i] >> 1;
        uint16_t sign = s[i] & 0x1;
        for (j = 0; j < (unsigned int)(CDF_TABLE_LEN - 1); j++) {
            sample += (uint16_t)(CDF_TABLE[j] - prnd) >> 15;
        }
        s[i] = ((-sign) ^ sample) + sign;
    }
}
  
```

✓ FrodoKEM Round 3 Reference c code CDT sampler 함수

■ 분석 방법 [5]

❖ CDT Sampler 함수의 구현 취약성 (FrodoKEM)

```
void frodo_sample_n(uint16_t *s, const size_t n)
{
    unsigned int i, j;

    for (i = 0; i < n; ++i) {
        uint16_t sample = 0;
        uint16_t prnd = s[i] >> 1;
        uint16_t sign = s[i] & 0x1;
        for (j = 0; j < (unsigned int)(CDF_TABLE_LEN - 1); j++) {
            sample += (uint16_t)(CDF_TABLE[j] - prnd) >> 15;
        }
        s[i] = ((-sign) ^ sample) + sign;
    }
}
```

$$\frac{E(\text{CDF_TABLE}[j] \geq \text{prnd})}{\text{sample에 더해지는 값} = 0} \rightarrow \text{HW} = 4.5 \rightarrow \text{Power} \approx \text{⚡}$$


$$\frac{E(\text{CDF_TABLE}[j] < \text{prnd})}{\text{sample에 더해지는 값} = 1} \rightarrow \text{HW} = 11.5 \rightarrow \text{Power} \approx \text{⚡}$$


■ 분석 방법 [5]

❖ CDT Sampler 함수의 구현 취약성 (FrodoKEM)

```
void frodo_sample_n(uint16_t *s, const size_t n)
{
    unsigned int i, j;

    for (i = 0; i < n; ++i) {
        uint16_t sample = 0;
        uint16_t prnd = s[i] >> 1;
        uint16_t sign = s[i] & 0x1;
        for (j = 0; j < (unsigned int)(CDF_TABLE_LEN - 1); j++) {
            sample += (uint16_t)(CDF_TABLE[j] - prnd) >> 15;
        }
        s[i] = ((-sign) ^ sample) + sign;
    }
}
```

sign = 0 → -sign = 0x0000 → HW = 0 → Power ≈ 

sign = 1 → -sign = 0xffff → HW = 16 → Power ≈ 

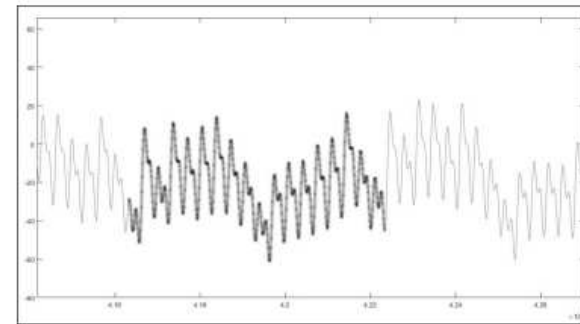
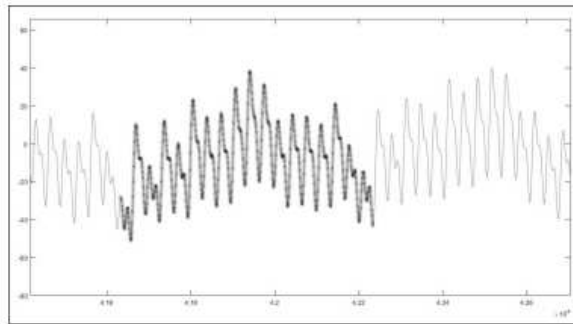
➤ 이로 인해 sample에 더해지는 값과 sign 비트를 복구할 수 있음

■ 분석 결과 [5]

❖ 단일 파형 분석을 통해 성공률 99%로 비밀키 복구



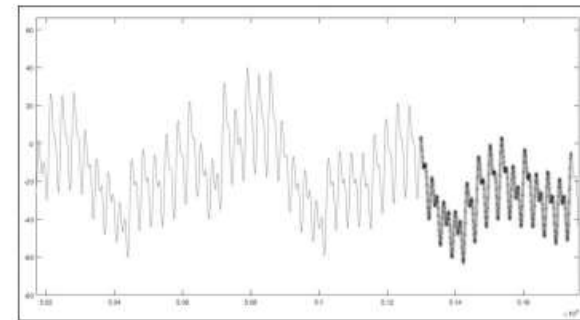
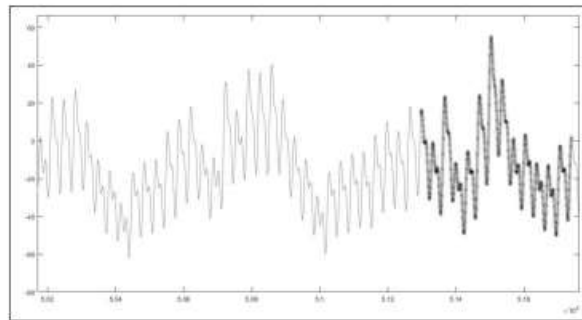
➤ sample에 더해지는 값



✓ 0인 경우

✓ 1인 경우

➤ sign 값



✓ 0인 경우

✓ 1인 경우


■ 격자 기반 PQC 부채널 분석 동향 [6]

논문	공격 기법	대상 연산	구현 코드	실험 환경	공격 파형 수	지표	성공률
[GJN20]	Timing Attack	암호문 비교 연산	Round 2 Ref.	i5-4200U CPU	약 2^{30}	비밀키 전체	100 %

❖ FrodoKEM Decapsulation의 암호문 비교에서 연산시간 취약점을 이용한 선택 암호문 공격

❖ FrodoKEM Round 2 버전에서 수행됨

FrodoKEM.Decaps()	
Input	$c = (C_1, C_2), sk$
Output	Session Key ss'
1: $m' \leftarrow \text{FrodoPKE.Dec}(c, sk)$ 2: $(r', k') \leftarrow H(H(pk) m')$ 3: $ss_0' \leftarrow H(c k')$ 4: $ss_1' \leftarrow H(c s)$ 5: $c' \leftarrow \text{FrodoPKE.Enc}(m', pk; r')$ 6: if $c = c'$ then $ss' \leftarrow ss_0'$ 7: else then $ss' \leftarrow ss_1'$ 8: return ss'	



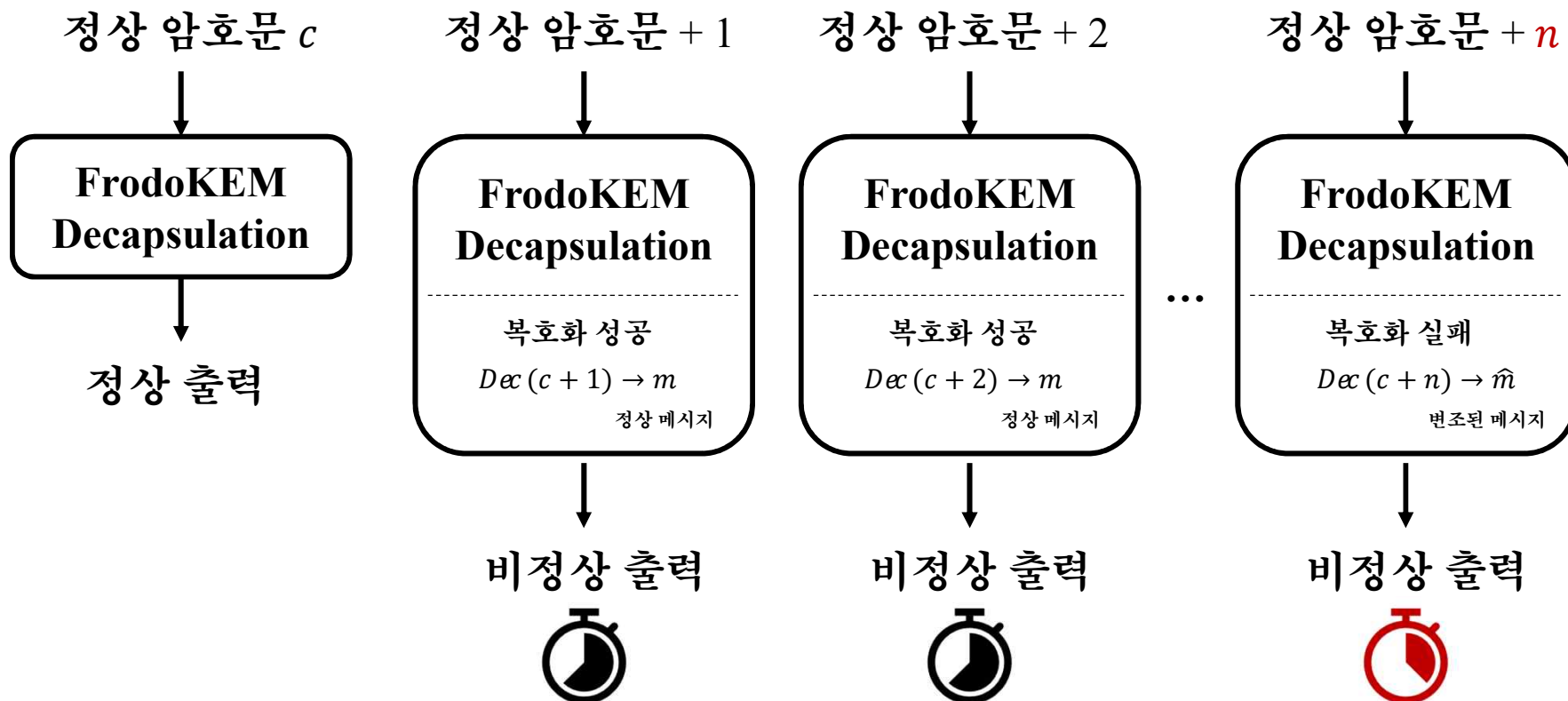
```

// Is (Bp == BBp & C == CC) = true Non Constant time
if (memcmp(Bp, BBp, 2*PARAMS_N*PARAMS_NBAR) == 0 &&
    memcmp(C, CC, 2*PARAMS_NBAR*PARAMS_NBAR) == 0) {
    // Load k' to do ss = F(ct || k')
    memcpy(Fin_k, kprime, CRYPTO_BYTES);
} else {
    // Load s to do ss = F(ct || s)
    memcpy(Fin_k, sk_s, CRYPTO_BYTES);
}
  
```

✓ FrodoKEM Round 2 Reference c code comparison 과정

■ 분석 방법 [6]

❖ Comparison 연산에서 선택 암호문 공격

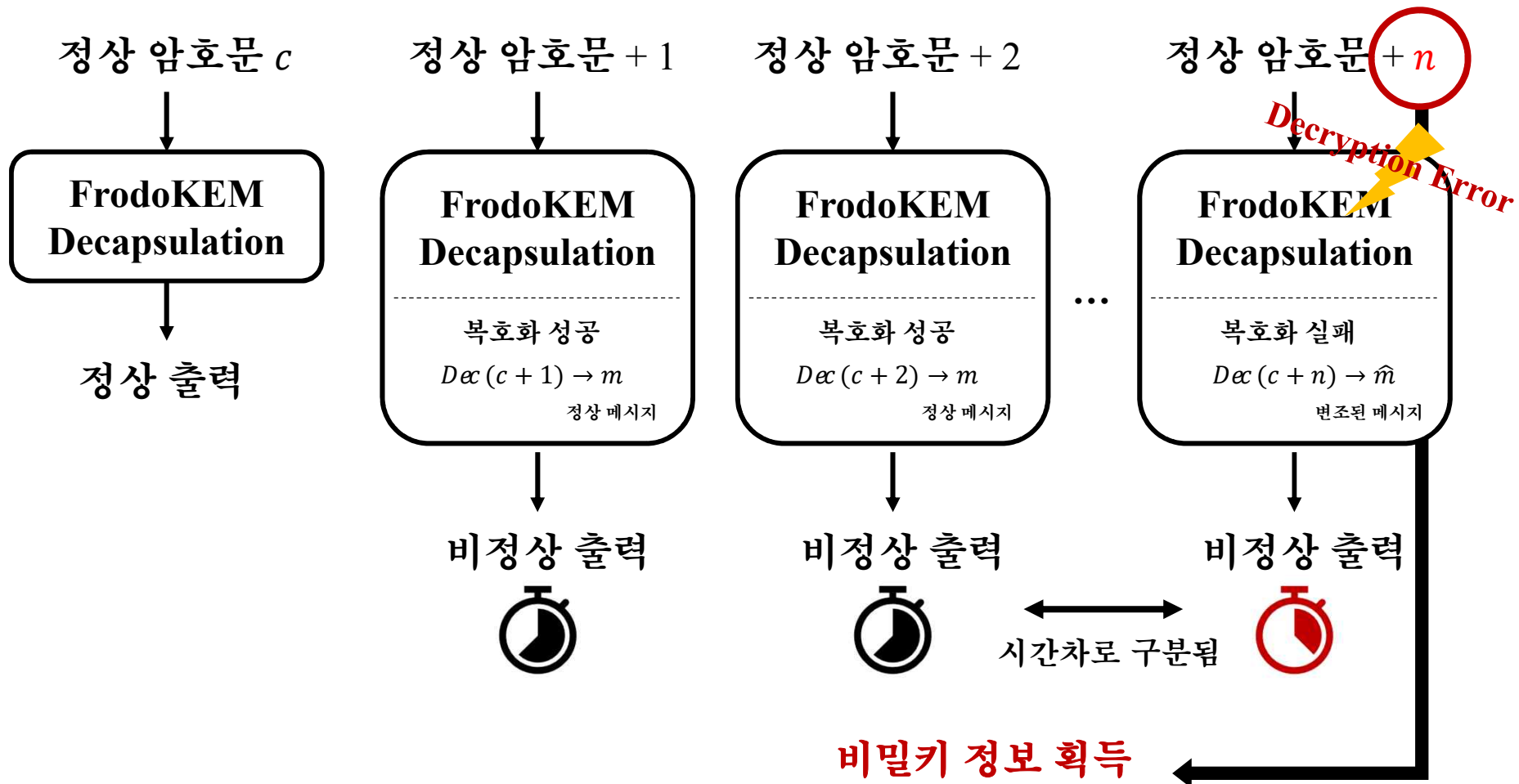


✓ 에러가 추가된 부분만 다르므로
비교 과정 시 시간이 오래 걸림

✓ 암호문이 아예 다르므로
비교 과정 시 시간이 짧게 걸림

■ 분석 방법 [6]

❖ Comparison 연산에서 선택 암호문 공격



■ 분석 이후 [6]

❖ FrodoKEM Round 3 버전의 문서와 코드에서 해당 암호문 비교연산 부분에 대한 수정

September 30, 2020

- Submission to NIST Post-Quantum Cryptography standardization process round 3.
- Guo, Johansson, and Nilsson (Reference [66] in the specification document) identified a key-recovery timing attack on the reference implementation of FrodoKEM due to branching in FrodoKEM.Decaps. The pseudocode of Algorithm 14 (FrodoKEM.Decaps) was updated to make it clearer that these operations needed to be completed in constant-time, and updated our reference and optimized implementations. A brief discussion of this issue was added to Section 6.1 and performance figures were updated in Section 3.
- Parameter search scripts are now compatible with Python3.
- The parameter search procedure (Section 2.4.2) was reconciled with the scripts. Thanks to Sabrina Sewer for pointing out the discrepancy.
- A new Section 5.2.4 has been added, detailing a refinement of the security analysis with respect to

NIST Round 3 FrodoKEM changes 문서

1 부채널 분석 관점에서의 PKC & PQC

2 PQC 알고리즘에 대한 부채널 분석 동향

✓ 3 상수시간 알고리즘은 항상 안전한가?

4 최적화 구현과 부채널 분석 취약점의 상관관계

Constant-time 암호문 비교 연산에 대한 부채널 분석 [7]

❖ FrodoKEM

➤ Round 2 → Round 3 암호문 비교 코드 수정

```
// Is (Bp == BBp & C == CC) = true Non Constant time
if (memcmp(Bp, BBp, 2*PARAMS_N*PARAMS_NBAR) == 0 &&
    memcmp(C, CC, 2*PARAMS_NBAR*PARAMS_NBAR) == 0) {
```

```
// If (Bp == BBp & C == CC) then ss = F(ct || k'), else ss = F(ct || s)
// Needs to avoid branching on secret data as per:
// Qian Guo, Thomas Johansson, Alexander Nilsson. A key-recovery
// timing attack on post-quantum primitives using the Fujisaki-Okamoto
// transformation and its application on FrodoKEM. In CRYPTO 2020.
```

```
int8_t selector = ct_verify(Bp, BBp, PARAMS_N*PARAMS_NBAR) |
                  ct_verify(C, CC, PARAMS_NBAR*PARAMS_NBAR);
```

```
int8_t ct_verify(const uint16_t *a, const uint16_t *b, size_t len)
{ // Compare two arrays in constant time.
  // Returns 0 if the byte arrays are equal, -1 otherwise.
  uint16_t r = 0;

  for (size_t i = 0; i < len; i++) {
    r |= a[i] ^ b[i];
  }

  r = -(int16_t)r >> (8*sizeof(uint16_t)-1);
  return (int8_t)r;
}
```

Constant time!!

Algorithm. FrodoKEM verify

Input Two ciphertexts (C, C')

Output semi_selector \mathfrak{x}

```
1: uint16_t  $\mathfrak{x}$  = 0
2: for  $i = 0; i < \text{size}(C); i \leftarrow i + 1$  do
3:    $\mathfrak{x} \mid= C[i] \oplus C'[i]$ 
4: end for
5:  $\mathfrak{x} = (-\mathfrak{x}) \gg 15$ 
6: return (int8_t) $\mathfrak{x}$ 
```

Constant-time 암호문 비교 연산에 대한 부채널 분석

❖ FrodoKEM


➤ Round 3에서도 잔여 취약점 발생


Algorithm. FrodoKEM verify	
Input	Two ciphertexts (C, C')
Output	semi_selector \mathfrak{s}
<pre> 1: uint16_t $\mathfrak{s} = 0$ 2: for $i = 0; i < \text{size}(C); i \leftarrow i + 1$ do 3: $\mathfrak{s} \mid= C[i] \oplus C'[i]$ 4: end for 5: $\mathfrak{s} = (-\mathfrak{s}) \gg 15$ 6: return (int 8_t)\mathfrak{s} </pre>	

Round 2 Round 3

시간정보 → 전력정보

✓ 부채널 정보의 종류가 바뀌었을 뿐
같은 방법론으로 분석 가능

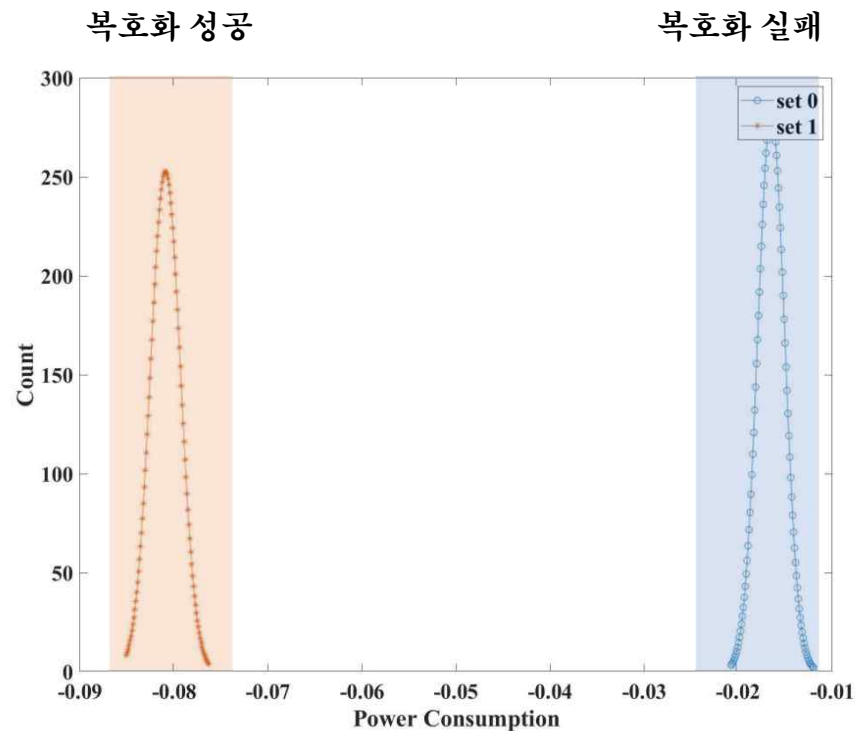
$C = C' \rightarrow \mathfrak{s} = 0x00 \rightarrow \text{HW}(\mathfrak{s}) = 0 \rightarrow \text{Power} \approx$ 

$C \neq C' \rightarrow \mathfrak{s} = 0xff \rightarrow \text{HW}(\mathfrak{s}) = 8 \rightarrow \text{Power} \approx$ 

Constant-time 암호문 비교 연산에 대한 부채널 분석

❖ FrodoKEM

- 복호화 성공 실패에 따른 소비 전력 파형의 분포



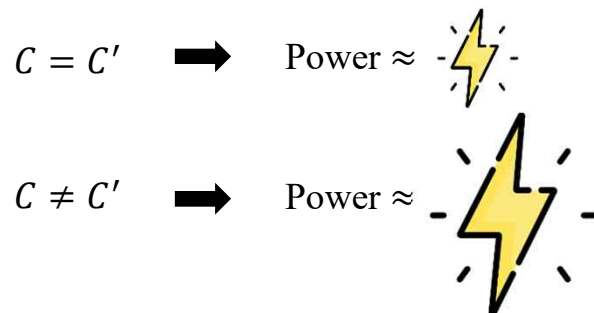
- ✓ 기존 연구에 비해 높은 성공률과 적은 파형으로 비밀키 분석이 가능함

Constant-time 암호문 비교 연산에 대한 부채널 분석

❖ KYBER와 SABER에서도 같은 취약성 발생

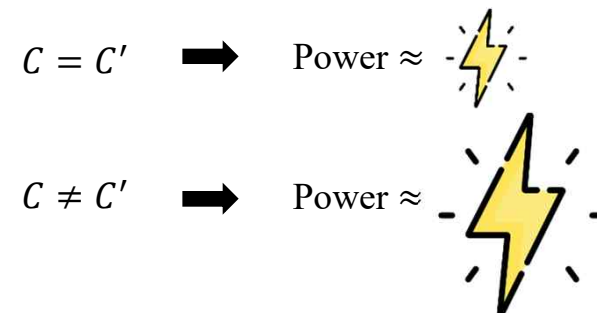
➤ KYBER

Algorithm. KYBER verify	
Input	Two ciphertexts (C, C')
Output	semi_selector \varnothing
<pre> 1: uint16_t $\varnothing = 0$ 2: for $i = 0; i < \text{size}(C); i \leftarrow i + 1$ do 3: $\varnothing \mid = C[i] \oplus C'[i]$ 4: end for 5: return $(-(\text{uint64_t})\varnothing) \gg 63;$ </pre>	



➤ SABER

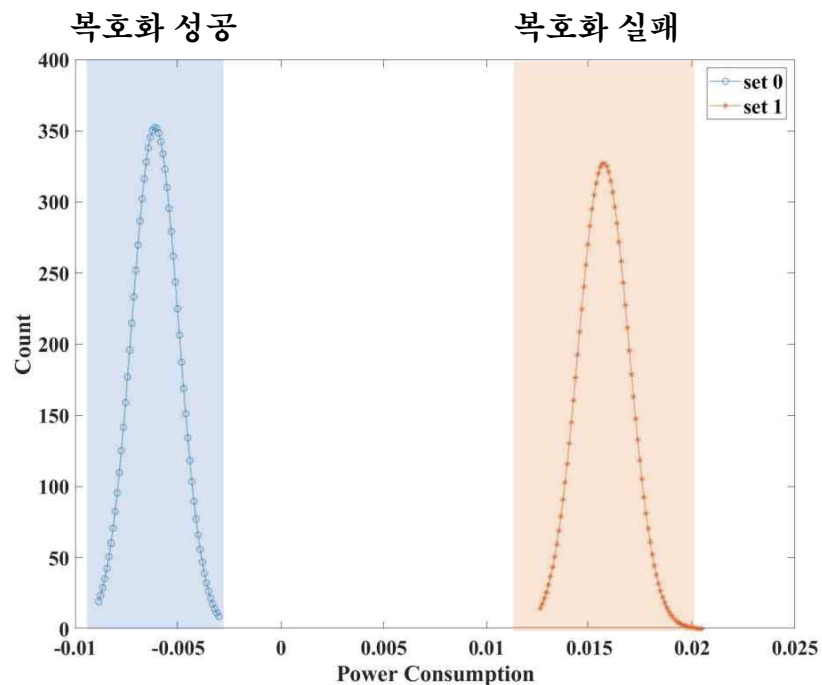
Algorithm. SABER verify	
Input	Two ciphertexts (C, C')
Output	semi_selector \varnothing
<pre> 1: uint16_t $\varnothing = 0$ 2: for $i = 0; i < \text{size}(C); i \leftarrow i + 1$ do 3: $\varnothing \mid = C[i] \oplus C'[i]$ 4: end for 5: $\varnothing = (-\varnothing) \gg 63$ 6: return $(-(\text{uint8_t})\varnothing);$ </pre>	



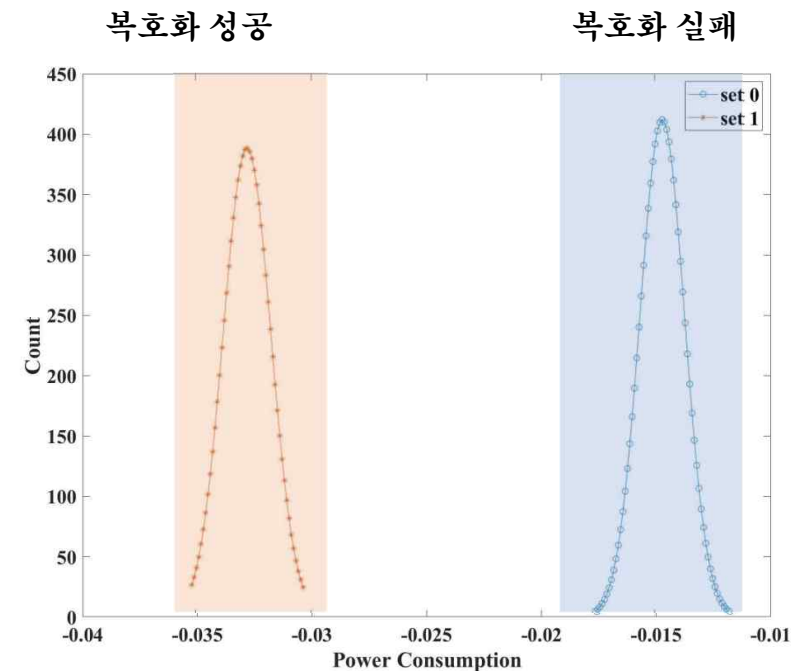
Constant-time 암호문 비교 연산에 대한 부채널 분석

❖ 복호화 성공 실패에 따른 소비 전력 파형의 분포

➤ KYBER



➤ SABER



- ✓ 다음과 같은 취약성을 통해 FrodoKEM은 에러다항식을 바로 복구할 수 있지만,
KYBER와 SABER는 에러다항식의 범위를 줄일 수 있음

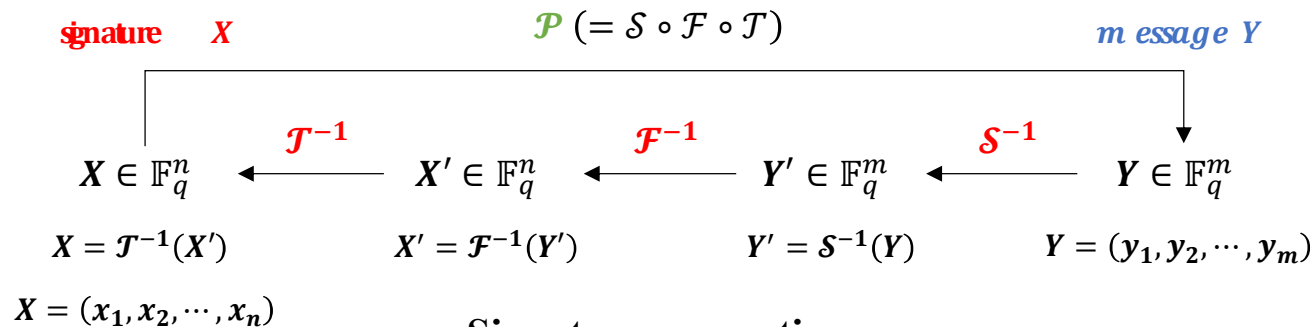
- 1 부채널 분석 관점에서의 PKC & PQC
- 2 PQC 알고리즘에 대한 부채널 분석 동향
- 3 상수시간 알고리즘은 항상 안전한가?
- ✓ 4 최적화 구현과 부채널 분석 취약점의 상관관계

다변수 기반 서명 알고리즘

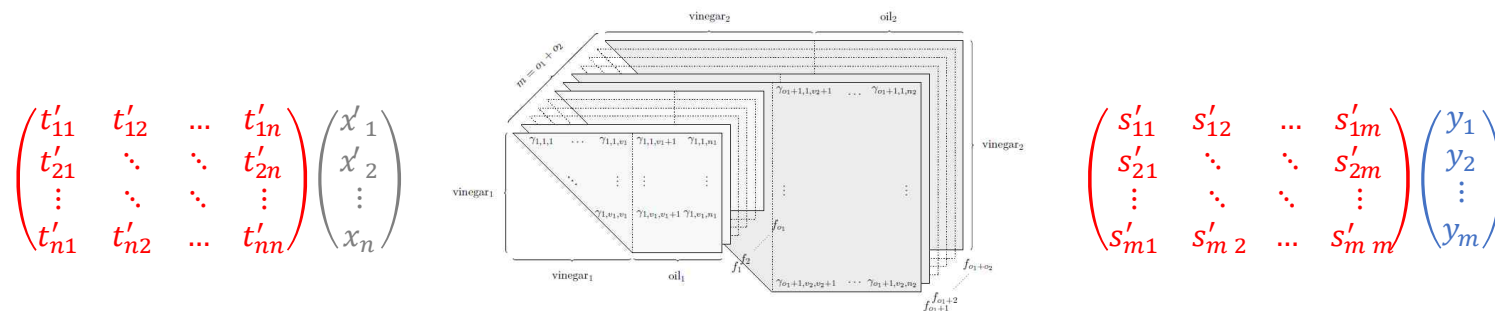
❖ Rainbow 서명 알고리즘

- UOV의 개선된 형태이며, UOV보다 작은 키 길이와 높은 효율성을 가짐

Signature verification



Signature generation



다변수 기반 서명 알고리즘

Rainbow 서명 알고리즘

➤ Central map인 \mathcal{F} 를 감추기 위해 행렬 S, T 사용

✓ S, T 가 복구된다면 공개키 $P = S \circ \mathcal{F} \circ T$ 를 통해 \mathcal{F} 계산 가능 → 비밀키 노출

Rainbow에 대한 부채널 분석 시나리오

부채널 분석을 통한
 S^{-1} 행렬 복구

➤ 1단계 : S^{-1} 행렬 복구

✓ 1차 상관 전력 분석을 이용한 S^{-1} 행렬 성분 복구

부채널 분석을 통한
 T^{-1} 행렬 복구

➤ 2단계 : T^{-1} 행렬 복구

✓ 1차 상관 전력 분석 제한 (행렬의 모든 원소가 랜덤)

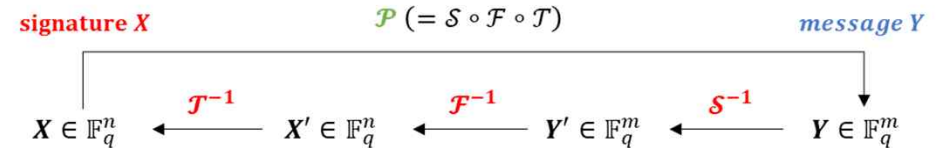
✓ 특정 구조에 대한 1차 상관 전력 분석을 통해 T^{-1} 행렬 복구

수학적 연산을 통한
 \mathcal{F}^{-1} 행렬 복구

➤ 3단계 : \mathcal{F}^{-1} 행렬 복구

✓ 1, 2단계 에서 복구한 S^{-1}, T^{-1} 을 이용

✓ 공개 값 $P(= S \circ \mathcal{F} \circ T)$ 에서 \mathcal{F} 를 복구



다변수 기반 서명 알고리즘

❖ Rainbow에 대한 부채널 분석 시나리오

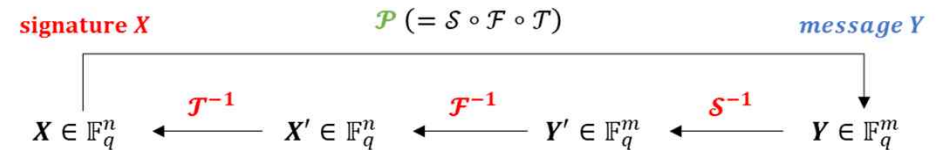
부채널 분석을 통한
 \mathcal{T}^{-1} 행렬 복구

➤ 2단계 : \mathcal{T}^{-1} 행렬 복구

✓ 1차 상관 전력 분석 제한 (행렬의 모든 원소가 랜덤)

❖ 상관 전력 분석

➤ 암호동작 동안에 수집되어진 전력소비와 중간값의 상관관계를 이용하여 분석하는 기술



다변수 기반 서명 알고리즘

Rainbow에 대한 부채널 분석 시나리오

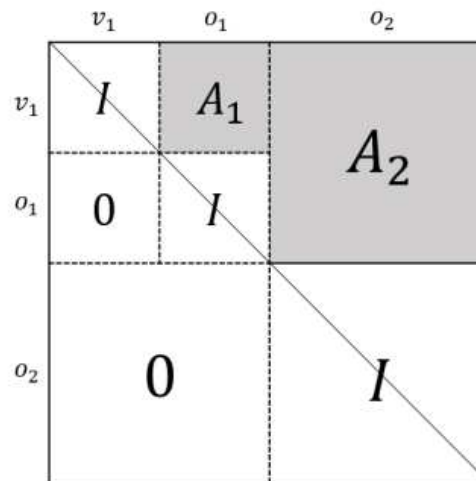
부채널 분석을 통한
 \mathcal{T}^{-1} 행렬 복구

➤ 2단계 : \mathcal{T}^{-1} 행렬 복구

- ✓ 1차 상관 전력 분석 제한 (행렬의 모든 원소가 랜덤)
- ✓ 특정 구조에 대한 1차 상관 전력 분석을 통해 \mathcal{T}^{-1} 행렬 복구

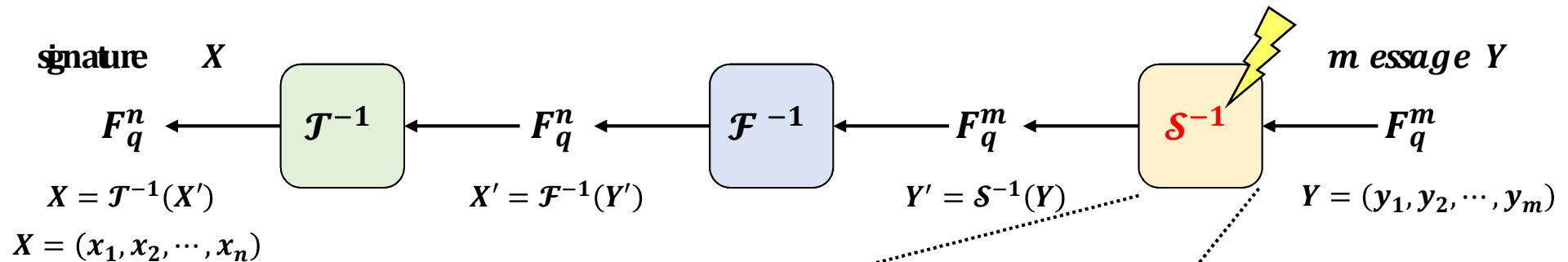


- [CHES 2012]에서 효율적인 구현을 위해 Rainbow에 그림과 같은 비밀키를 사용하도록 제안



■ Rainbow 서명 알고리즘에 대한 부채널 분석 [9]

❖ 상관 전력 분석을 이용한 \mathcal{S}^{-1} 행렬 복구



$$\begin{pmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_m \end{pmatrix} = \begin{pmatrix} s'_{11} & s'_{12} & \dots & s'_{1m} \\ s'_{21} & \ddots & \ddots & s'_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ s'_{m1} & s'_{m2} & \dots & s'_{mm} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

$$s'_{11} \cdot y_1 \oplus s'_{12} \cdot y_2 \oplus \dots \oplus s'_{1m} \cdot y_m = y'_1$$

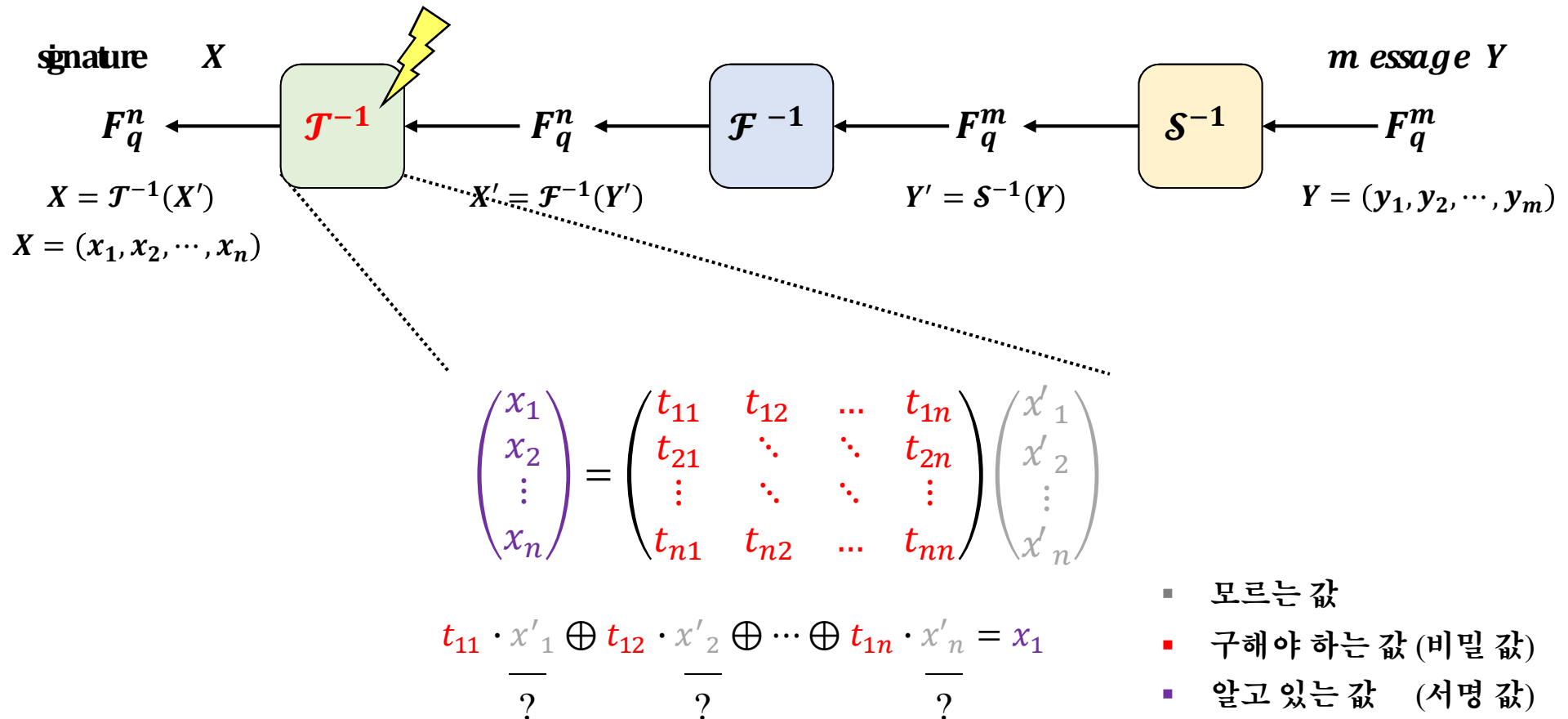
중간 값 : $guess \cdot y_1$ (예시)

- 구해야 하는 값 (비밀 값)
- 알고 있는 값 (메시지 값)

➤ 위와 같이 중간 값을 추측하여 1차 상관 전력 분석을 수행 $\Rightarrow \mathcal{S}^{-1}$ 행렬 복구

■ Rainbow 서명 알고리즘에 대한 부채널 분석 [9]

❖ 상관 전력 분석을 이용한 \mathcal{T}^{-1} 행렬 복구 (행렬의 모든 원소가 랜덤)



➤ \mathcal{T}^{-1} 의 연산에서 알고 있는 값이 없음 \Rightarrow 중간값 추측 불가능 \Rightarrow 상관 전력 분석에 한계점 존재

■ Rainbow 서명 알고리즘에 대한 부채널 분석 [9]

❖ 상관 전력 분석을 이용한 T^{-1} 행렬 복구 ([CHES 2012] 행렬을 사용하는 경우)

$$\begin{array}{|c|c|c|} \hline & \overbrace{\quad}^{o_1} & \overbrace{\quad}^{o_2} \\ \hline \begin{array}{|c|c|} \hline I & T'_1 \\ \hline 0 & I \end{array} & \left. \begin{array}{|c|} \hline T'_2 \\ \hline \end{array} \right\} v & \\ \hline \end{array} \left. \vphantom{\begin{array}{|c|c|} \hline I & T'_1 \\ \hline 0 & I \end{array}} \right\} v + o_1$$

$$T^{-1} = \begin{pmatrix} 01 & 00 & t_{13} & t_{14} & t_{15} & t_{16} & t_{17} & t_{18} \\ 00 & 01 & t_{23} & t_{24} & t_{25} & t_{26} & t_{27} & t_{28} \\ 00 & 00 & 01 & 00 & t_{35} & t_{36} & t_{37} & t_{38} \\ 00 & 00 & 00 & 01 & t_{45} & t_{46} & t_{47} & t_{48} \\ 00 & 00 & 00 & 00 & 01 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 01 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 01 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01 \end{pmatrix}$$

$$T^{-1} \begin{pmatrix} 01 & 00 & t_{13} & t_{14} & t_{15} & t_{16} & t_{17} & t_{18} \\ 00 & 01 & t_{23} & t_{24} & t_{25} & t_{26} & t_{27} & t_{28} \\ 00 & 00 & 01 & 00 & t_{35} & t_{36} & t_{37} & t_{38} \\ 00 & 00 & 00 & 01 & t_{45} & t_{46} & t_{47} & t_{48} \\ 00 & 00 & 00 & 00 & 01 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 01 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 01 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01 \end{pmatrix} \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \\ x'_5 \\ x'_6 \\ x'_7 \\ x'_8 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix}$$

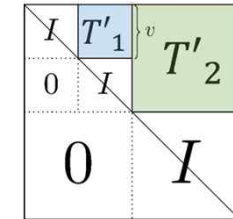
Signature X

- 모르는 값
- 구해야 하는 값 (비밀 값)
- 알고 있는 값 (서명 값)

■ Rainbow 서명 알고리즘에 대한 부채널 분석 [9]

❖ 상관 전력 분석을 이용한 T^{-1} 행렬 복구 ([CHES 2012] 행렬을 사용하는 경우)

➤ 1단계 : 서명 값과 동일한 값을 결정(모르는 값의 결정)



$$\begin{pmatrix}
 01 & 00 & t_{13} & t_{14} & t_{15} & t_{16} & t_{17} & t_{18} \\
 00 & 01 & t_{23} & t_{24} & t_{25} & t_{26} & t_{27} & t_{28} \\
 00 & 00 & 01 & 00 & t_{35} & t_{36} & t_{37} & t_{38} \\
 00 & 00 & 00 & 01 & t_{45} & t_{46} & t_{47} & t_{48} \\
 00 & 00 & 00 & 00 & 01 & 00 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 01 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 01 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01
 \end{pmatrix}
 \begin{pmatrix}
 x'_1 \\
 x'_2 \\
 x'_3 \\
 x'_4 \\
 x'_5 \\
 x'_6 \\
 x'_7 \\
 x'_8
 \end{pmatrix}
 =
 \begin{pmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 x_6 \\
 x_7 \\
 x_8
 \end{pmatrix}$$

- 모르는 값
- 구해야 하는 값
- 알고 있는 값

■ Rainbow 서명 알고리즘에 대한 부채널 분석 [9]

❖ 상관 전력 분석을 이용한 T^{-1} 행렬 복구 ([CHES 2012] 행렬을 사용하는 경우)

➤ 1단계 : 서명 값과 동일한 값을 결정(모르는 값의 결정)

I	T'_1	T'_2
0	I	
0	I	

$$\begin{pmatrix}
 01 & 00 & t_{13} & t_{14} & t_{15} & t_{16} & t_{17} & t_{18} \\
 00 & 01 & t_{23} & t_{24} & t_{25} & t_{26} & t_{27} & t_{28} \\
 00 & 00 & 01 & 00 & t_{35} & t_{36} & t_{37} & t_{38} \\
 00 & 00 & 00 & 01 & t_{45} & t_{46} & t_{47} & t_{48} \\
 00 & 00 & 00 & 00 & 01 & 00 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 01 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 01 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01
 \end{pmatrix}
 \begin{pmatrix}
 x'_1 \\
 x'_2 \\
 x'_3 \\
 x'_4 \\
 x'_5 \\
 x'_6 \\
 x'_7 \\
 x'_8
 \end{pmatrix}
 =
 \begin{pmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 x_6 \\
 x_7 \\
 x_8
 \end{pmatrix}$$

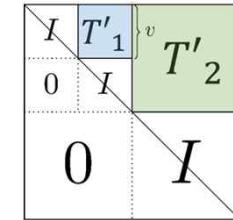
- 모르는 값
- 구해야 하는 값
- 알고 있는 값

$$\checkmark x'_8 = x_8, x'_7 = x_7, x'_6 = x_6, x'_5 = x_5$$

■ Rainbow 서명 알고리즘에 대한 부채널 분석 [9]

❖ 상관 전력 분석을 이용한 T^{-1} 행렬 복구 ([CHES 2012] 행렬을 사용하는 경우)

➤ 2단계 : 상관 전력 분석을 통한 T'_2 행렬 성분분석



$$\begin{pmatrix}
 01 & 00 & t_{13} & t_{14} & t_{15} & t_{16} & t_{17} & t_{18} \\
 00 & 01 & t_{23} & t_{24} & t_{25} & t_{26} & t_{27} & t_{28} \\
 00 & 00 & 01 & 00 & t_{35} & t_{36} & t_{37} & t_{38} \\
 00 & 00 & 00 & 01 & t_{45} & t_{46} & t_{47} & t_{48} \\
 00 & 00 & 00 & 00 & 01 & 00 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 01 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 01 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01
 \end{pmatrix}
 \begin{pmatrix}
 x'_1 \\
 x'_2 \\
 x'_3 \\
 x'_4 \\
 x'_5 \\
 x'_6 \\
 x'_7 \\
 x'_8
 \end{pmatrix}
 =
 \begin{pmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 x_6 \\
 x_7 \\
 x_8
 \end{pmatrix}$$

■ 모르는 값
 ■ 구해야 하는 값
 ■ 알고 있는 값

$$1 \cdot x'_1 \oplus 0 \cdot x'_1 \oplus t_{13} \cdot x'_3 \oplus t_{14} \cdot x'_4 \oplus t_{15} \cdot x'_5 \oplus t_{16} \cdot x'_6 \oplus t_{17} \cdot x'_7 \oplus t_{18} \cdot x'_8 = x_1$$

$$0 \cdot x'_1 \oplus 1 \cdot x'_1 \oplus t_{23} \cdot x'_3 \oplus t_{24} \cdot x'_4 \oplus t_{25} \cdot x'_5 \oplus t_{26} \cdot x'_6 \oplus t_{27} \cdot x'_7 \oplus t_{28} \cdot x'_8 = x_2$$

⋮

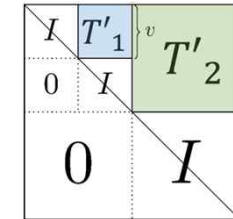
중간 값 : $guess \cdot x'_5, guess \cdot x'_6, guess \cdot x'_7, guess \cdot x'_8$

➤ 위와 같이 중간 값을 추측하여 1차 상관 전력 분석을 수행

■ Rainbow 서명 알고리즘에 대한 부채널 분석 [9]

❖ 상관 전력 분석을 이용한 T^{-1} 행렬 복구 ([CHES 2012] 행렬을 사용하는 경우)

➤ 3단계 : 추가 정보 계산



$$\begin{pmatrix}
 01 & 00 & t_{13} & t_{14} & t_{15} & t_{16} & t_{17} & t_{18} \\
 00 & 01 & t_{23} & t_{24} & t_{25} & t_{26} & t_{27} & t_{28} \\
 00 & 00 & 01 & 00 & t_{35} & t_{36} & t_{37} & t_{38} \\
 00 & 00 & 00 & 01 & t_{45} & t_{46} & t_{47} & t_{48} \\
 00 & 00 & 00 & 00 & 01 & 00 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 01 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 01 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01
 \end{pmatrix}
 \begin{pmatrix}
 x'_1 \\
 x'_2 \\
 x'_3 \\
 x'_4 \\
 x'_5 \\
 x'_6 \\
 x'_7 \\
 x'_8
 \end{pmatrix}
 =
 \begin{pmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 x_6 \\
 x_7 \\
 x_8
 \end{pmatrix}$$

- 모르는 값
- 구해야 하는 값
- 알고 있는 값

$$x_4 = x'_4 \oplus t_{45}x'_5 \oplus t_{46}x'_6 \oplus t_{47}x'_7 \oplus t_{48}x'_8$$

$$x'_4 = x_4 \oplus t_{45}x'_5 \oplus t_{46}x'_6 \oplus t_{47}x'_7 \oplus t_{48}x'_8$$

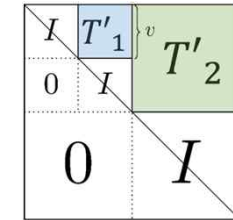
$$x_3 = x'_3 \oplus t_{35}x'_5 \oplus t_{36}x'_6 \oplus t_{37}x'_7 \oplus t_{38}x'_8$$

$$x'_3 = x_3 \oplus t_{35}x'_5 \oplus t_{36}x'_6 \oplus t_{37}x'_7 \oplus t_{38}x'_8$$

■ Rainbow 서명 알고리즘에 대한 부채널 분석 [9]

❖ 상관 전력 분석을 이용한 T^{-1} 행렬 복구 ([CHES 2012] 행렬을 사용하는 경우)

➤ 4단계 : 상관 전력 분석을 통한 T'_1 행렬 성분분석



$$\begin{pmatrix}
 01 & 00 & t_{13} & t_{14} & t_{15} & t_{16} & t_{17} & t_{18} \\
 00 & 01 & t_{23} & t_{24} & t_{25} & t_{26} & t_{27} & t_{28} \\
 00 & 00 & 01 & 00 & t_{35} & t_{36} & t_{37} & t_{38} \\
 00 & 00 & 00 & 01 & t_{45} & t_{46} & t_{47} & t_{48} \\
 00 & 00 & 00 & 00 & 01 & 00 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 01 & 00 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 01 & 00 \\
 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01
 \end{pmatrix}
 \begin{pmatrix}
 x'_1 \\
 x'_2 \\
 x'_3 \\
 x'_4 \\
 x'_5 \\
 x'_6 \\
 x'_7 \\
 x'_8
 \end{pmatrix}
 =
 \begin{pmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 x_6 \\
 x_7 \\
 x_8
 \end{pmatrix}$$

■ 모르는 값
 ■ 구해야 하는 값
 ■ 알고 있는 값

$$1 \cdot x'_1 \oplus 0 \cdot x'_1 \oplus t_{13} \cdot x'_3 \oplus t_{14} \cdot x'_4 \oplus t_{15} \cdot x'_5 \oplus t_{16} \cdot x'_6 \oplus t_{17} \cdot x'_7 \oplus t_{18} \cdot x'_8 = x_1$$

$$0 \cdot x'_1 \oplus 1 \cdot x'_1 \oplus t_{23} \cdot x'_3 \oplus t_{24} \cdot x'_4 \oplus t_{25} \cdot x'_5 \oplus t_{26} \cdot x'_6 \oplus t_{27} \cdot x'_7 \oplus t_{28} \cdot x'_8 = x_2$$

⋮

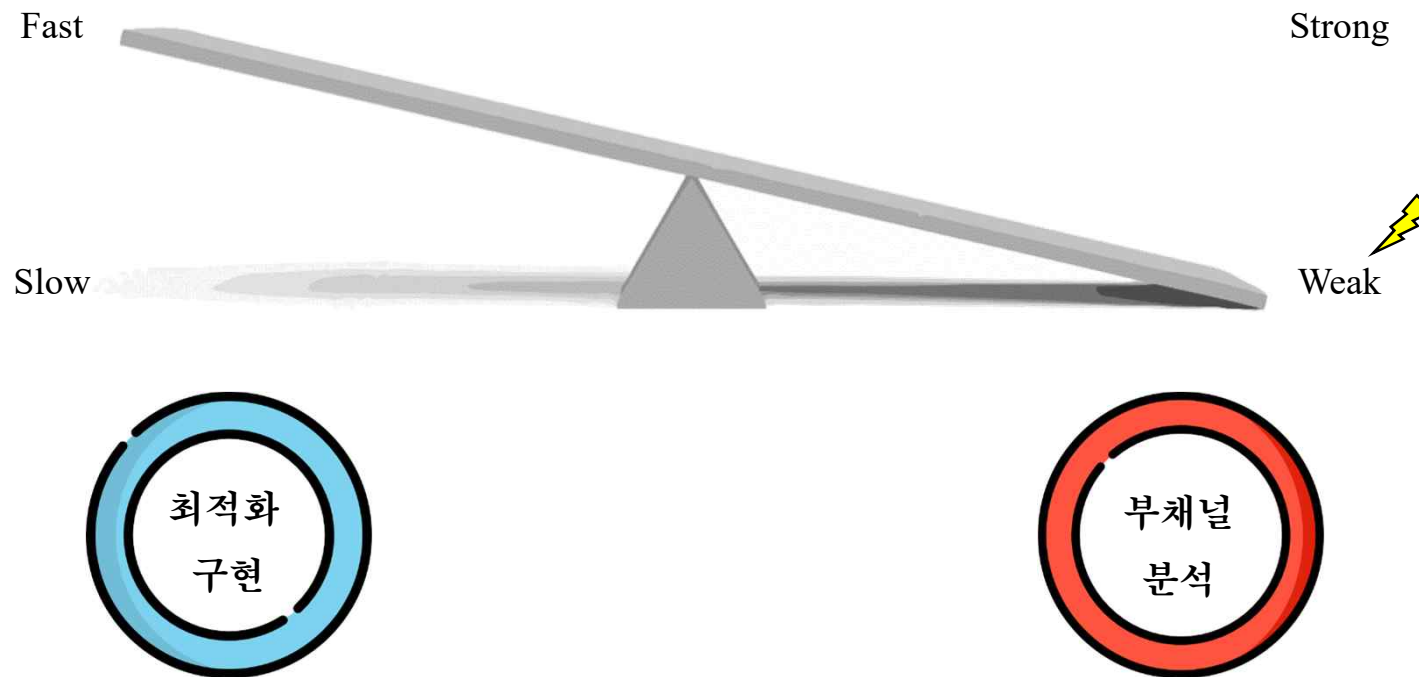
중간 값 : $guess \cdot x'_3, guess \cdot x'_4$

➤ 위와 같이 중간 값을 추측하여 1차 상관 전력 분석을 수행 $\Rightarrow T^{-1}$ 행렬 복구

■ 최적화 구현과 부채널 분석 취약점의 상관관계

❖ 효율성을 위한 최적화 구현이 부채널 분석 취약성을 발생시킬 수 있음

- Rainbow 알고리즘의 경우 키 사이즈 축소하여 효율성을 높였지만,
이로 인해 부채널 분석으로 비밀키를 복구할 수 있음 ⇒ 대응기법을 적용해야 함

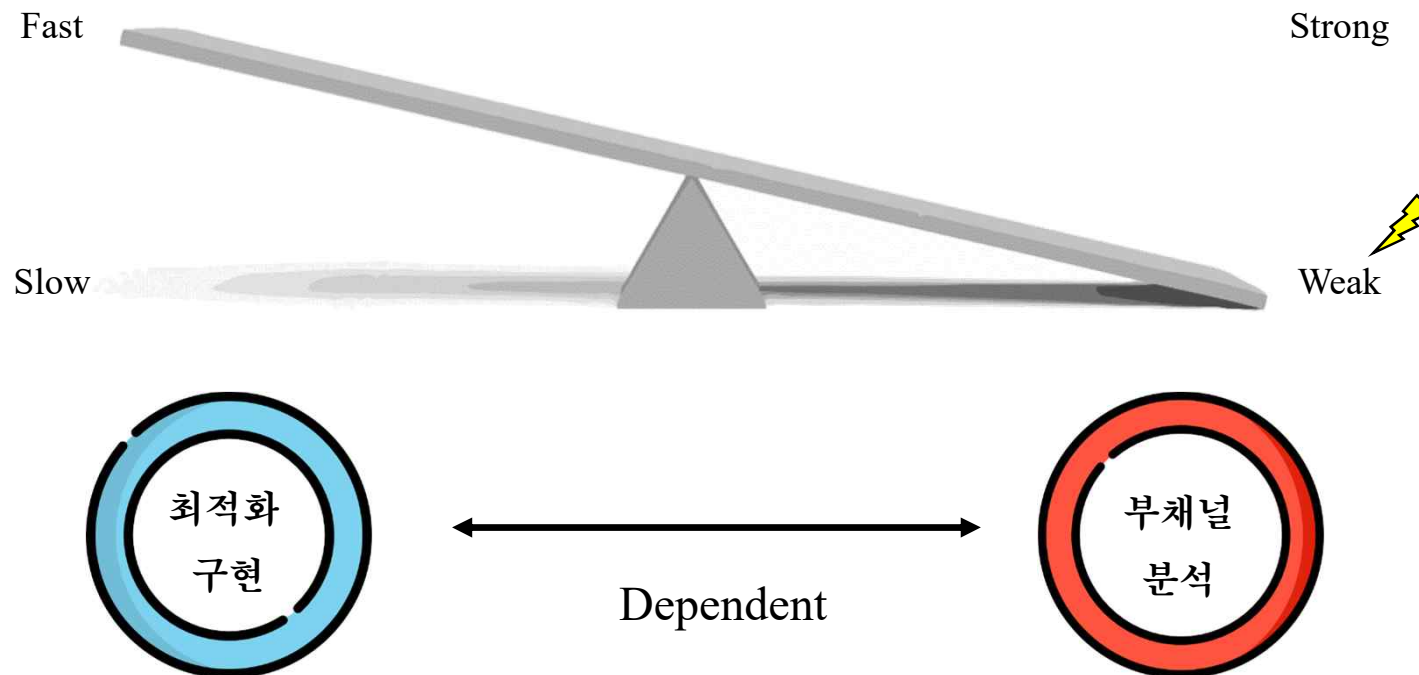


■ 최적화 구현과 부채널 분석 취약점의 상관관계

❖ 효율성을 위한 최적화 구현이 부채널 분석 취약성을 발생시킬 수 있음

➢ Rainbow 알고리즘의 경우 키 사이즈 축소하여 효율성을 높였지만,
이로 인해 부채널 분석으로 비밀키를 복구할 수 있음 ⇒ 대응기법을 적용해야 함

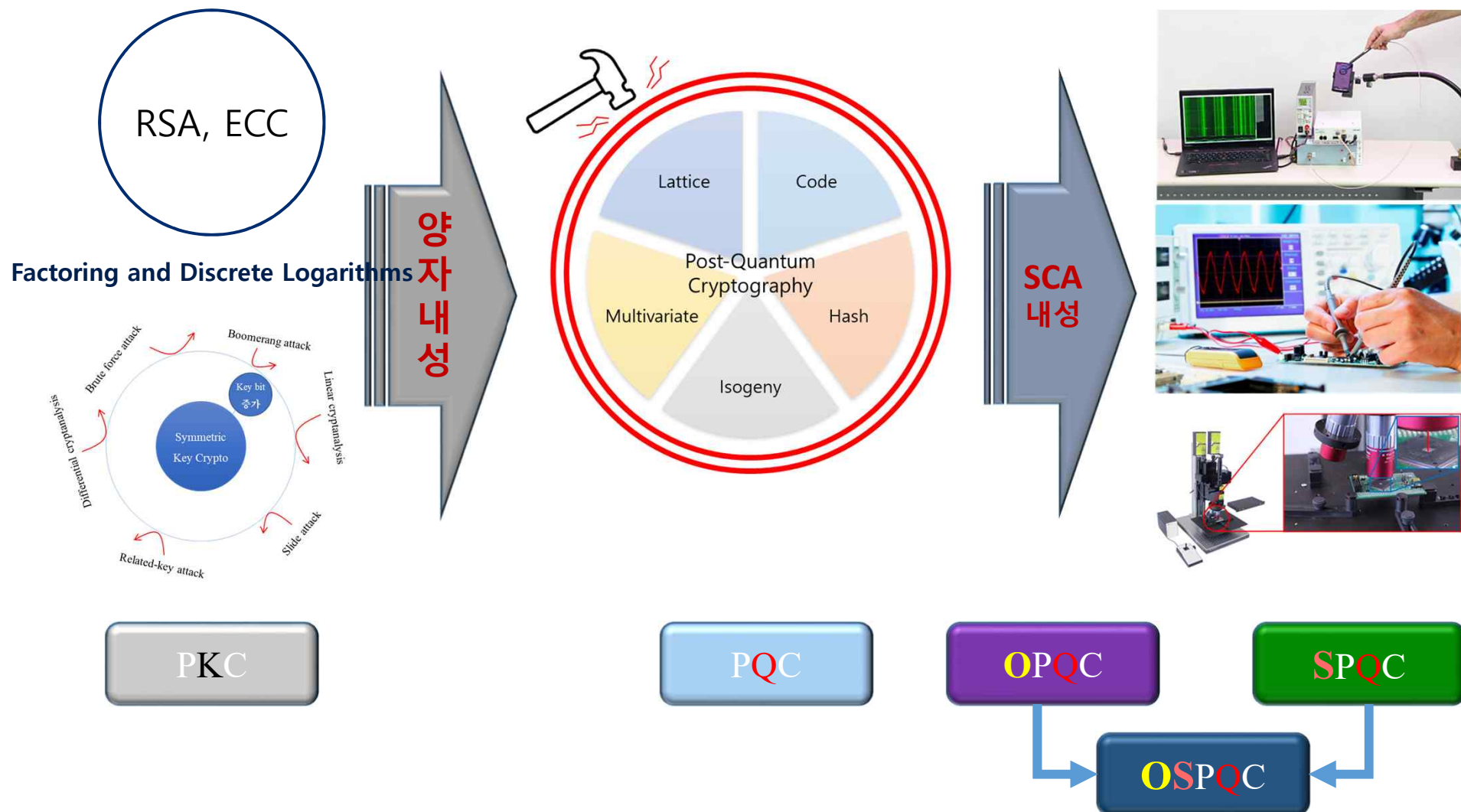
✓ 부채널 분석 취약점을 고려하여 최적화 구현을 수행해야 함



■ From PKC to PQC in Kookmin Univ.

- Chosen-Ciphertext Clustering Attack on CRYSTALS-KYBER using the Side-Channel Leakage of Barrett Reduction, IEEE Internet of Things Journal, 2022.
- Single-Trace Attack on NIST Round 3 Candidate Dilithium Using Machine Learning-Based Profiling, IEEE ACCESS, 2021.
- Single-Trace Attacks on Message Encoding in Lattice-Based KEMs, IEEE ACCESS, 2020.
- Novel Side-Channel Attacks on Quasi-Cyclic Code-Based Cryptography, CHES 2019, 2019.
- Side-Channel Attacks on Post-Quantum Signature Schemes based on Multivariate Quadratic Equations - Rainbow and UOV -, CHES 2018, 2018.
- 다량의 노이즈가 적용된 환경에서 KYBER 선택 암호문 공격의 현실적인 문제점, 2021
- NIST PQC Round 3 Finalist 후보 KYBER NTT 곱셈에 대한 상관전력분석, 2021.
- 마스크된 상수 시간 곱셈을 사용하는 준순환 부호 기반 암호 알고리즘에 대한 단일 파형 분석, 2019.
- [삼성에스디에스 주식회사/국민대학교산학협력단]
- 부채널 공격을 방지하기 위한 암호문 비교 장치 및 방법, 10-2021-0145850 (2021-10-28)
- 부채널 공격에 안전한 행렬 곱 연산을 수행하기 위한 장치 및 방법, 10-2019-0122600 (2019-10-02)

From PKC to PQC

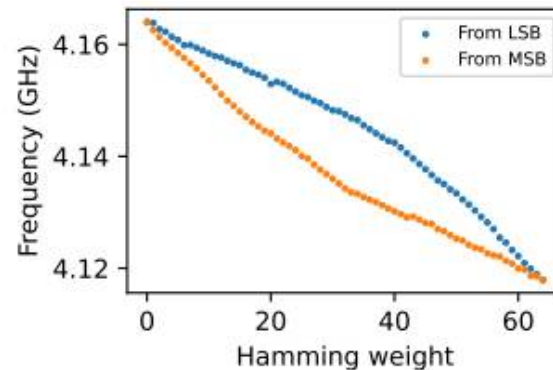


A detailed image of a circuit board with various components like capacitors, resistors, and integrated circuits. A golden key is placed on a dark green square in the center. The text "Q & A" is overlaid in white serif font.

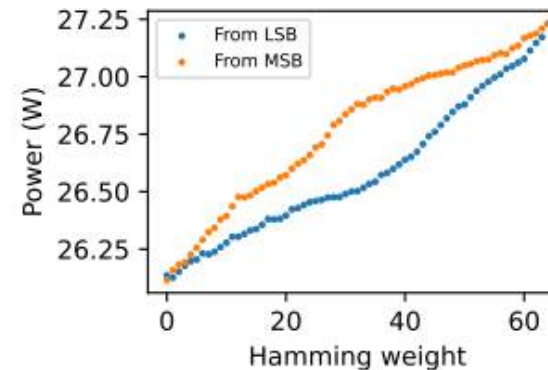
Q & A

Remote timing attack on x86 CPU [10]

- ❖ 기존의 상수 시간 구현은 **동적 주파수**를 고려하지 않음
 - 동적 주파수인 경우 상수 시간 구현 \Rightarrow 상수 시간
- ❖ X86 CPU는 소모 전력에 따라 동적으로 주파수를 조절하도록 설계됨
 - 상수 시간 구현에 대해 **Timing attack** 가능
- ❖ i7-9700에서 Hamming weight와 Frequency, Power와 Hamming weight의 관계



(a) Frequency vs HW

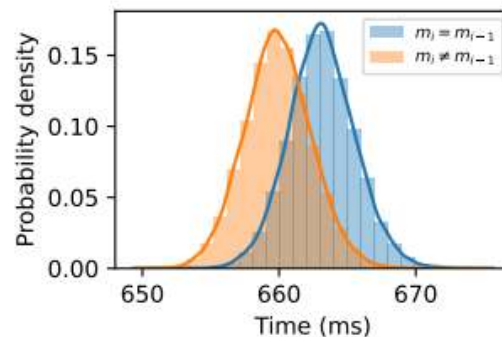


(b) Power vs HW

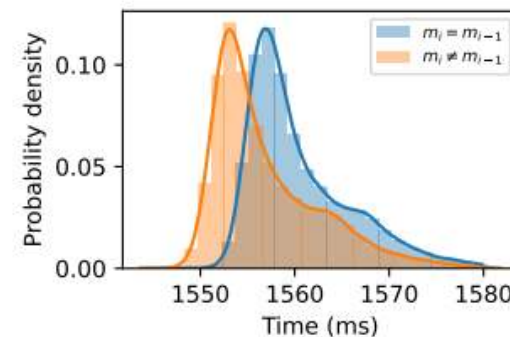
→ x86 CPU에서 상수 시간 구현 SIKE에 적용하여 실제 비밀 키 분석 가능!

■ 분석 방법 및 결과 [10]

- ❖ 선택 암호문 공격으로 비밀키 비트별 복구
- ❖ 비밀키 비트가 $m_i \neq m_{i-1}$ 일때는 중간값 0을 연속적으로, $m_i = m_{i-1}$ 이면 랜덤 중간값을 계산하도록 암호문 선택
- ❖ $m_i \neq m_{i-1} \rightarrow 0$ 이 연속적 \rightarrow HW $\downarrow \rightarrow$ Frequency $\uparrow \rightarrow$ Time \downarrow
- ❖ $m_i = m_{i-1} \rightarrow 0$ 이 불연속적 \rightarrow HW $\uparrow \rightarrow$ Frequency $\downarrow \rightarrow$ Time \uparrow



(a) CIRCL histogram



(b) PQCrypto-SIDH histogram

i7-9700에서 비밀키 비트에 따른 SIKE 연산 시간 차이

참고 문헌

- [1] Prasanna, R., Shivam, B., Sujoy, S. R., & Anupam, C., “Drop by Drop you break the rock - Exploiting generic vulnerabilities in Lattice-based PKE/KEMs using EM-based Physical Attacks”, ePrint, 2020.
- [2] Sim, B. Y., Kwon, J., Lee, J., Kim, I. J., Lee, T. H., Han, J., ... & Han, D. G., “Single-trace attacks on message encoding in lattice-based KEMs”, IEEE Access, vol. 8, pp.183175-183191, 2020.
- [3] Han, J., Lee, T., Kwon, J., Lee, J., Kim, I. J., Cho, J., ... & Sim, B. Y., “Single-Trace Attack on NIST Round 3 Candidate Dilithium Using Machine Learning-Based Profiling”, IEEE Access, vol. 9, pp.166283-166292, 2021.
- [4] Sim, B. Y., Park, A. S., & Han, D. G., “Chosen-Ciphertext Clustering Attack on CRYSTALS-KYBER using the Side-Channel Leakage of Barrett Reduction”, IEEE Internet of Things Journal, 2022.
- [5] Kim, S.; Hong, S. “Single Trace Analysis on Constant Time CDT Sampler and Its Countermeasure”. Appl. Sci. 2018, 8, 1809. <https://doi.org/10.3390/app8101809>
- [6] Guo, Q., Johansson, T., & Nilsson, A., “A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM”, CRYPTO 2020, pp. 359-386, 2020.
- [7] 이태호, ”NIST PQC Round 3 격자 기반 PKE/KEM에 대한 암호문 비교 연산 취약성 연구 “, 석사학위 논문, 국민대학교, 2021. 02.
- [8] Keita, X., Akira, I., Rei, U., Junko, T., & Naofumi, H., ”Fault-injection attacks against NIST’s post-quantum cryptography round 3KEM candidates”. IACR ePrint archive: Report 2021/840, 2021
- [9] Park, A. S., Shim, K. A., Koo, N.H., & Han, D. G., “ Side-Channel Attacks on Post-Quantum Signature Schemes based on Multivariate Quadratic Equations”, TCHES 2018.
- [10] Wang, Yingchen, et al. “Hertzbleed: Turning Power {Side-Channel} Attacks Into Remote Timing Attacks on x86”, 31st USENIX Security Symposium (USENIX Security 22). 2022.