

부채널 분석 및 대응 핵심기술

고려대학교 인공지능사이버보안학과

김희석

80khs@korea.ac.kr



Contents



1. 부채널 분석
2. 부채널 대응 핵심기술
3. PKC 부채널 분석 및 대응기술
4. PQC 부채널 분석 및 대응기술
5. 결론

부채널 분석



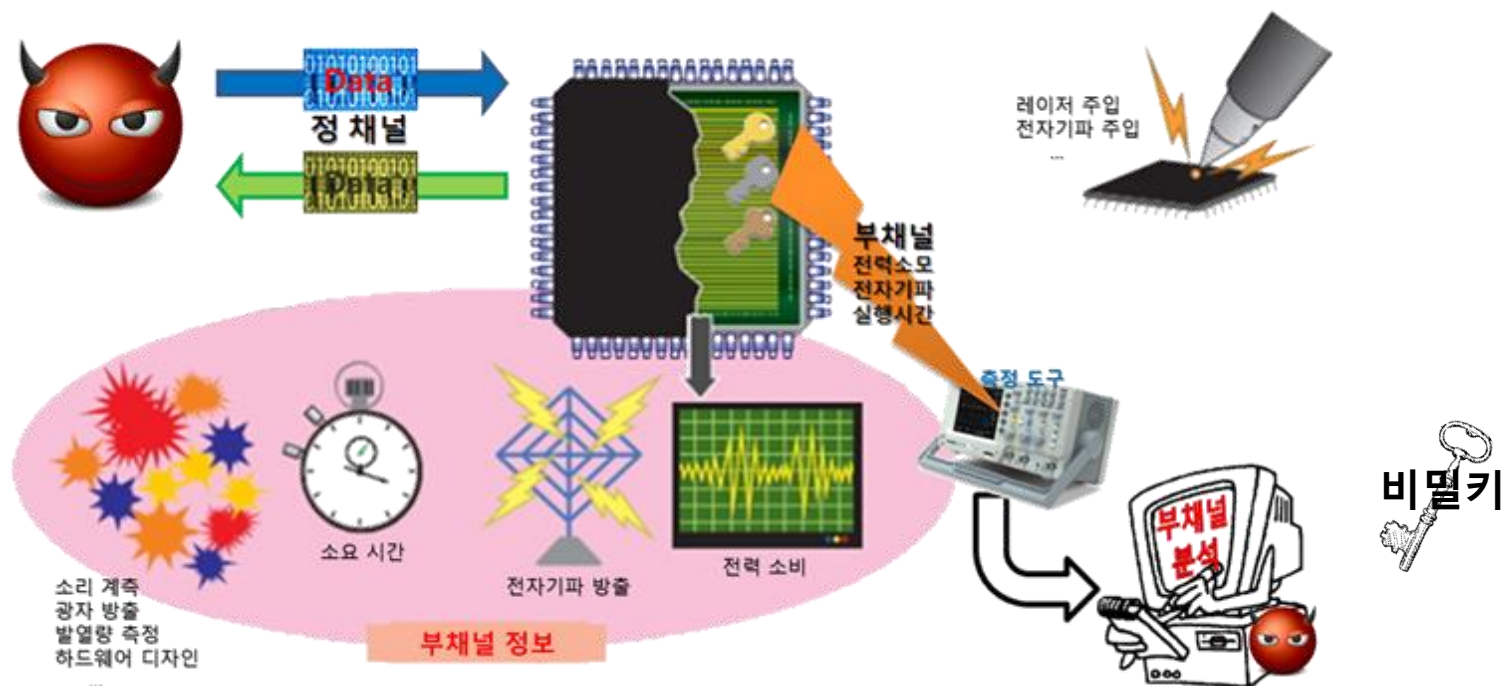
부채널 분석이란?

■ 부채널 분석으로 대표되는 암호시스템의 물리적 보안에 대한 이슈 증가

- 사물인터넷(Internet of Things) 환경에 발달 등으로 암호시스템의 수학적 안전성과 더불어 부채널 분석과 같은 물리적 공격에 대한 안전성이 필요

■ 부채널 분석이란?

- 암호 시스템의 물리적인 구현 및 동작 과정에서 발생하는 전력소모 정보 등의 부채널 정보를 기반으로 비밀 정보를 분석하는 공격 기법



부채널 분석의 위협 (1/2)



부채널 분석 툴



하이패스 (국내)



무선키보드 (국내)



Researchers crack KeeLoq RFID technology - Again
by Steve Ragan - Apr 3 2008, 19:39

자동차 스마트 키 해킹



유럽 교통카드 해킹

Mifare DESFire MF3ICD40: Death by Side-Channels

Posted by Marcin Wojcik

Definitively the best and the most practical attack on the Mifare DESFire MF3ICD40: Power Analysis and Template Matching. The work has been described in the paper by David Oswald and Christof Paar both from Ruhr-University Bochum.

So what has really happened? Authors of the attack have extracted the bits key from a card. This allows to clone the card very easily.

So why is it so important? Firstly, this is the first practical attack on a device widely deployed in the world (the authors) the first available in an open source world device. Mifare DESFire MF3ICD40 is used in many systems (e.g., by Czech railways (in-ka) and in many other systems) to perform a non-invasive power attack. David and Christof used low cost equipment.



FPGA 해킹



보안 USB 해킹



Ricoh HACK!

How to reprogram toner cartridge chip
Arduino 24c02 EEPROM I2C programming



토너 정품 인증칩 해킹

부채널 분석의 위협 (2/2)

사물인터넷 부채널 공격 위험에 놓였다..25만원짜리 암호 해독 장비 등장

진수희 "IC카드 복제 가능...금융사고 위험 노출"

뉴스스

입력 2008.10.20 14:47



질의하는 한나라당 진수희 의원

복제가 불가능한 것으로 알려진 금융 IC카드가 금융사고 부채널 공격에 취약해 교체 주장이 제기됐다.

국회 기획재정위원회 소속 진수희 의원(한나라당, 서울 성동 갑)은 20일 한국은행에서 "복제가 불가능해 안전하다고 알려진 IC카드가 여전히 복제가 가능하며 복제 가능한 교체가 없는 한 복제로 인한 금융사고의 위험에 노출돼 있다"고 밝혔다.

이날 진 의원에 따르면 MS(Magstripe) 방식의 IC카드가 금융사고가 발생 사고를 막기 위해 지난 1996년

하지만 일반적인 IC칩은 동작 시 획득할 수 있는 부채널 공격에

"컴퓨터에 손만 갖다대도 데이터 빼간다"

공유 330 댓글 0

언어 선택 Google 번역에서 제공

내 정보를 철저히 지키고 싶으면 아무도 내 컴퓨터에 손 댈 수 없도록 해야 할지도 모르겠다. 컴퓨터에 손만 대도 컴퓨터가 처리 중인 데이터를 빼돌릴 수 있다는 연구결과가 나왔다고 'MIT테크놀로지리뷰'가 8월20일(현지시간) 보도했다.

앨버트브대학 소속 컴퓨터 보안 전문가 에란 트로머 등 연구진 3명은 컴퓨터에 손만 대도 데이터를 추출할 수 있는지 실험했다. 이들은 컴퓨터가 작동할 때 전위값이 변화한다는 데서 아이디어를 얻었다.

연구진은 금속으로 된 컴퓨터 표면에 전극을 붙이고 컴퓨터가 수행하는 작업을 확인했다. 컴퓨터에 특정 작업을 시키면 중앙처리장치(CPU)의 전위값이 변화한다는 데서 아이디어를 얻었다. 연구진은 컴퓨터의 전위값 변동을 측정해 CPU 작동 패턴을 외부에

삼성SDS, 전자서명 부채널 공격 차단 암호기술 세계 최초 확보

좋아요 0개 | 입력: 2018-10-02 08:48

#정보보호 #정보보안 #IT보안 #사이버보안 #전자서명

[보안뉴스 박미영 기자] 삼성SDS는 세계 최초로 전자서명에 대한 부채널 공격을 차단하는 해킹 방지 암호기술을

아무 해커나 쉽게 따라할 수 있는 난이도의 부채널 공격 등장!

좋아요 60개 | 입력: 2019-01-11 21:08

#정보보호 #정보보안 #IT보안 #사이버보안 #부채널 #멜트다운 #스펙터 #시스템콜

그 어떤 정보도 보호될 수 없게 하는 공격...난이도가 낮기까지 시스템 콜 남용하는 방법...OS와 앱 제조사들의 개별적 패치로 해결

[보안뉴스 문가용 기자] 특정 칩을 공격할 수 있게 해주는 새로운 부채널 공격 기법이 발견됐다. 현대 운영 시스템의 근본적인 기능을 악용하는 방법으로, 완벽하게 감춰져 있다고 여겨지는 데이터에 접근할 수 있게 해준다.

한끼 식사로 암호화 키를 훔치는 가장 완벽한 방법

Jeremy Kirk | IDG News Service

이스라엘 연구원들이 피타 브레드(Pita Bread)를 이용해서 컴퓨터에 저장된 암호화 키를 훔쳐낼 수 있는 빠르고 저렴한 방법을 고안했다고 밝혔다.

이들이 발표한 최신 논문에서는 컴퓨터가 계산을 수행하는 과정에서 파장되어 나오는 전기적

새로운 컴퓨터 프로세서 부채널 공격! RSA 알고리즘이 위험

좋아요 32개 | 입력: 2018-08-20 10:39

#정보보호 #정보보안 #IT보안 #사이버보안 #CPU #ARM #프로세서 #자기장 #RSA

CPU가 아닌 GPU에서도 부채널 공격 가능하다

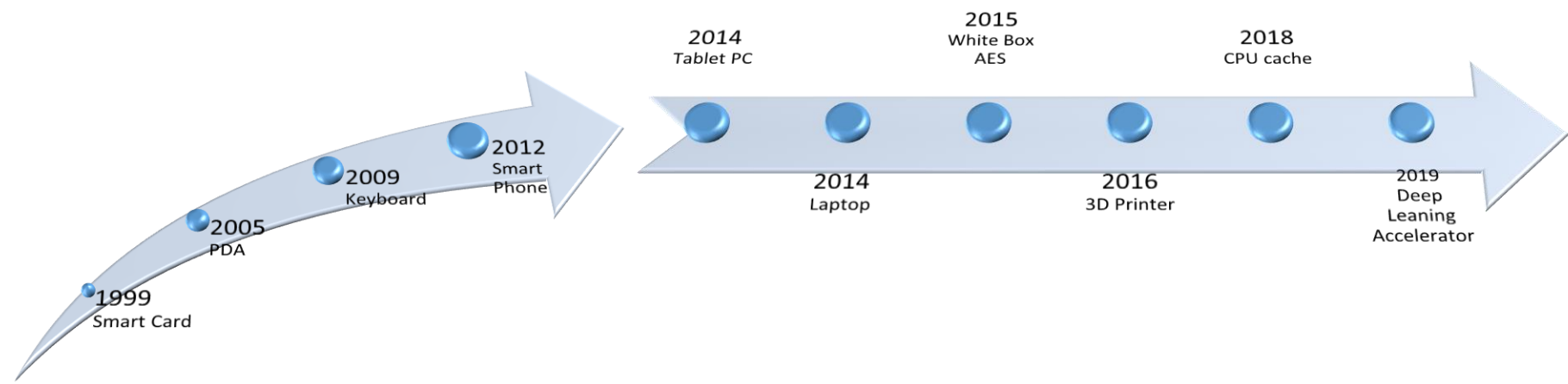
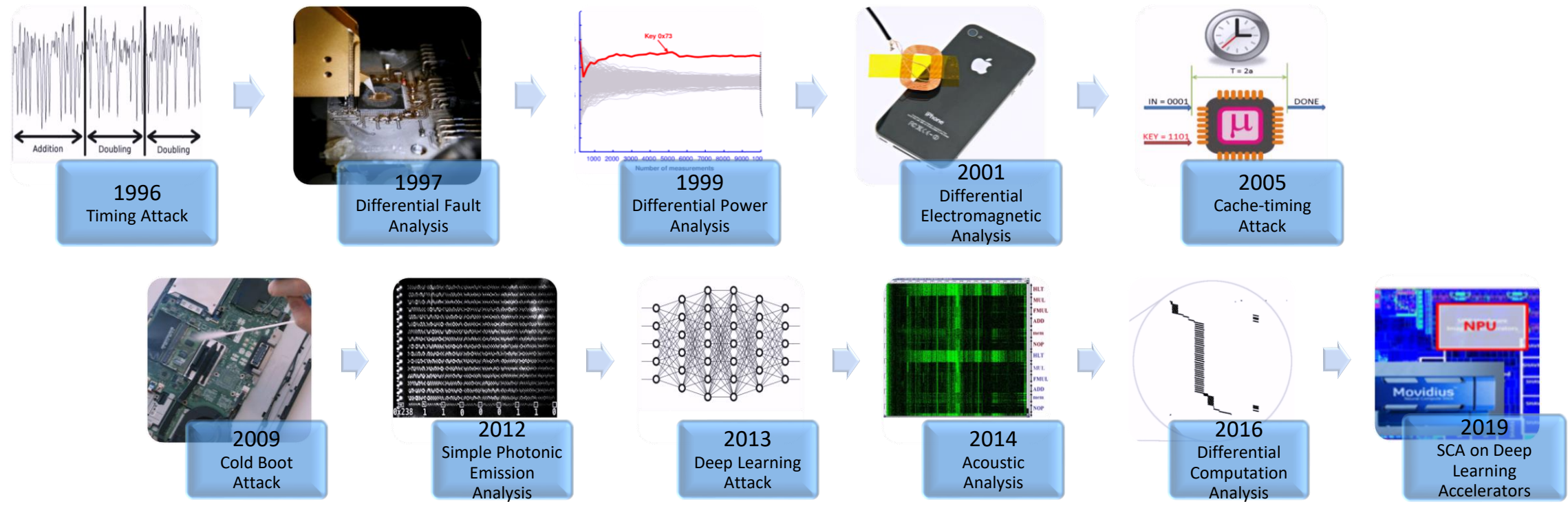
좋아요 19개 | 입력: 2018-11-08 13:57

#정보보호 #정보보안 #IT보안 #사이버보안 #GPU #부채널 #공격

GPU 부채널 공격은 CPU 부채널 공격보다 실현 가능성 높아 세 가지 공격 시나리오...민감 정보 유출 혹은 내부 기술 유출

[보안뉴스 문가용 기자] CPU에서 부채널 공격을 가능케 하는 취약점들이 꾸준히 나오더니, 이번에는 GPU에

부채널 분석 기법 및 분석 대상 연대기



일반적인 부채널 분석 대응 고려 순서

- 시간차 분석 (상수시간 구현)
- 전력/전자파 분석 – 논프로파일링 공격 (마스킹, 블라인딩 구현) ... 어느 수준까지 대응 필요?
- 캐시 타이밍 공격 (상수시간 구현, 록업테이블 변경)
- 전력/전자파 분석 – 프로파일링 공격 (대응이 어려움)

부채널 분석 예시 – 소요 시간 공격

■ 소요 시간 공격 (Timing Attack)

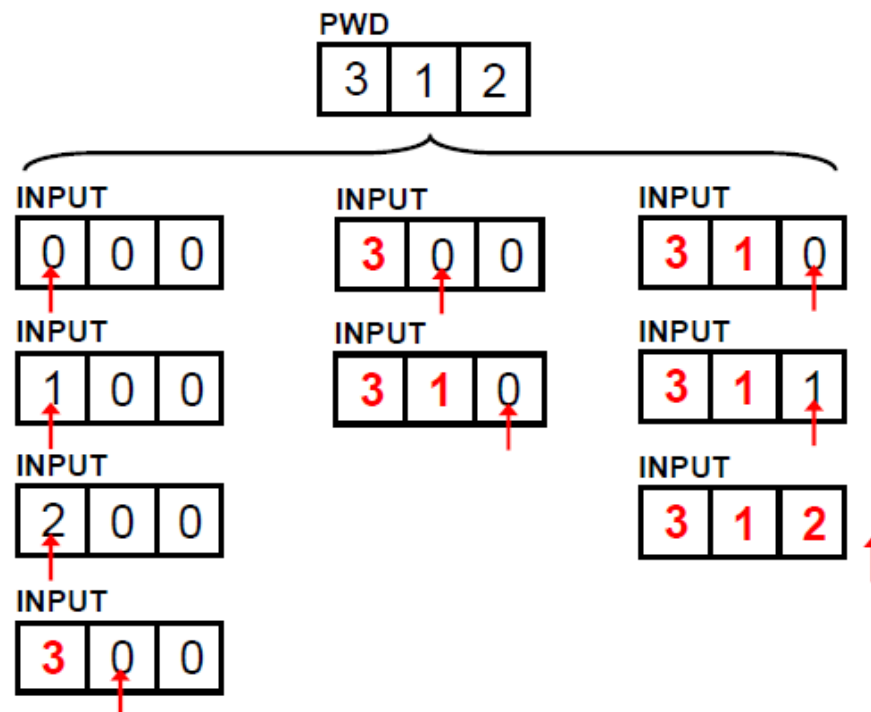
- 데이터에 따라 계산에 소요되는 시간이 다를을 이용하여 비밀정보를 복원하는 공격 기법
- 부채널 분석에 대한 고려가 없이 코딩을 하는 경우, 심각한 취약점 발생

```
for i = 0 to 2
  if (INPUT[i] ≠ PWD[i])
    return("REJECT")

return("ACCEPT")
```



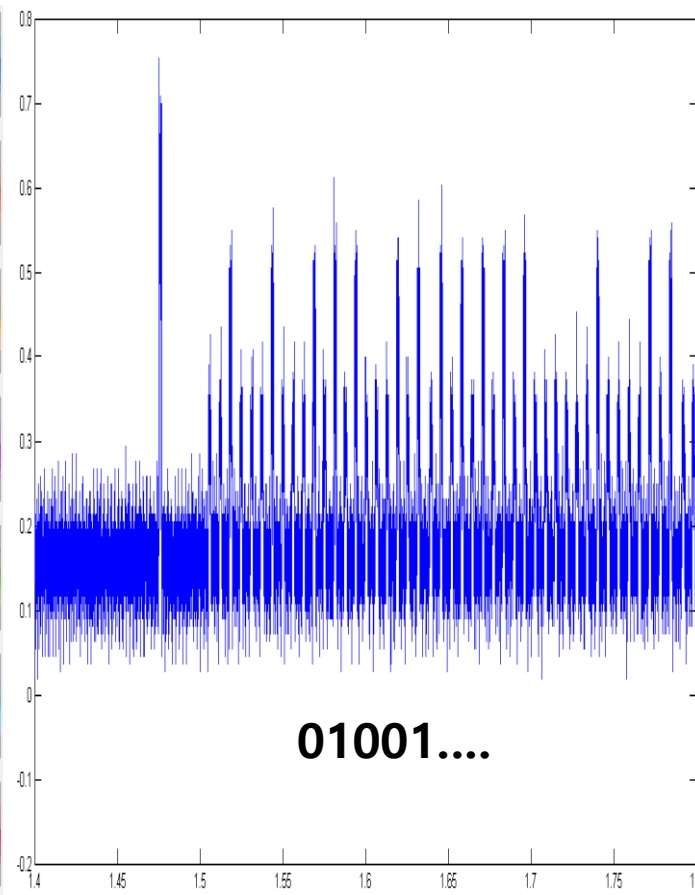
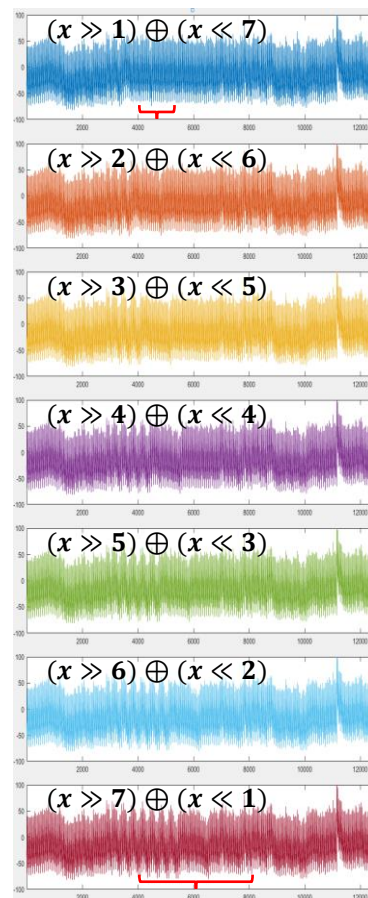
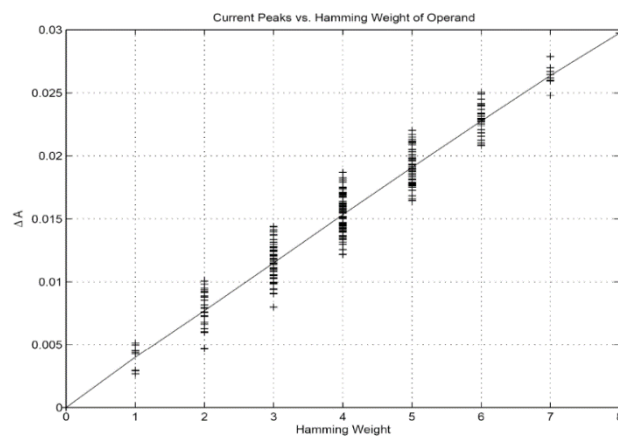
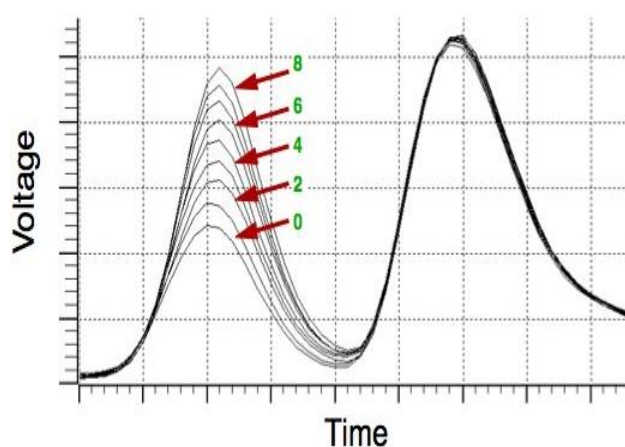
PIN verification step



부채널 분석 예시 – 전력/전자파 분석 공격

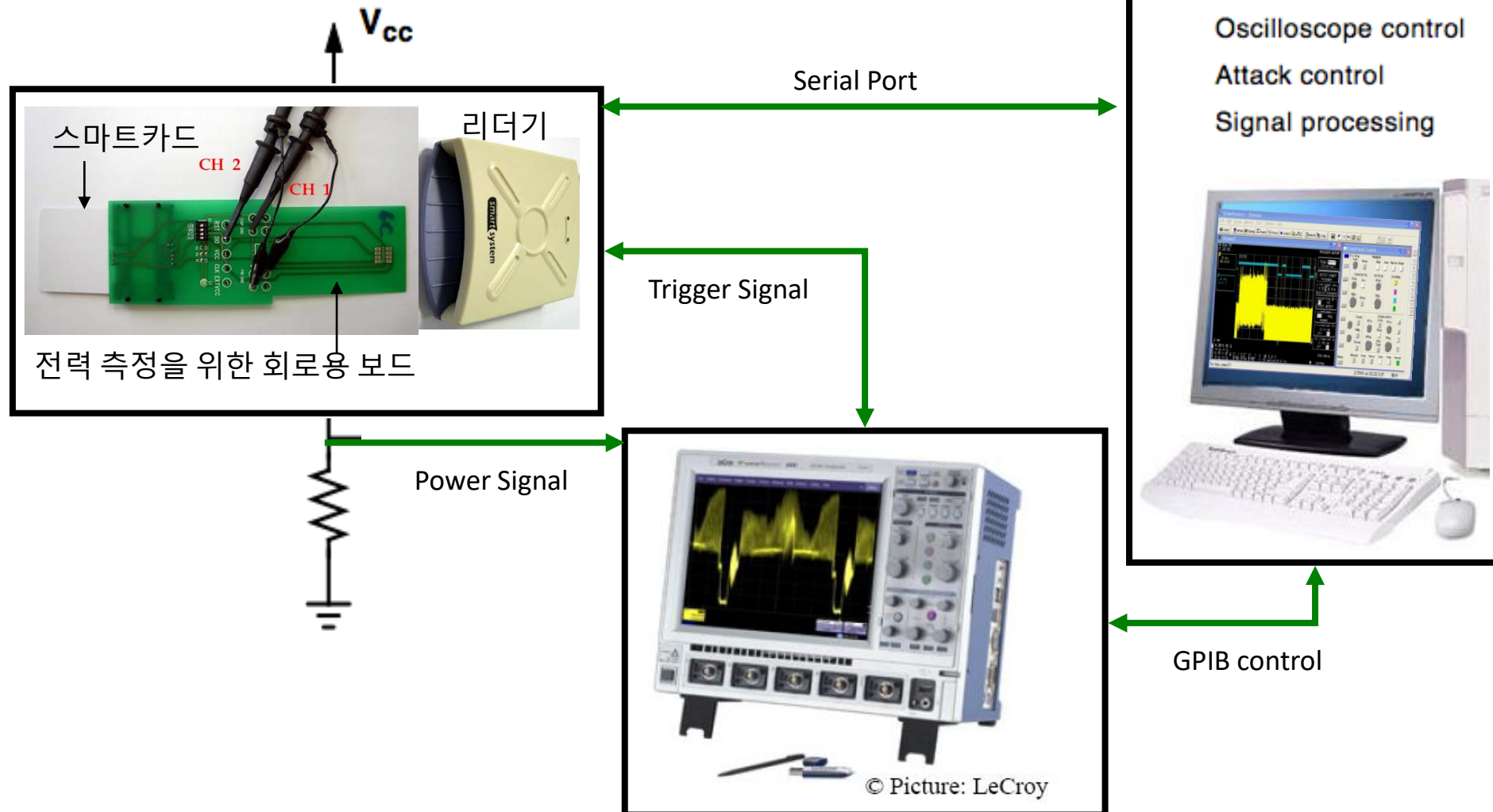
■ 전력/전자파 분석 공격은 왜 가능한가?

- 장비 내부에서 연산되는 데이터 값, 연산코드 값에 의존해 전력/전자파 소비
- 일반적으로 데이터의 해밍웨이트에 의존하여 전력/전자파를 소비



부채널 분석 예시 – 전력/전자파 분석 공격

■ 실험 환경 전체 다이어그램



부채널 분석 예시 – 전력/전자파 분석 공격

■ 실험 환경 전체 다이어그램

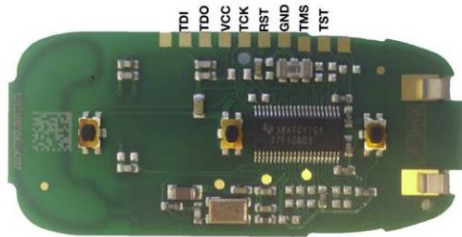
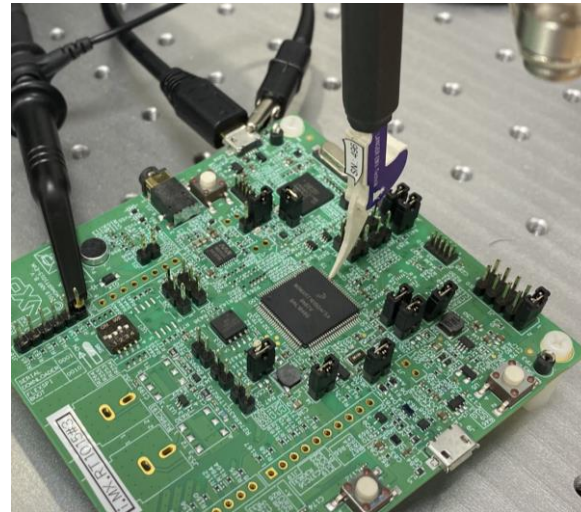
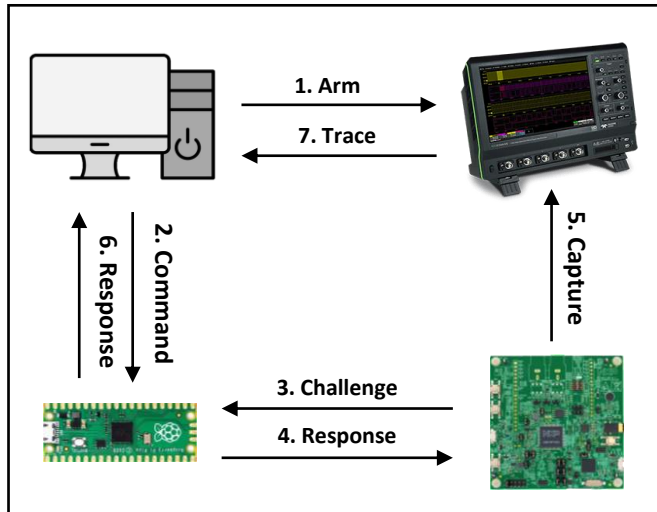
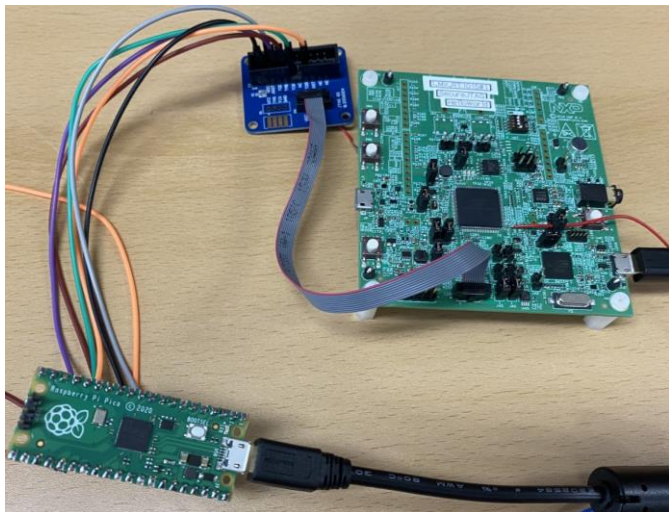


Figure 3: The Tesla Model S key fob PCB with the TMS37128 chip in the middle. The

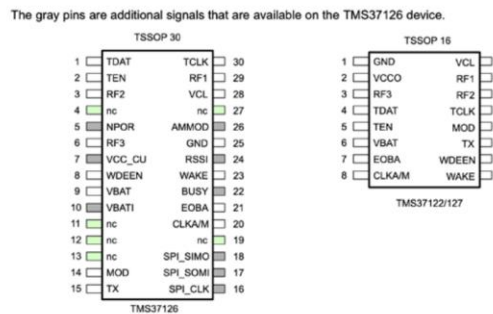


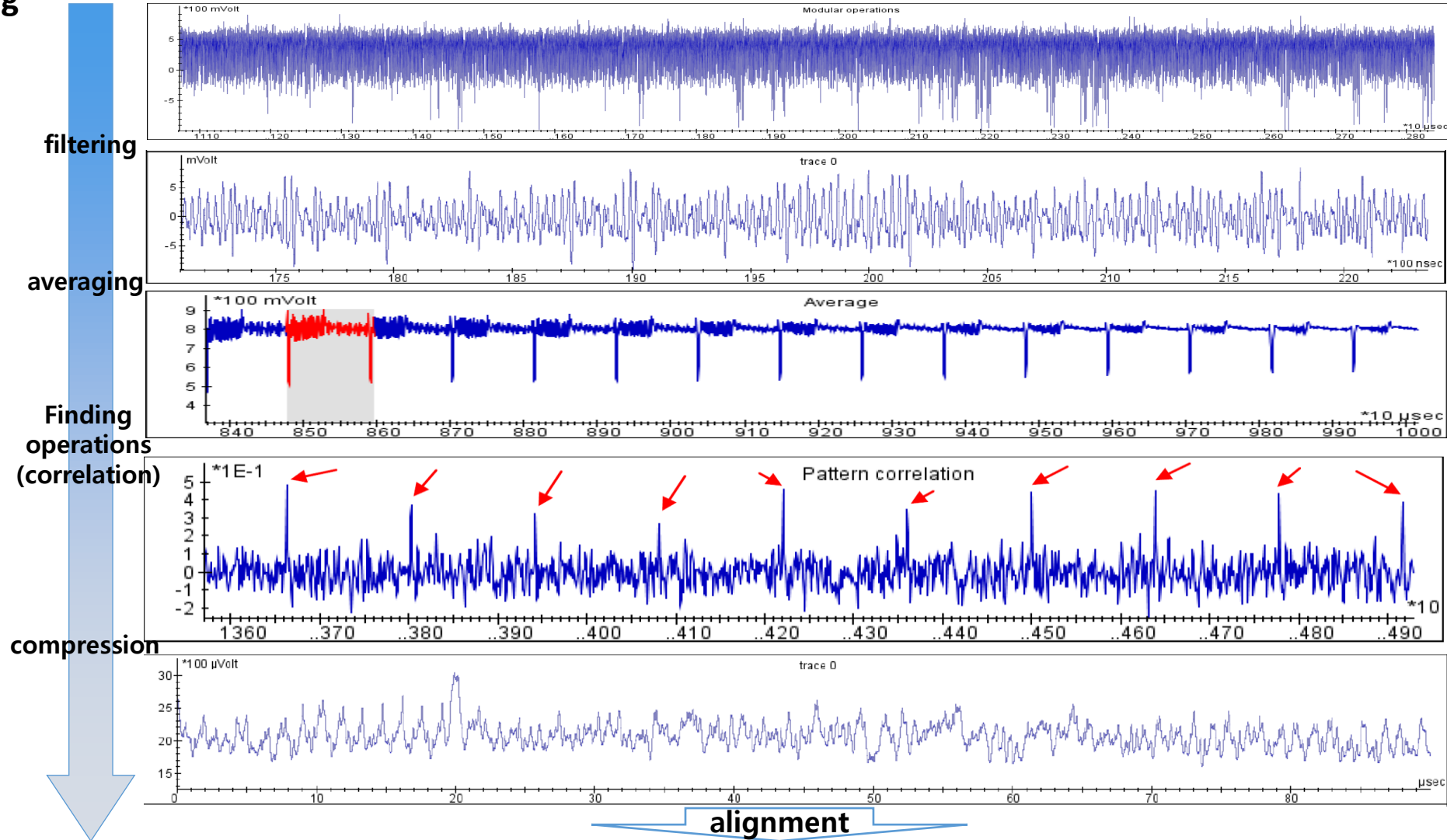
Figure 1. Pin Assignment Comparison



Figure 4: Measurement setup for acquiring power traces from a TMS37126 IC (on the yellow breakout board). Top-right: Tektronix CT-1 current probe. Bottom-left: Arduino Pro Mini functioning as a serial to SPI translator.

부채널 분석 예시 – 전력/전자파 분석 공격

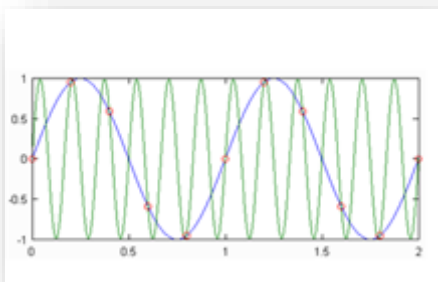
Pre-processing



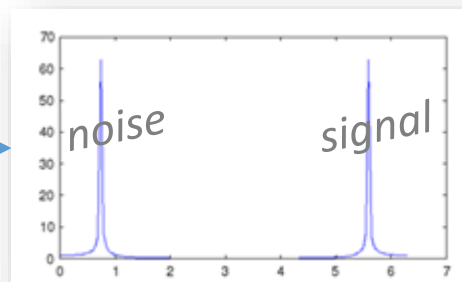
대표적인 부채널 신호 처리 기법 (1/3)

잡음 제거 기법 (Denoising)

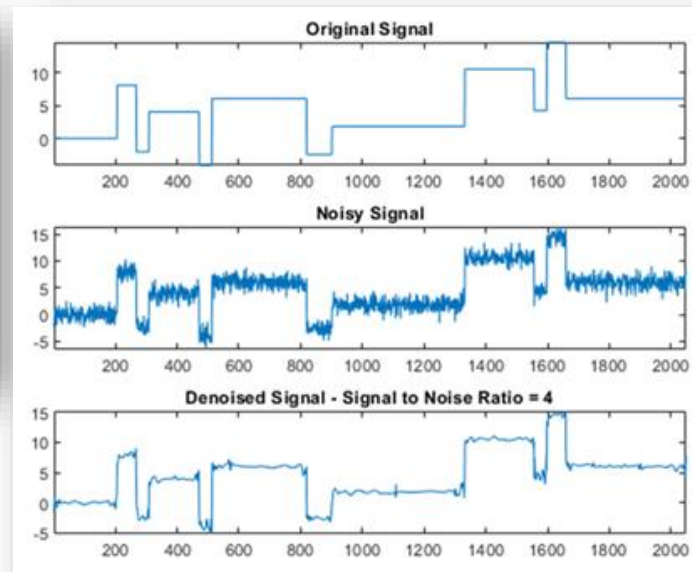
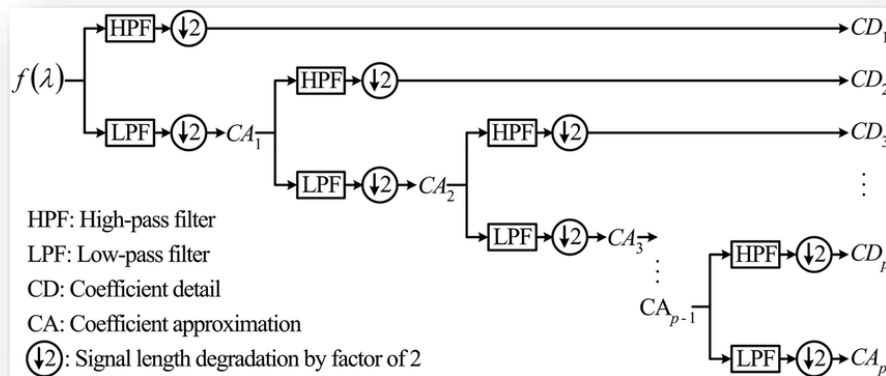
- Discrete Fourier Transform: 신호의 주기성을 계산한 뒤 불필요한 주파수 성분 제거



$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n / N}$$



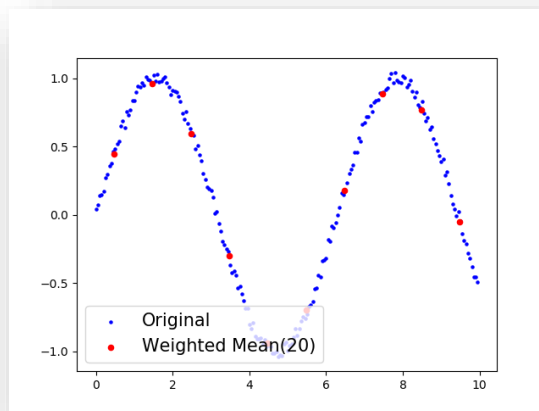
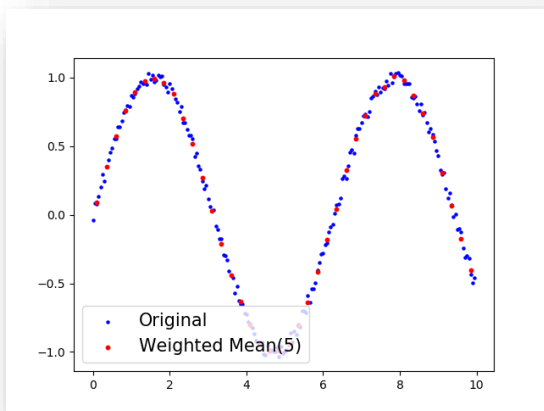
- Dynamic Wavelet Transform: 잡음이 심한 저주파 대역의 신호는 기여도 축소



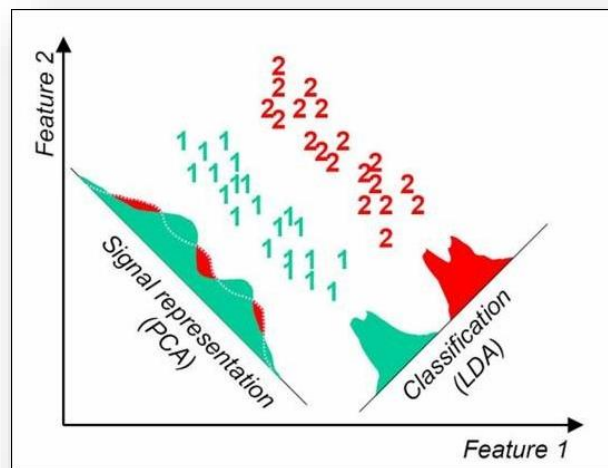
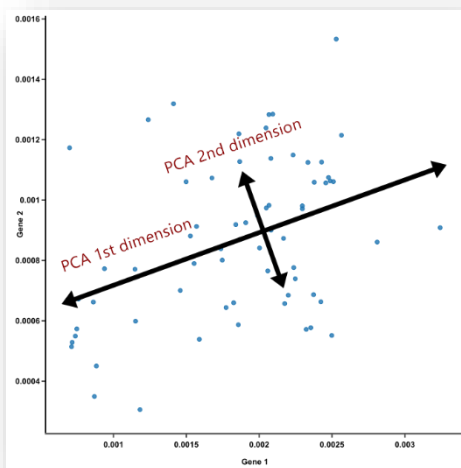
대표적인 부채널 신호 처리 기법 (2/3)

차원 축소 기법 (Resampling)

- Weighted Mean: 특정 구간 값들의 가중 평균을 대푯값으로 사용



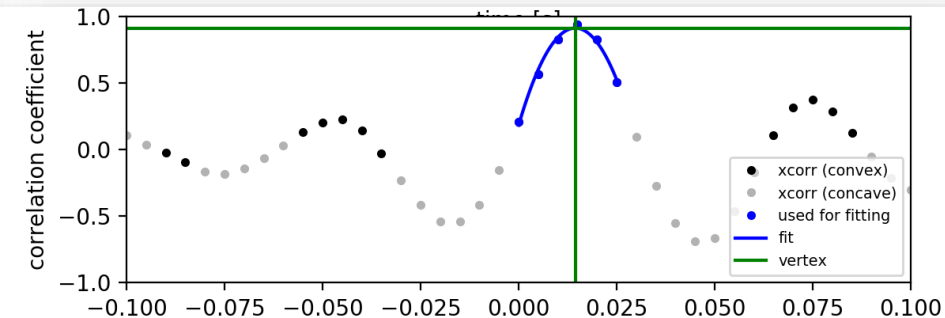
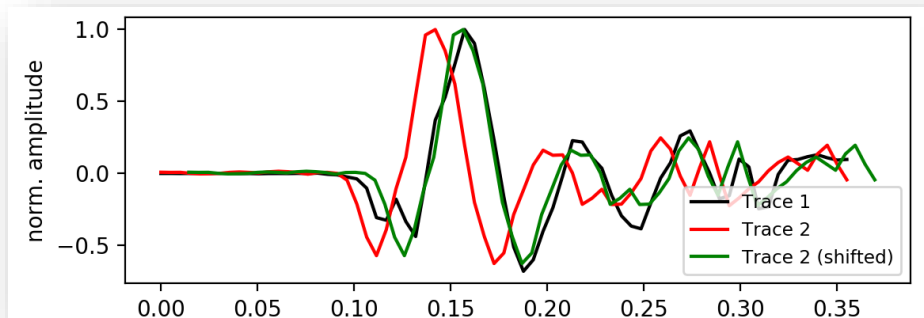
- PCA / LDA: 기여도가 적은 축의 값을 사영시켜 저차원의 데이터로 변환



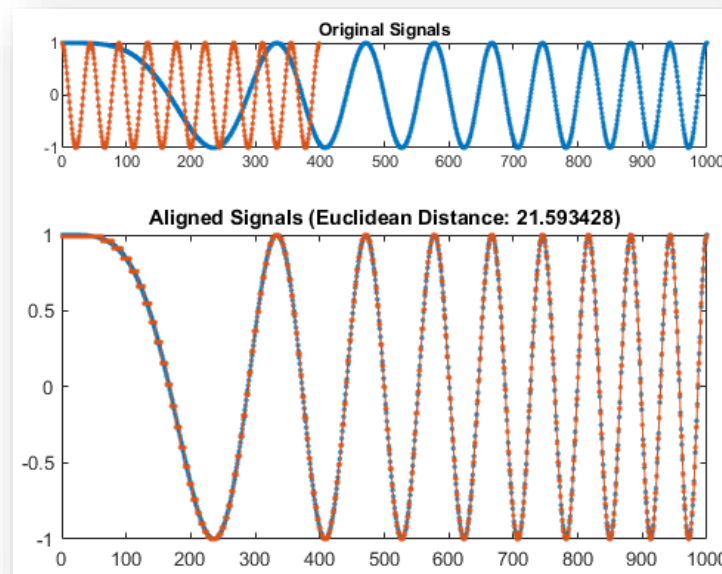
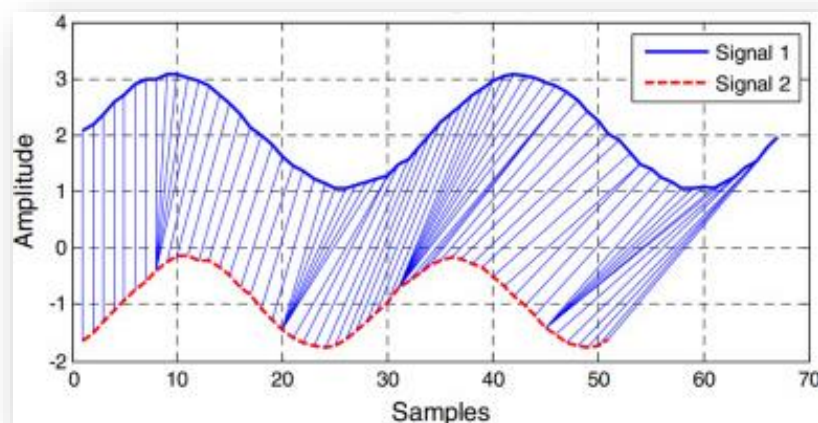
대표적인 부채널 신호 처리 기법 (3/3)

정렬 기법 (Aligning)

- Cross-correlation: 기준 신호와 상관계수가 가장 높도록 목표 신호를 시프트



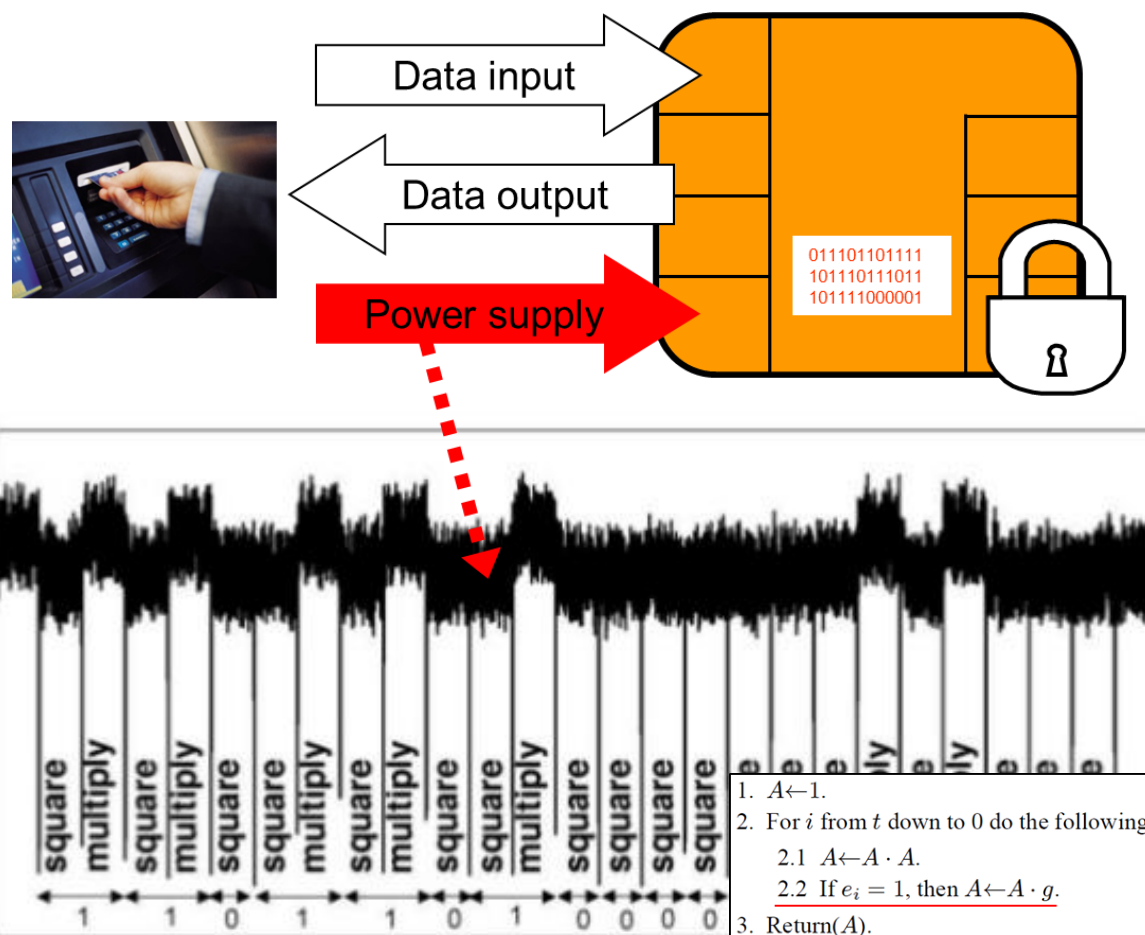
- Dynamic Time Warping: 길이가 다른 두 신호를 일치시키기 위해 시간 축을 늘여 놓음



부채널 분석 예시 – 단순 전력 분석

■ 단순 전력 분석 (Simple Power Analysis)

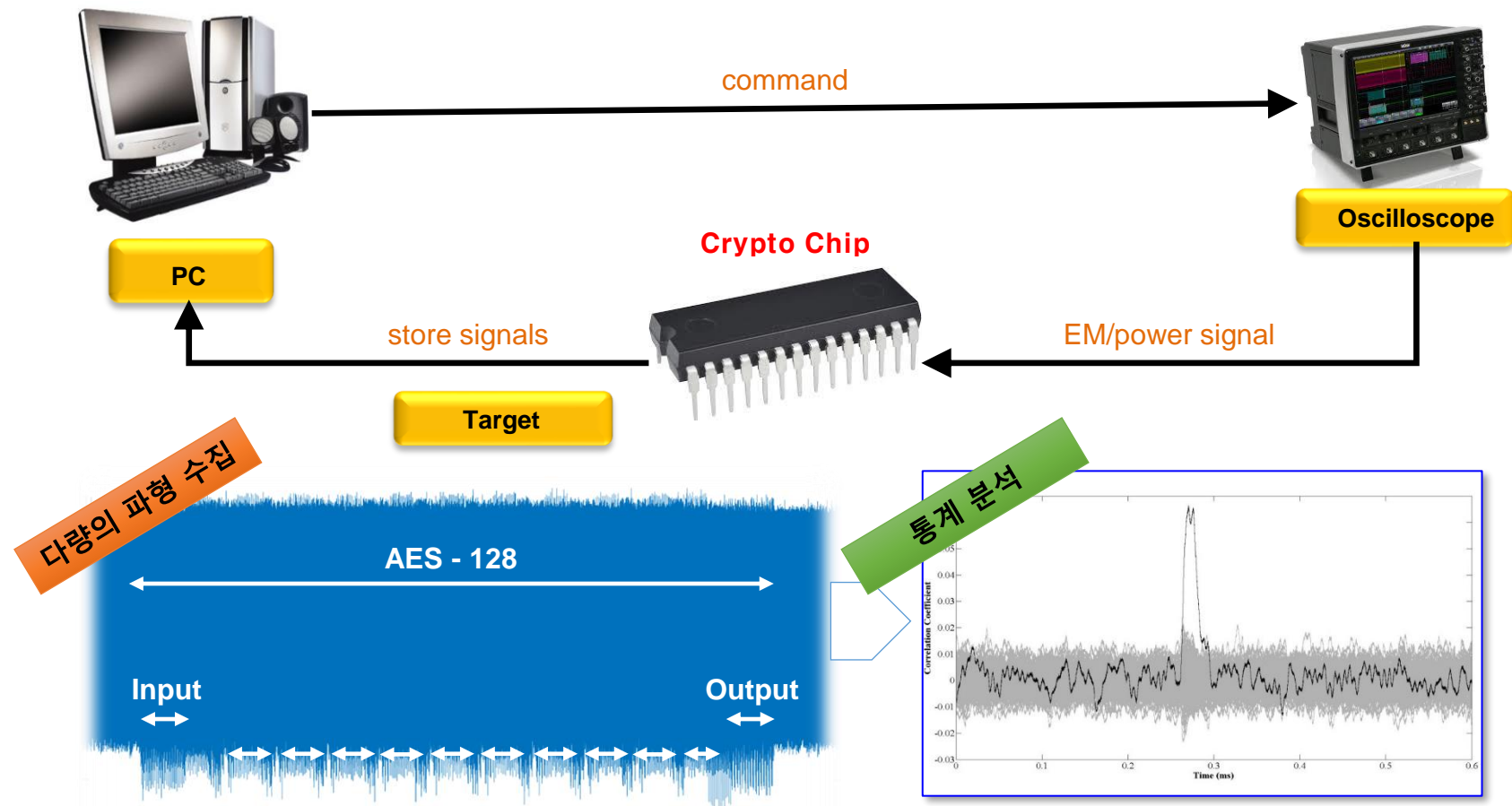
- 암호화 연산 동안의 전력 소모량을 수집하여 각각의 연산 별 파형의 특징을 이용하여 비밀 키를 복원하는 공격 기법



부채널 분석 예시 – 차분 전력 분석

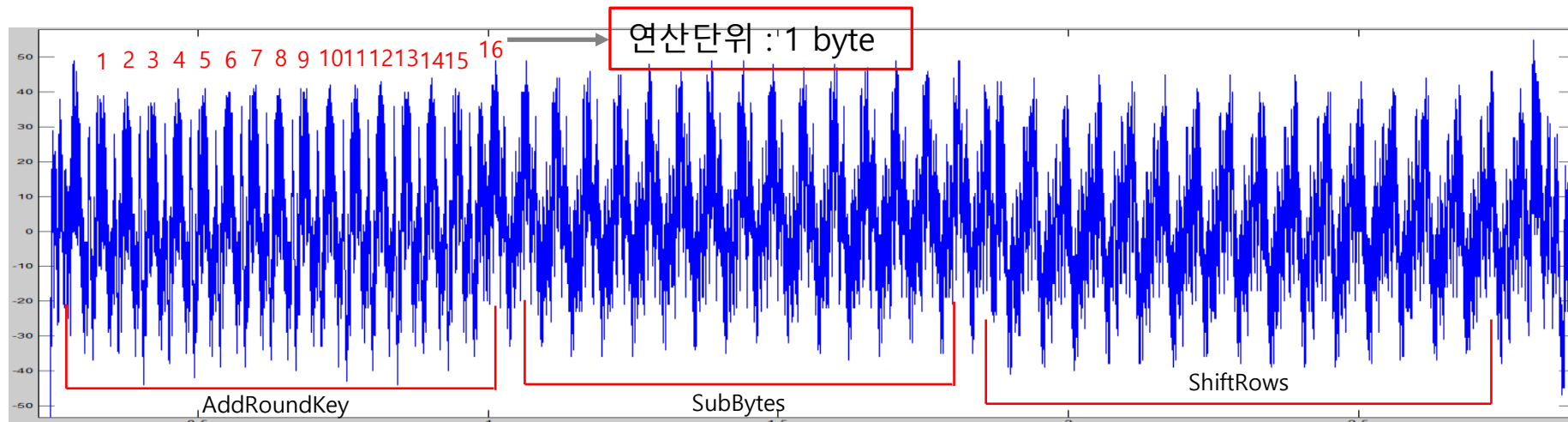
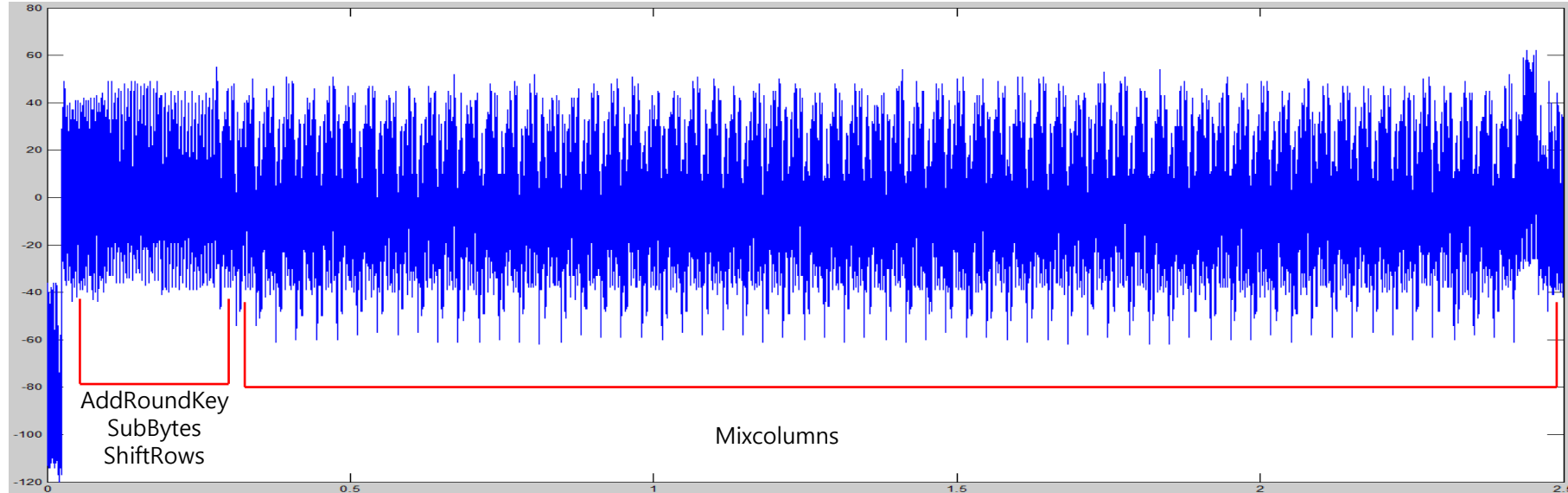
■ 차분 전력 분석 (Differential Power Analysis)

- 반도체로 이루어진 마이크로프로세서는 연산 데이터에 의존하여 전력 소모가 발생함을 이용하여, 통계기법으로 비밀키를 복원하는 공격 기법



부채널 분석 예시 – 차분 전력 분석

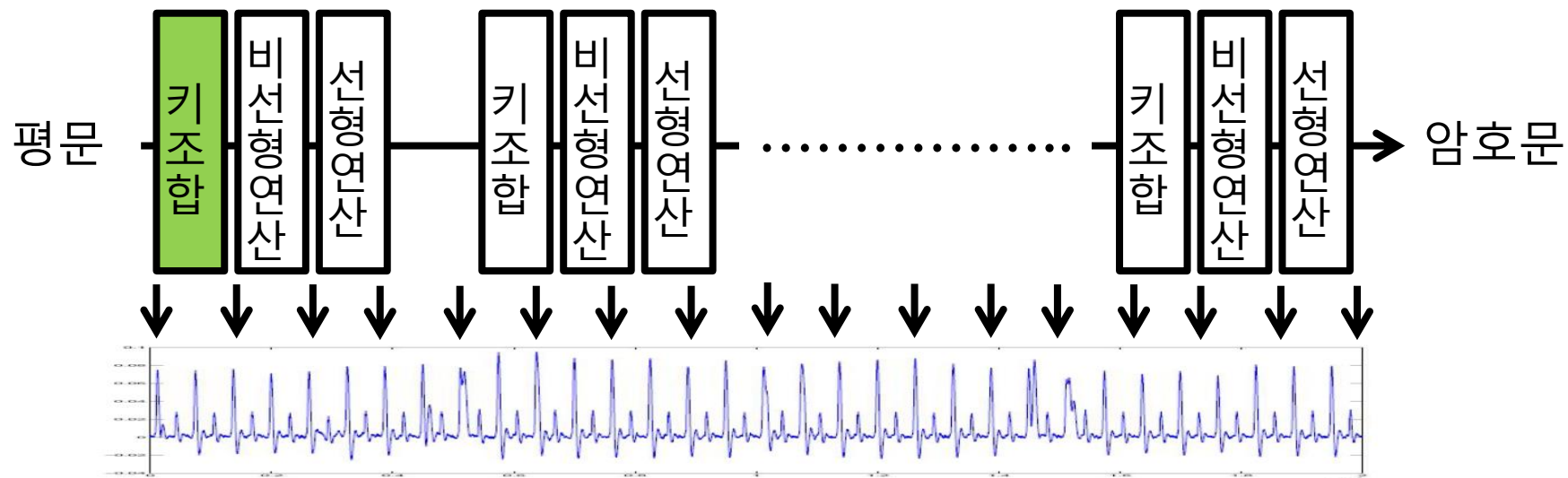
Power trace (AES S/W)



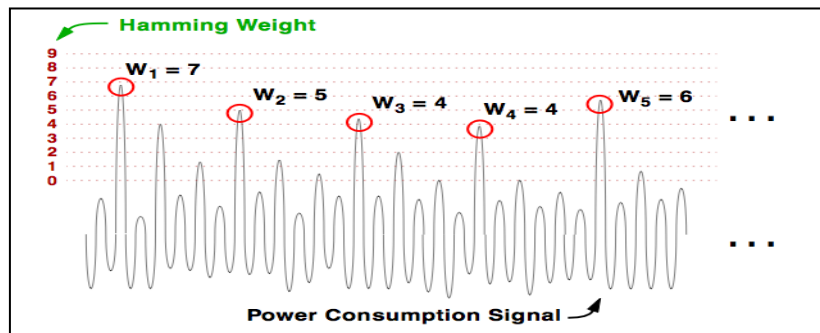
부채널 분석 예시 – 차분 전력 분석



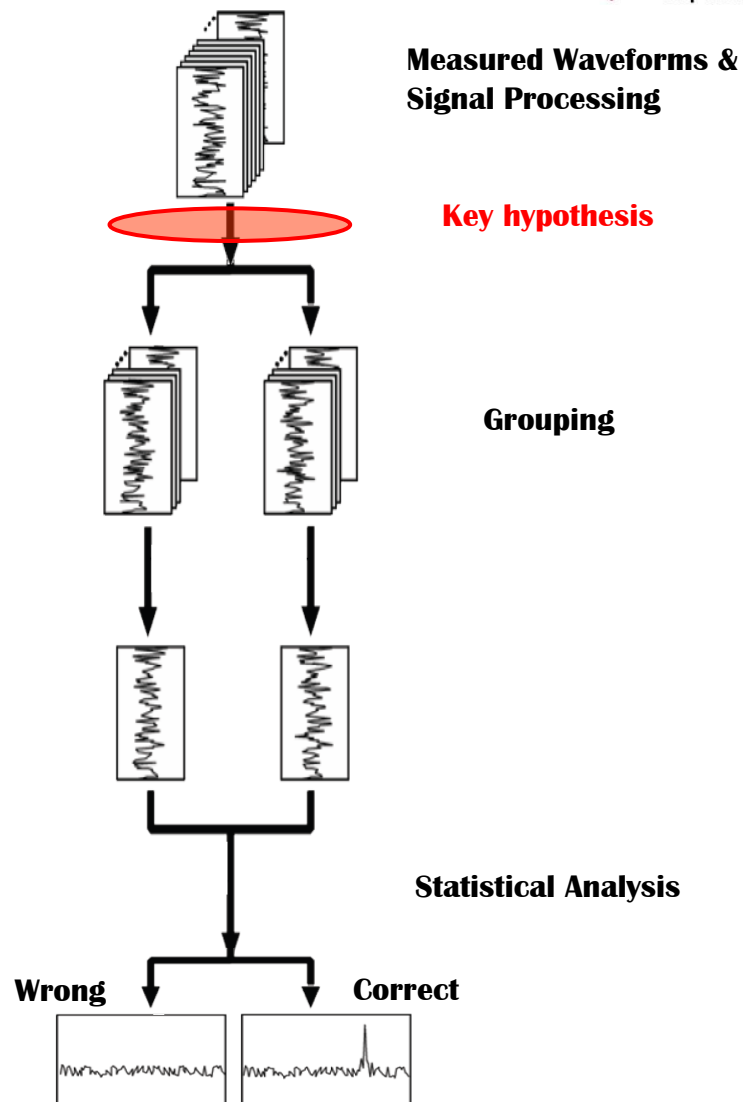
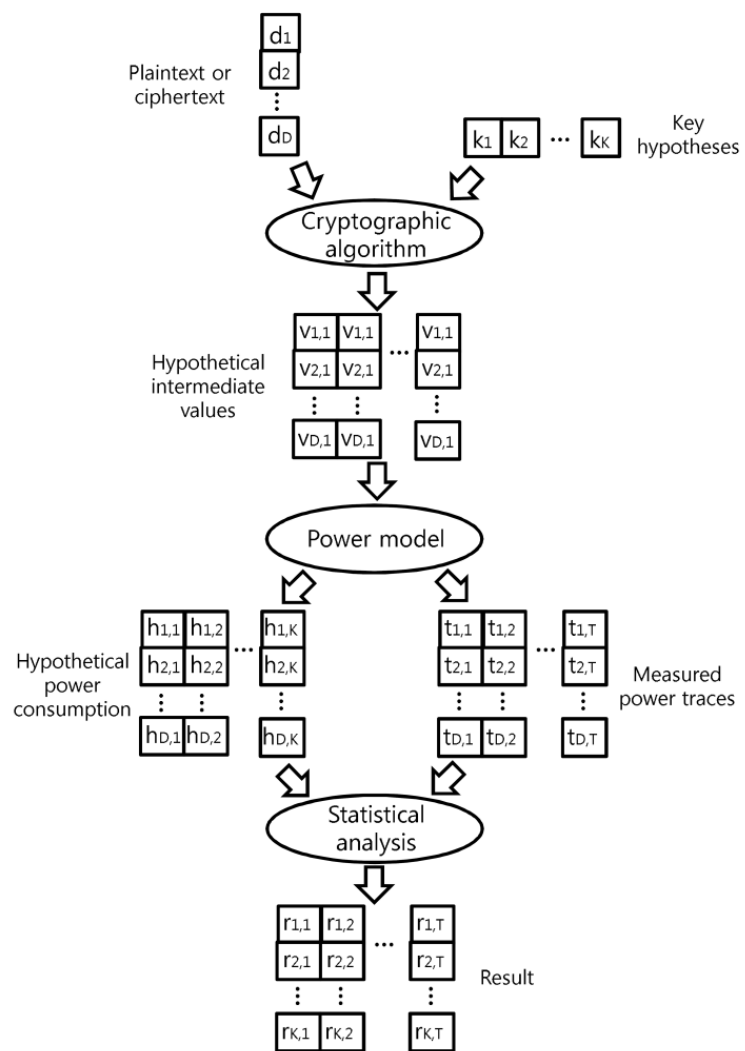
AES, ARIA 등의 대칭키 암호 연산 방식



Hamming
weight



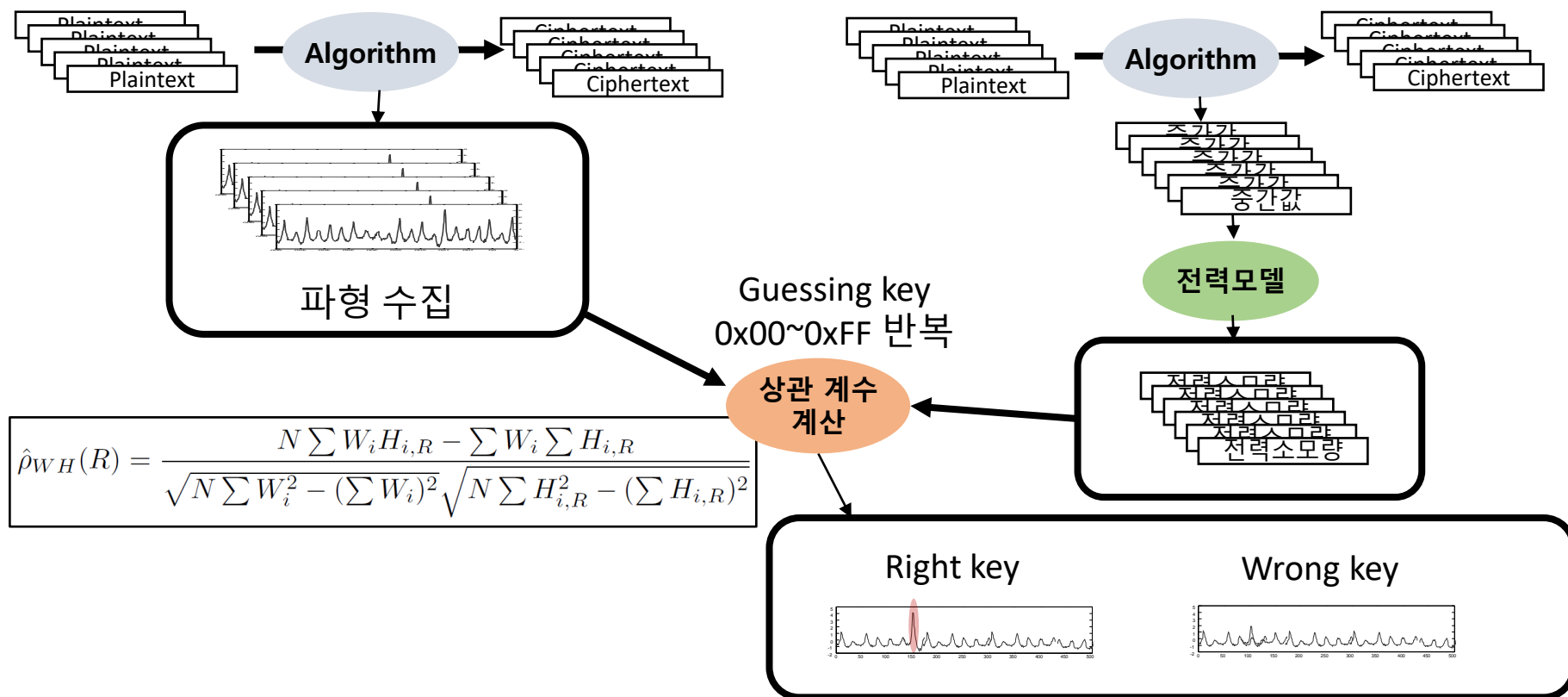
부채널 분석 예시 – 차분 전력 분석



부채널 분석 예시 – 상관계수 전력 분석

▪ DPA / CPA (Differential Power Analysis / Correlation Power Analysis)

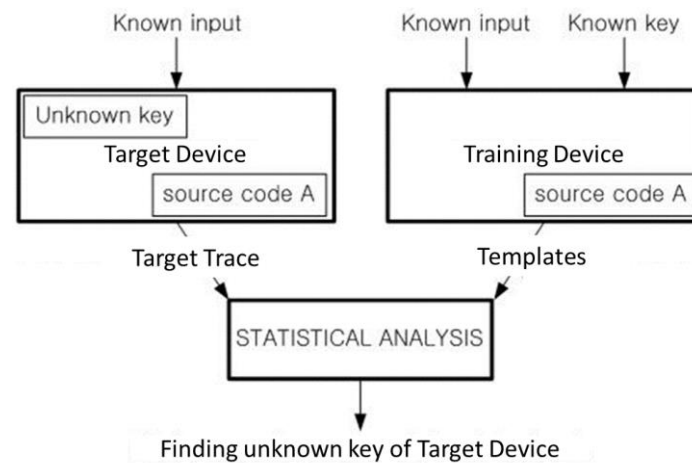
- 공격 대상 알고리즘 : 일반 블록암호
- 공격 가정 : 다수의 임의 평문·암호문 사용가능
- 공격 개요 : 연산 데이터에 의존하여 전력 소모가 발생함을 이용하여 통계적 기법으로 비밀정보를 분석



부채널 분석 예시 – 프로파일링 부채널 분석

▪ Profiling Attack & Non-Profiling Attack

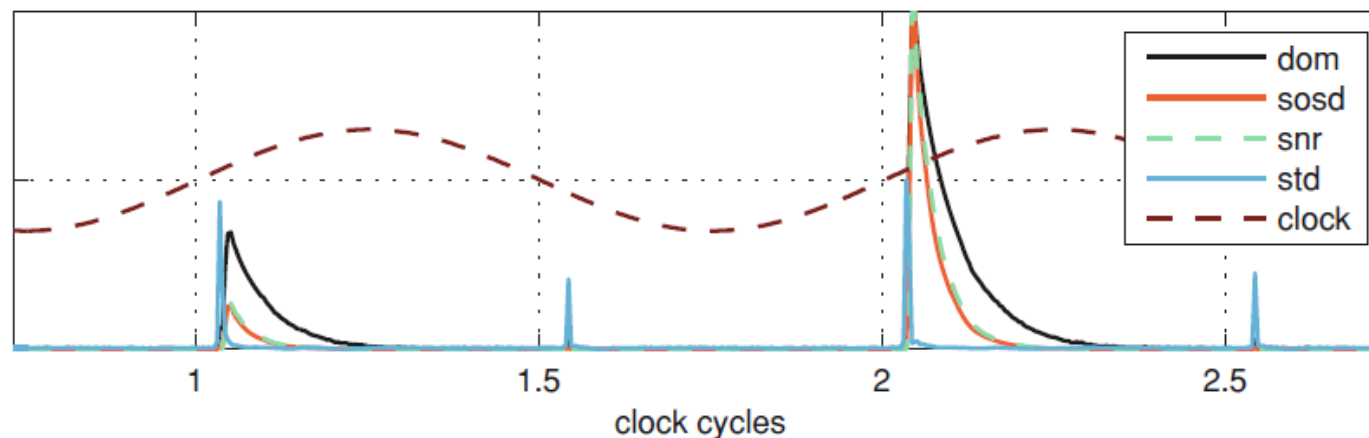
- Non-Profiling Power Analysis : SPA, DPA, ROSETTA, HCCA, HODPA 등등 공격대상 장비로부터 프로파일 생성 없이 공격하는 전력 분석 기법
 - 키를 모르는 장비만 가진 공격자가 키를 찾기 위해 수행하는 대부분의 공격들이 이에 해당
- Profiling Power Analysis : Template Attack, Stochastic Model 등이 존재
 - 공격대상 장비 혹은 이기종 장비로부터 프로파일 생성
 - 프로파일과 공격대상 장비로부터 얻은 타겟 트레이스와의 매칭 확률을 통해 키를 찾음
 - 일반적인 프로파일링 기반 전력분석
 - 공격대상장비(키를 모르고 있는)와 동일사양의 다른장비(키를 알고있는 혹은 변경 가능한)로부터 프로파일 생성
 - 프로파일과 공격대상 장비로부터 얻은 타겟 트레이스의 매칭을 통해 공격 대상 장비의 키를 찾는 템플릿 공격을 지칭



부채널 분석 예시 – 프로파일링 부채널 분석

■ 템플릿 공격 – 템플릿 구성 (Profiling Stage)

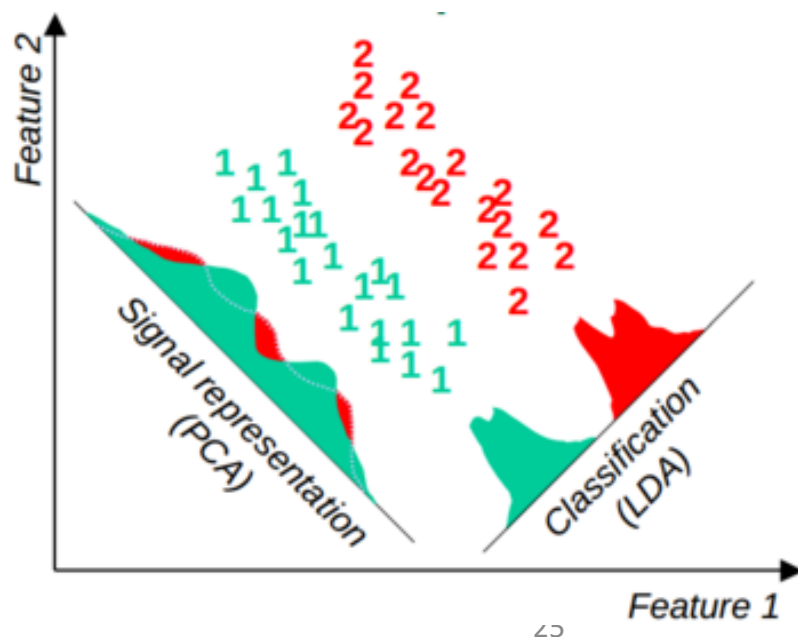
- 템플릿 구성 단계 (Profiling stage)
 - 연산과 데이터에 대한 누출이 발생하는 지점 탐색
 - Feature selection (특징 선택)
 - Feature extraction (특징 추출)
- (1단계) 유의미한 시점(POIs, Points of Interest) 선택 – 부채널 분석 능력 요구
 - SNR / DoM / SOAD / SOSD / SOST / CPA / ...
 - 각 방법으로 POIs 선택시 미묘하게 다를 수 있음



부채널 분석 예시 – 프로파일링 부채널 분석

■ 템플릿 공격 – 템플릿 구성 (Profiling Stage)

- (2단계) 데이터를 대표하는 특징 추출
 - PCA(Principle Component Analysis)
 - 고차원 데이터의 정보를 최대한 유지하도록 차원을 축소시키는 방법
 - LDA(Linear Discriminant Analysis)
 - 고차원 데이터의 클래스가 최대한 구분되도록 차원을 축소시키는 방법



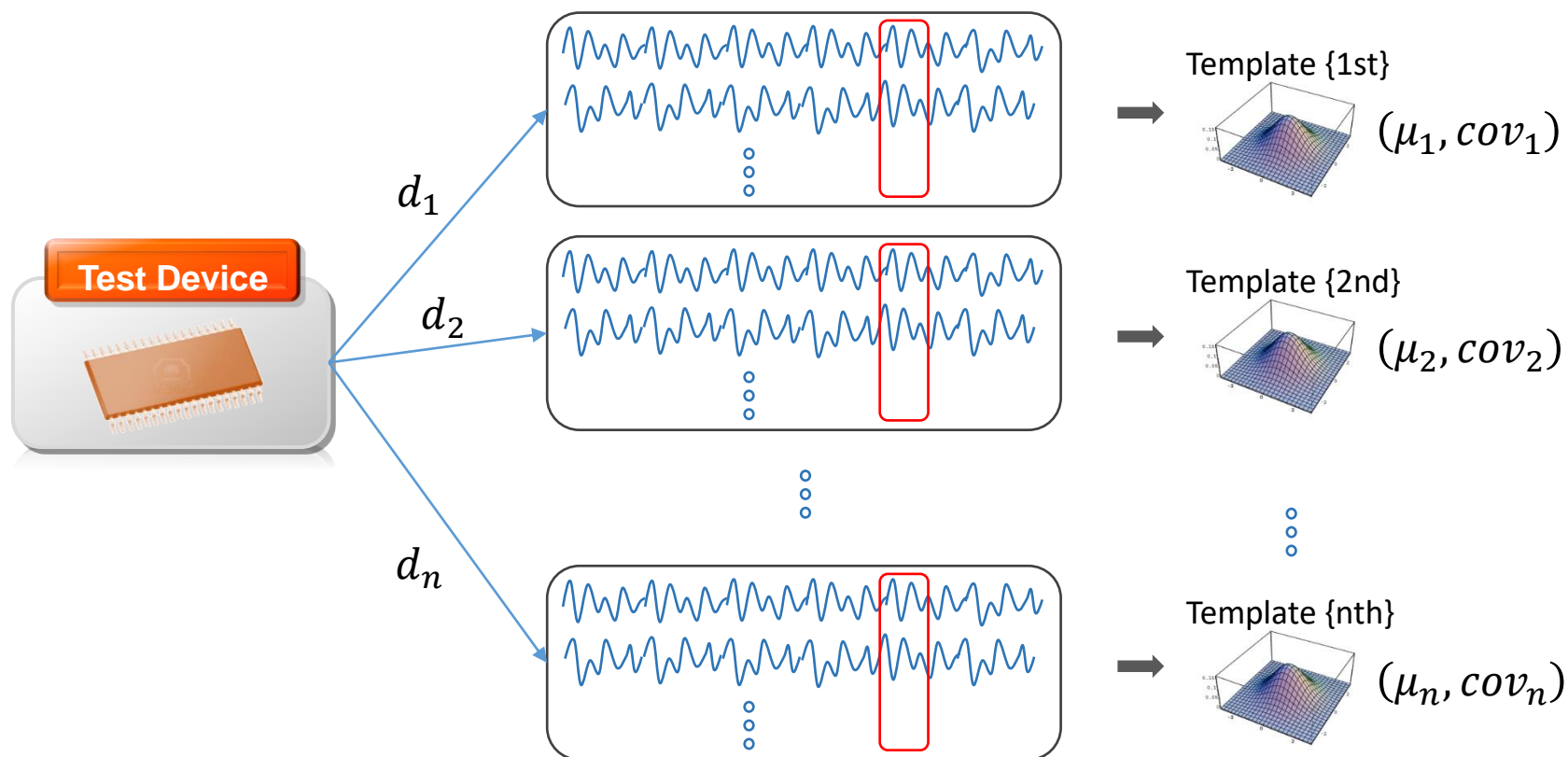
PCA : 데이터의 최적 표현

LDA : 데이터의 최적 분류

부채널 분석 예시 – 프로파일링 부채널 분석

■ 템플릿 공격 – 템플릿 구성 (Profiling Stage)

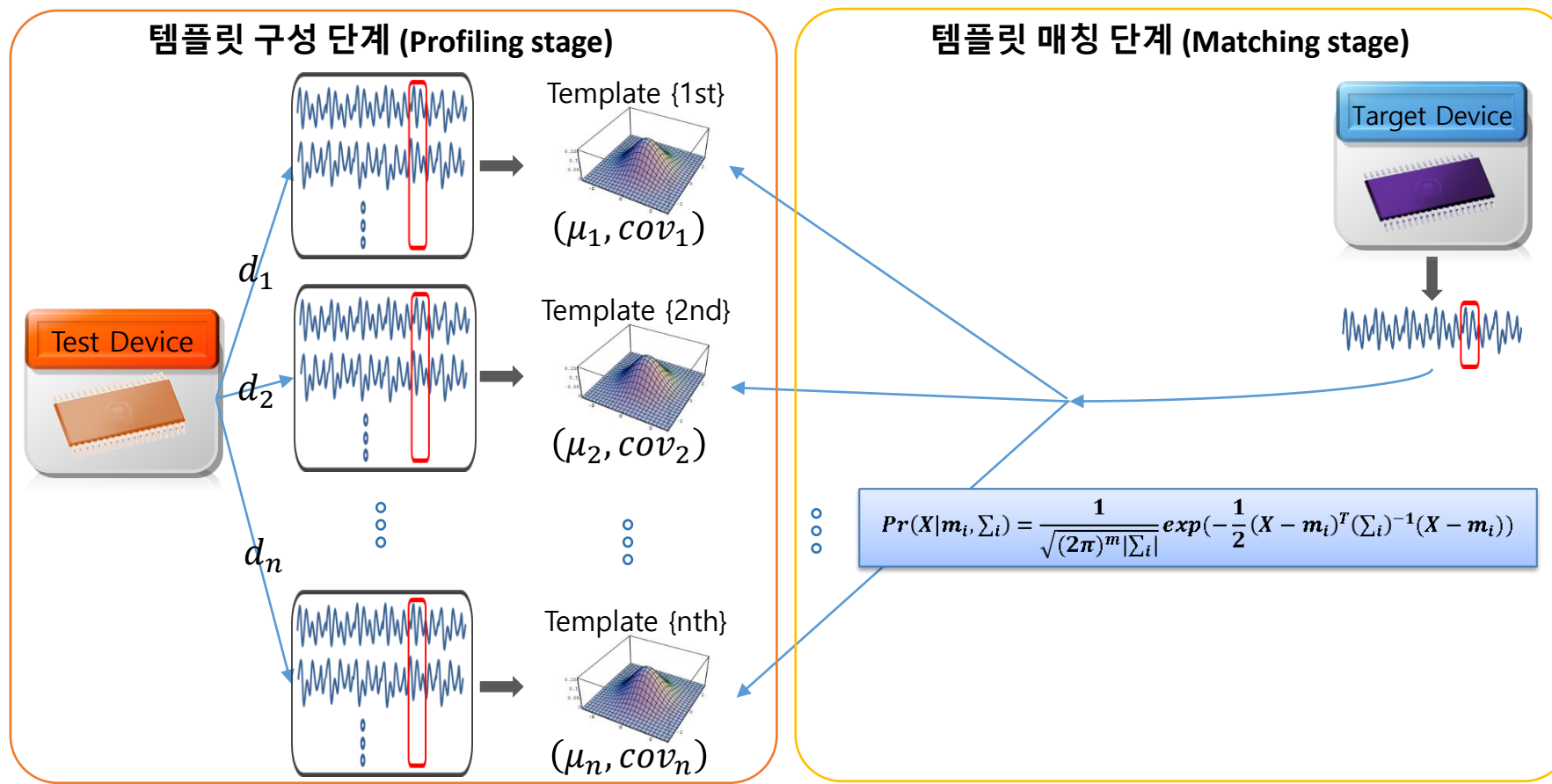
- (3단계) 추출된 데이터로부터 템플릿 구성
 - 선택/추출한 유의미한 시점에 대해 각 데이터에 대한 템플릿 구성
 - 가우시안 분포 → 유의미한 시점들의 평균 벡터와 공분산 행렬



부채널 분석 예시 – 프로파일링 부채널 분석

■ 템플릿 공격 – 템플릿 매칭 (Matching Stage)

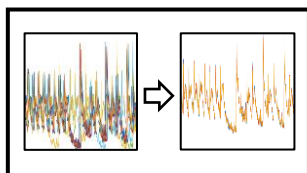
- 공격 타겟 트레이스로부터 얻은 파형과 템플릿을 매칭
 - 공격 대상 장비로부터의 파형을 템플릿 구성 단계에서 특징화한 템플릿과 비교
 - 최대우도법에 따라 가장 유사한 템플릿과 같은 클래스로 간주



부채널 분석 예시 - 딥러닝 기반 부채널 분석

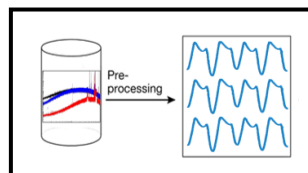
고전적인 부채널분석 vs 딥러닝 기반 부채널분석

프로파일링 공격



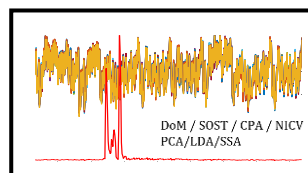
신호정렬

- Static
- DTW



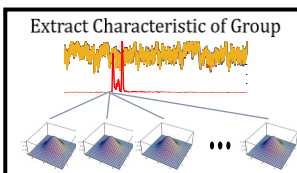
전처리

- 노이즈 필터
- 차원 압축



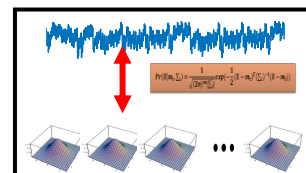
특징 식별

- SOST/CPA/...
- PCA/LDA/SSA



그룹별 특징화

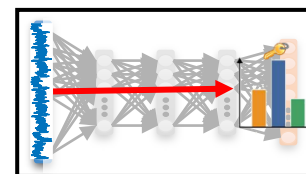
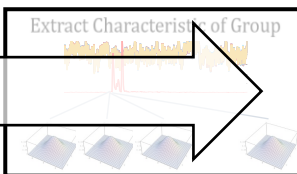
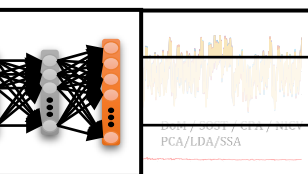
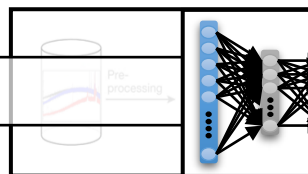
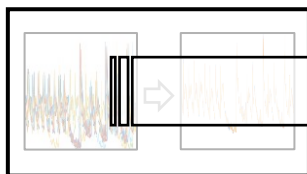
- 가우시안 분포
파라미터 추정



최대우도추정

- 베이지안 룰

딥러닝 기반 부채널 공격



네트워크 학습

- 네트워크 모델 선택
- 네트워크 파라미터 선택

분류 / 회귀

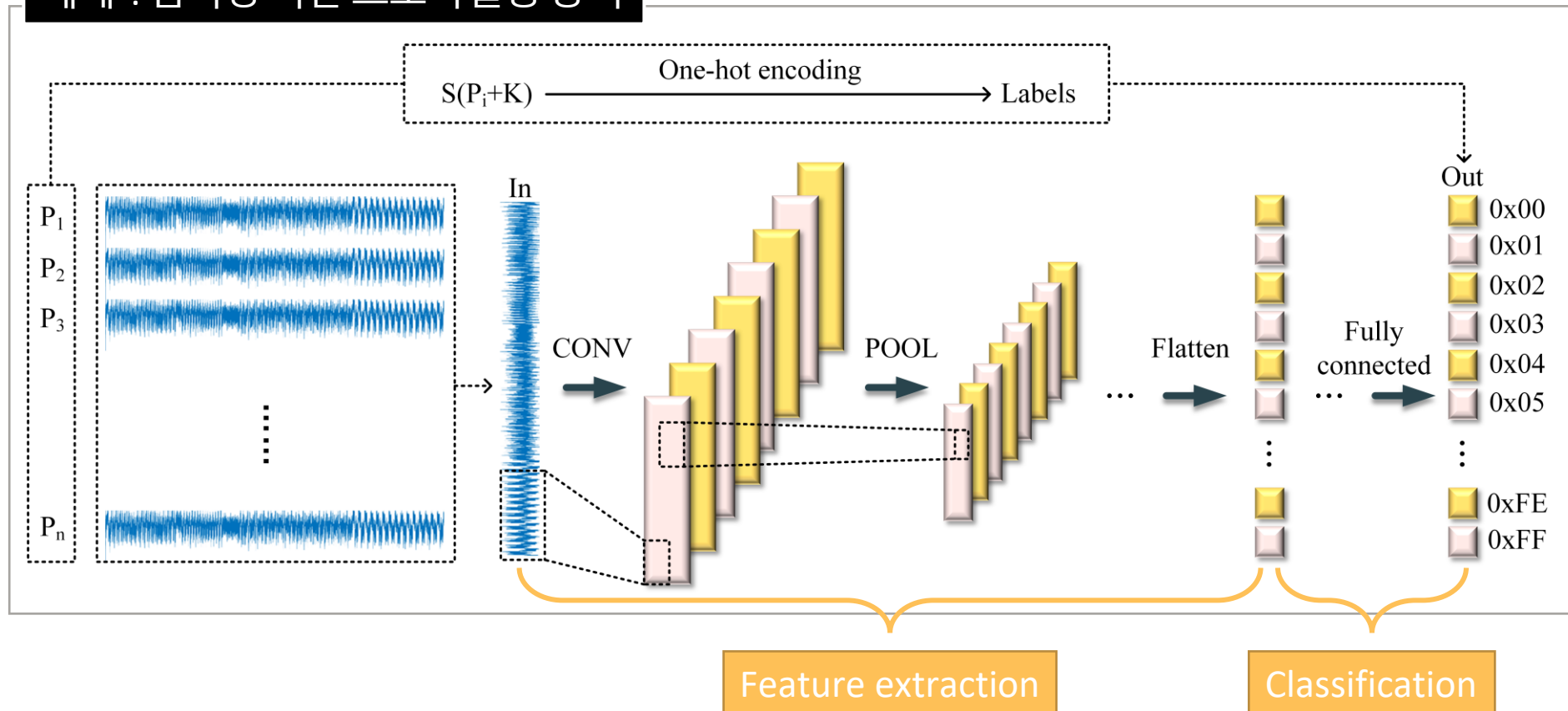
- 기학습된 네트워크
이용한 분류

딥러닝의 특징으로 전처리 과정을 최소화 및 파라미터 자동 탐색이 가능한 분석

부채널 분석 예시 - 딥러닝 기반 부채널 분석

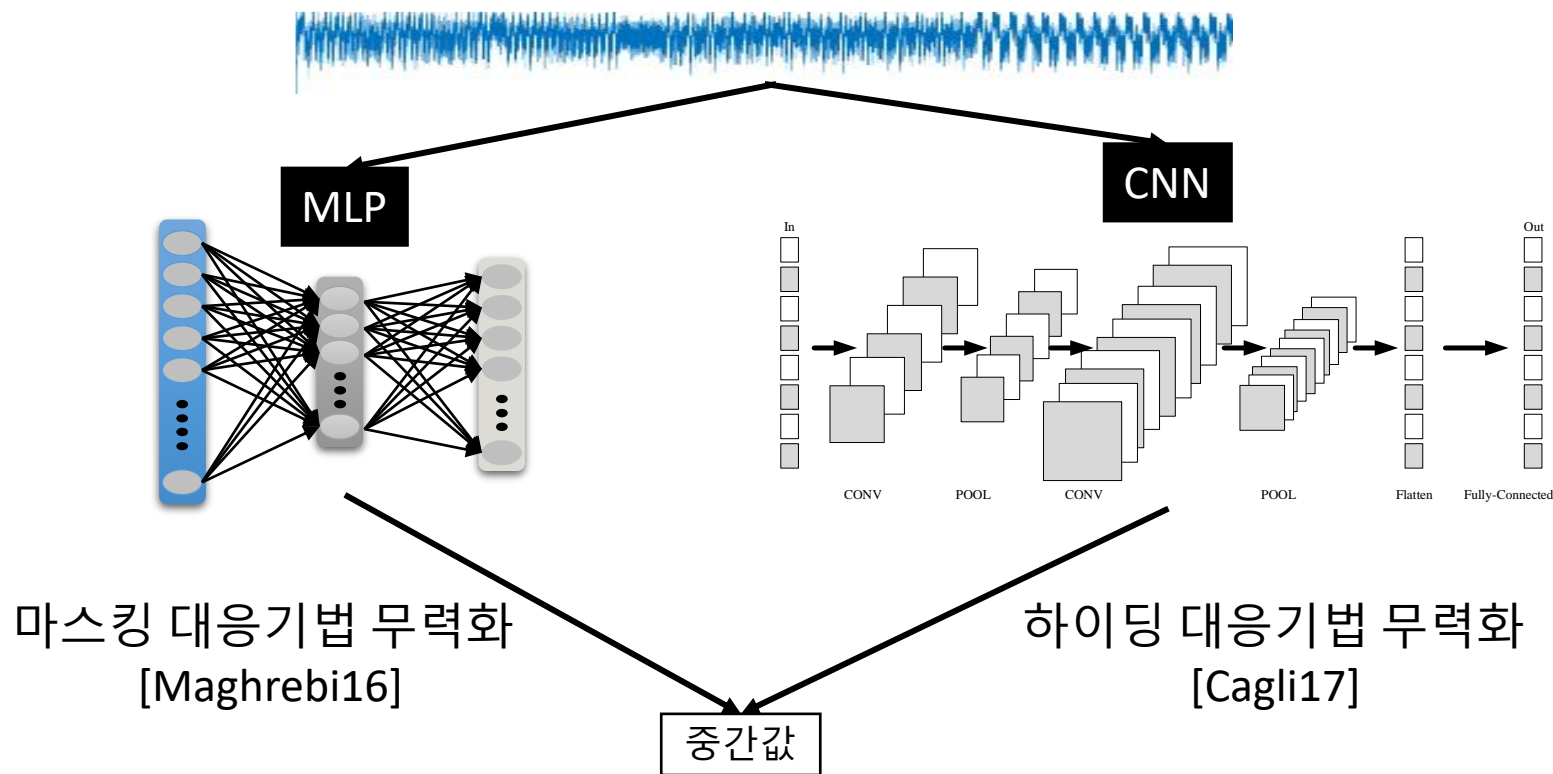


예제 : 딥러닝 기반 프로파일링 공격



부채널 분석 예시 - 딥러닝 기반 부채널 분석

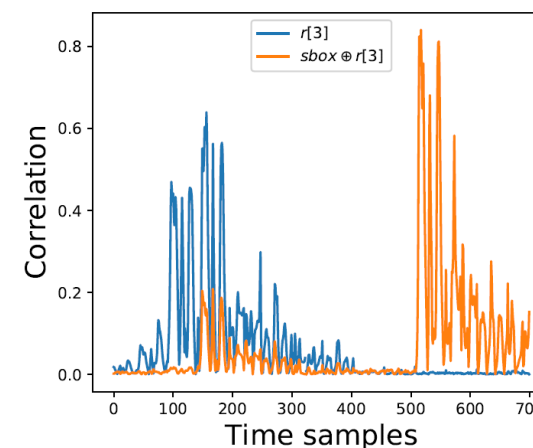
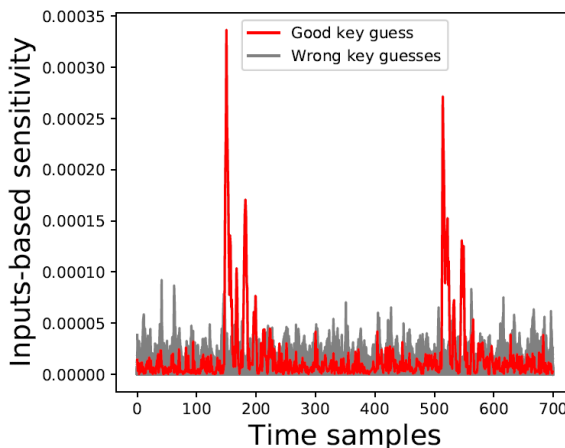
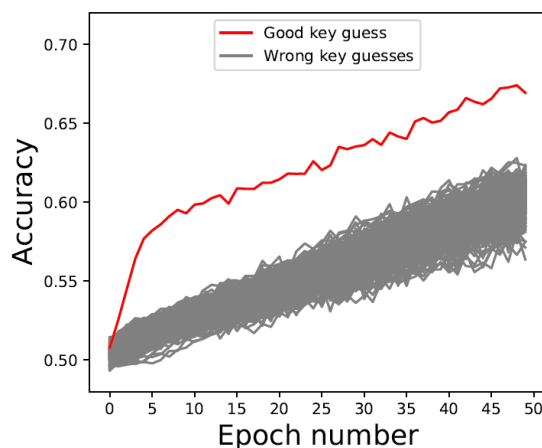
- 기존 프로파일링 기반 부채널 분석의 분류 기능을 딥러닝으로 대체 활용
- 뉴럴 네트워크의 자체 학습 특성을 이용한 파형 분류
- 대칭키 뿐만 아니라 공개키 암호시스템에 대한 분석도 진행됨 [Carbone19]



부채널 분석 예시 - 딥러닝 기반 부채널 분석

- 기존 비프로파일링 기반 부채널 분석에서 키를 찾기 위한 도구로 '딥러닝의 학습속도'를 활용
- 뉴럴 네트워크의 입출력이 아닌 훈련되는 경향성을 구별자로 사용 [Timon19]
- 키를 추측하여 계산한 중간값을 라벨로 사용하여 훈련
 - 옳은 키로 계산한 중간값은 파형과 연관 있음 → 학습됨
 - 틀린 키로 계산한 중간값은 파형과 연관 없음 → 학습되지 않음

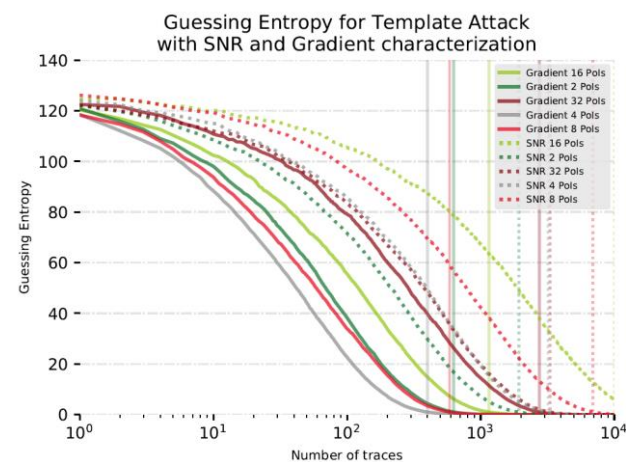
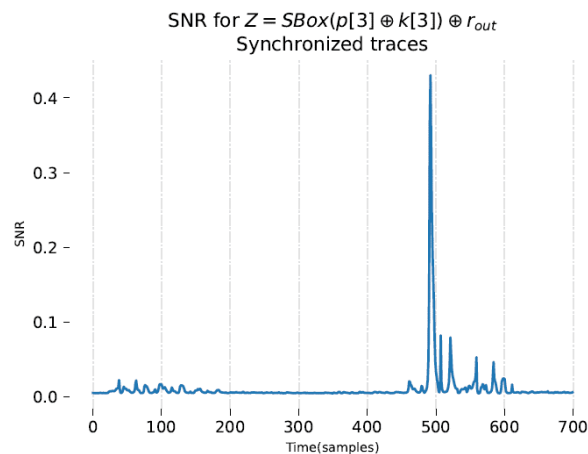
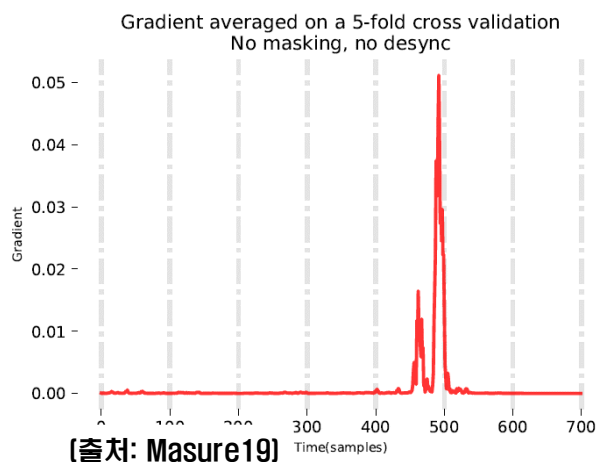
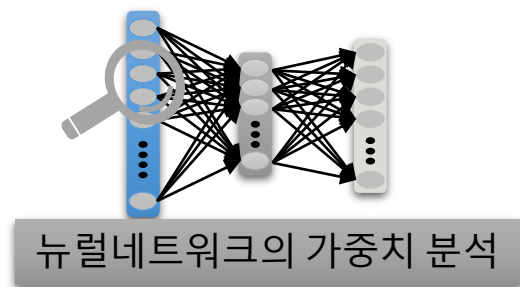
DDLA 를 이용한 마스킹 대응기법 무력화 (출처: Timon19)



Timon, Benjamin. "Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis." IACR Transactions on Cryptographic Hardware and Embedded Systems (2019): 107-131.

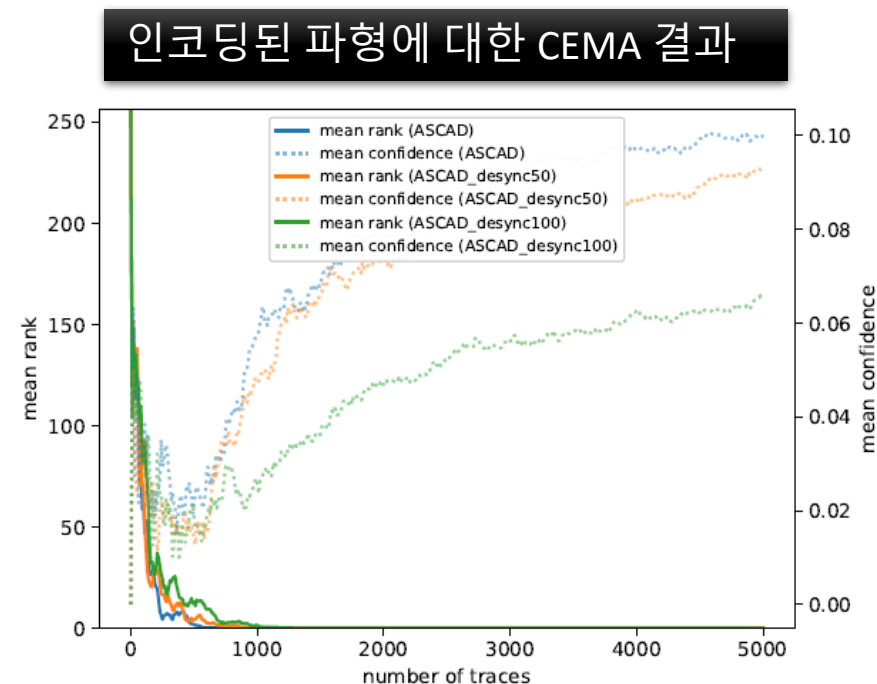
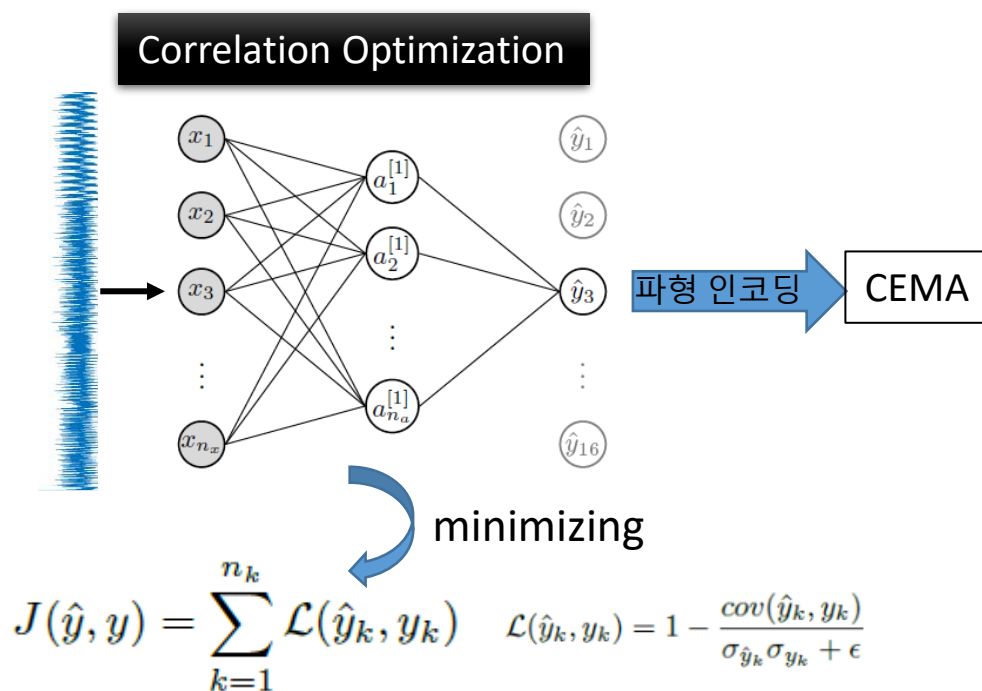
부채널 분석 예시 - 딥러닝 기반 부채널 분석

- 부채널 분석에서 정보 누출 시점을 찾기 위한 많은 기존 연구가 있으며 딥러닝을 활용하여 이 시점을 찾는 연구들이 최근 진행됨
- 독립된 3개 팀이 연구 제안 [Masure19, Timon19, Hettwer19]
- 뉴럴 네트워크 분석 기법을 이용한 누출정보 탐색
- 해당 부채널 누출정보 탐색을 이용한 POI 선택 Template attack 성능 향상됨



부채널 분석 예시 - 딥러닝 기반 부채널 분석

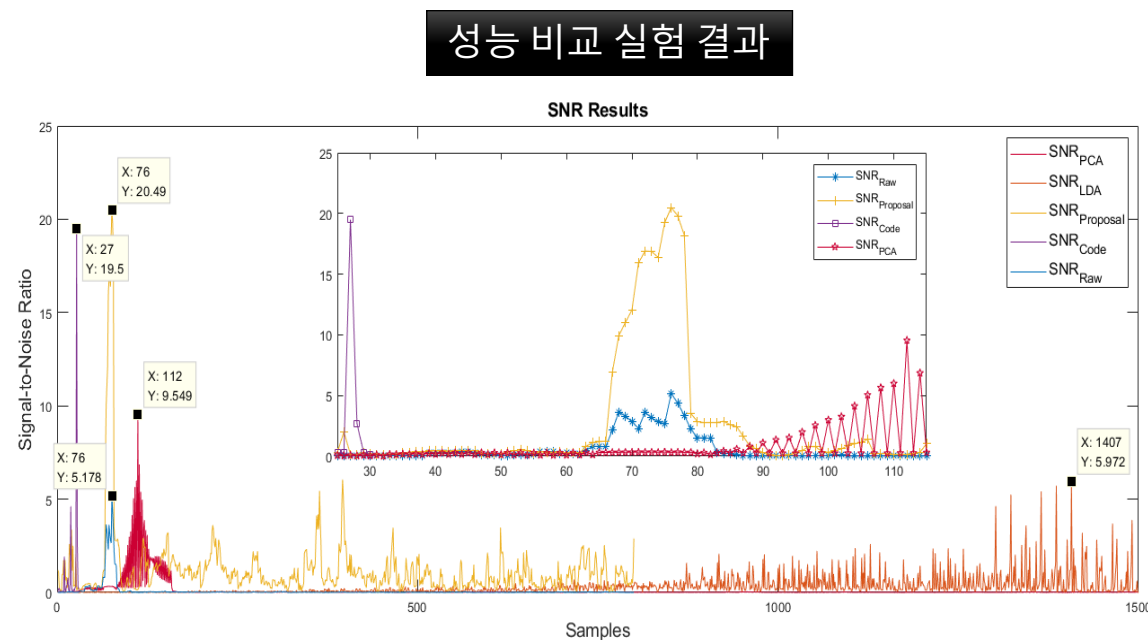
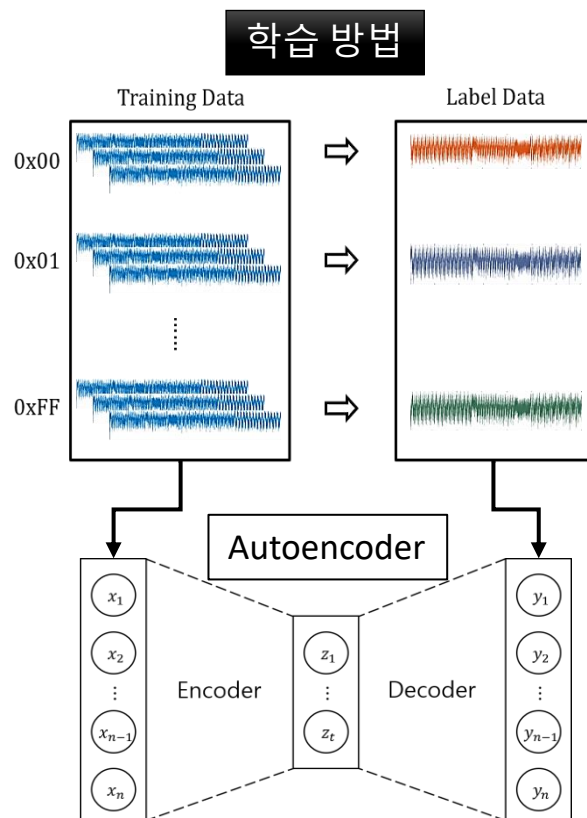
- 딥러닝을 활용한 소비 전력 모델링 기법 연구
- CPA peak이 최대가 되게끔 파형을 인코딩하도록 뉴럴 네트워크를 학습
- 주파수 도메인 정보를 입력데이터로 사용하면 MLP도 하이딩 무력화 가능



부채널 분석 예시 - 딥러닝 기반 부채널 분석

Autoencoder 를 이용한 잡음 제거 기술 제안

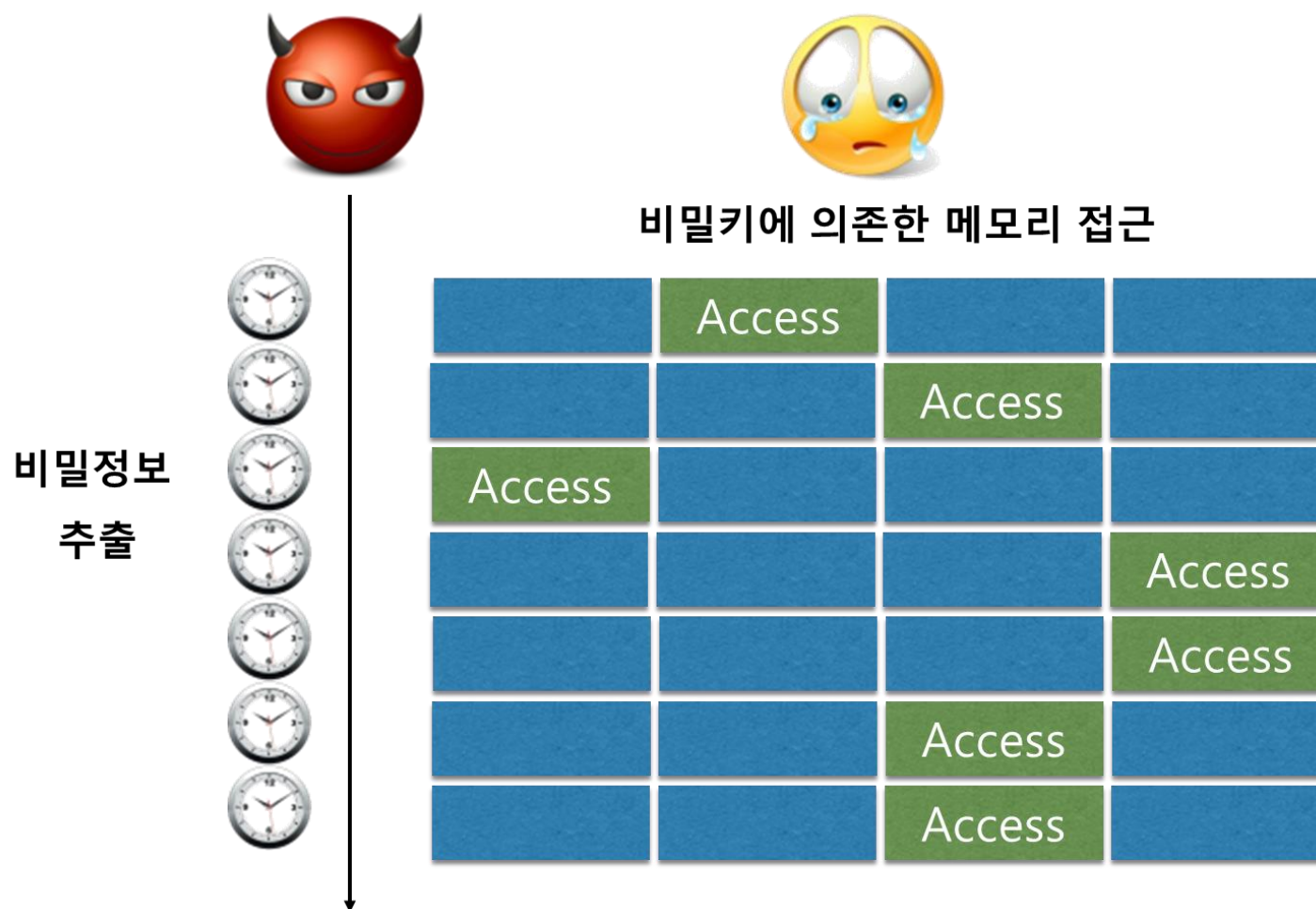
- Denoising autoencoder 와 다르게 노이즈가 감소된 평균 파형을 라벨로 사용하여 잡음 제거 학습 유도



기존 잡음 제거 기법들 보다 성능이 향상됨을 확인
인코딩된 파형에 대해서도 분석 성능 향상

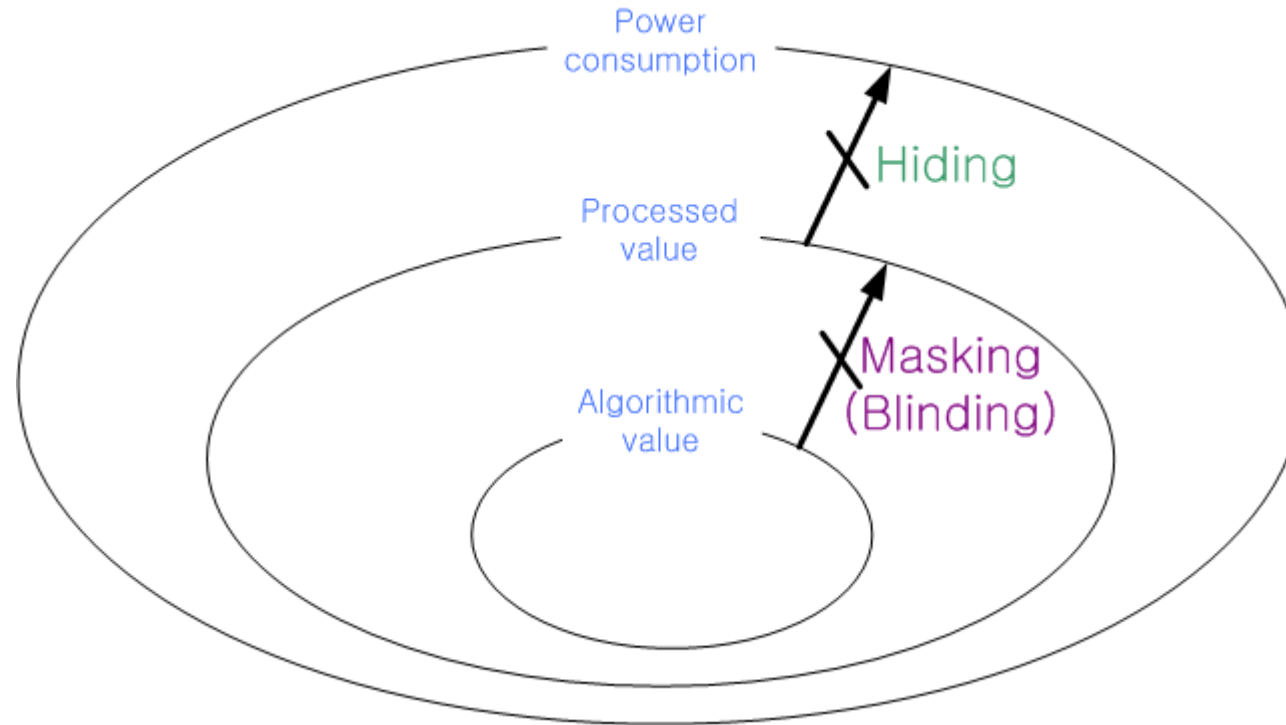
부채널 분석 예시 – 캐시 타이밍 어택

- 비밀키에 의존한 메모리 주소 접근이 있을 경우 비밀정보 추출 가능



부채널 분석 대응 핵심기술





Masking(Blinding) :

Break relation between algorithmic value and processed value

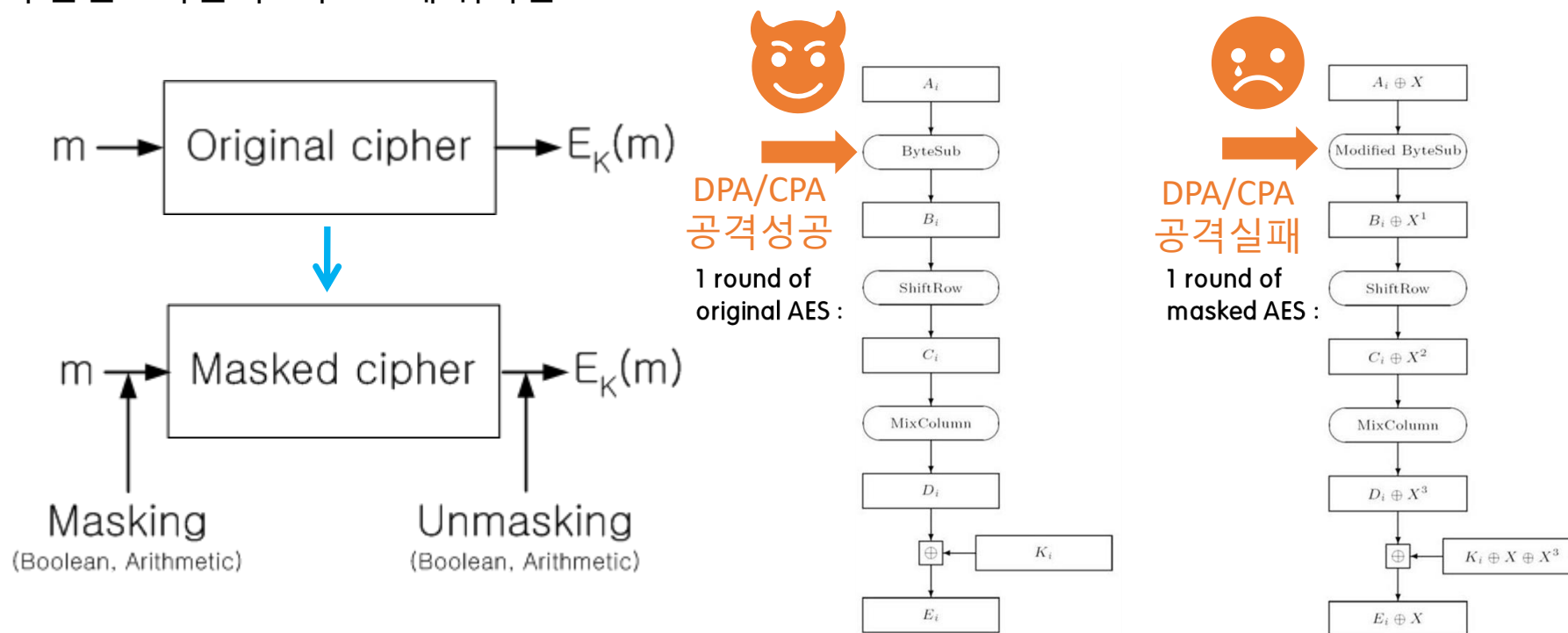
Hiding :

Break relation between processed value and power consumption

전력 분석 대응 기법 – Masking

■ 마스크 기법

- 암호가 연산되는 동안 계산되어질 수 있는 모든 중간 값을 랜덤한 난수로 마스크 시키는 방법
- 마스크 기법의 장점 : 1차 DPA에 대한 이론적 안전성 보장, 효율적임
- 마스크 기법의 단점 : 여전히 2차 DPA에 취약함



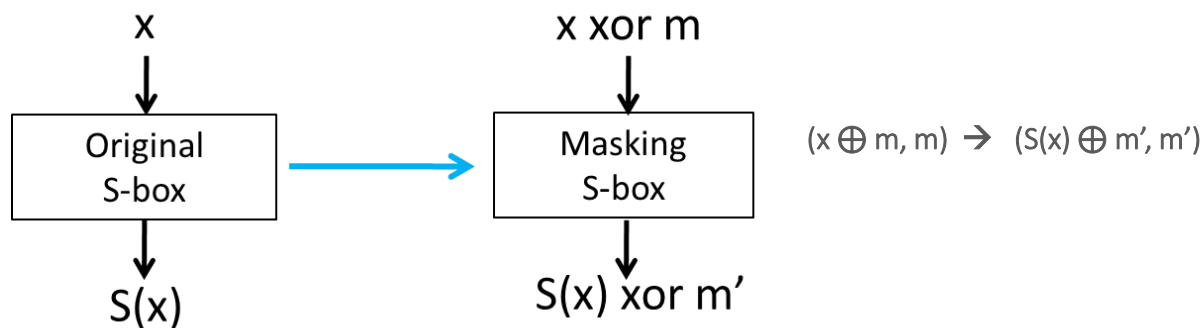
전력 분석 대응 기법 – Masking

■ 선형연산에 대한 마스킹 처리 : 상당히 쉬움

- $(x \oplus m, m) \rightarrow (L(x \oplus m), L(m))$ (x : 알고리즘값, m : 마스킹 난수)
- $(x \oplus m) \oplus m = x \rightarrow L(x \oplus m) \oplus L(m) = L(x)$

■ 비선형연산에 대한 마스킹 처리 : 상대적으로 매우 어려움

- $(x \oplus m, m) \rightarrow (NL(x \oplus m), NL(m))$ 으로 연산될 수 없음
- S-box에 대한 마스킹 처리 : 암호알고리즘 수행 전 마스킹 테이블 생성
 1. m, m' 생성
 2. $0 \leq x \leq 255$ 에 대해 $MS(x \oplus m) = S(x) \oplus m'$ 을 만족하는 MS 생성

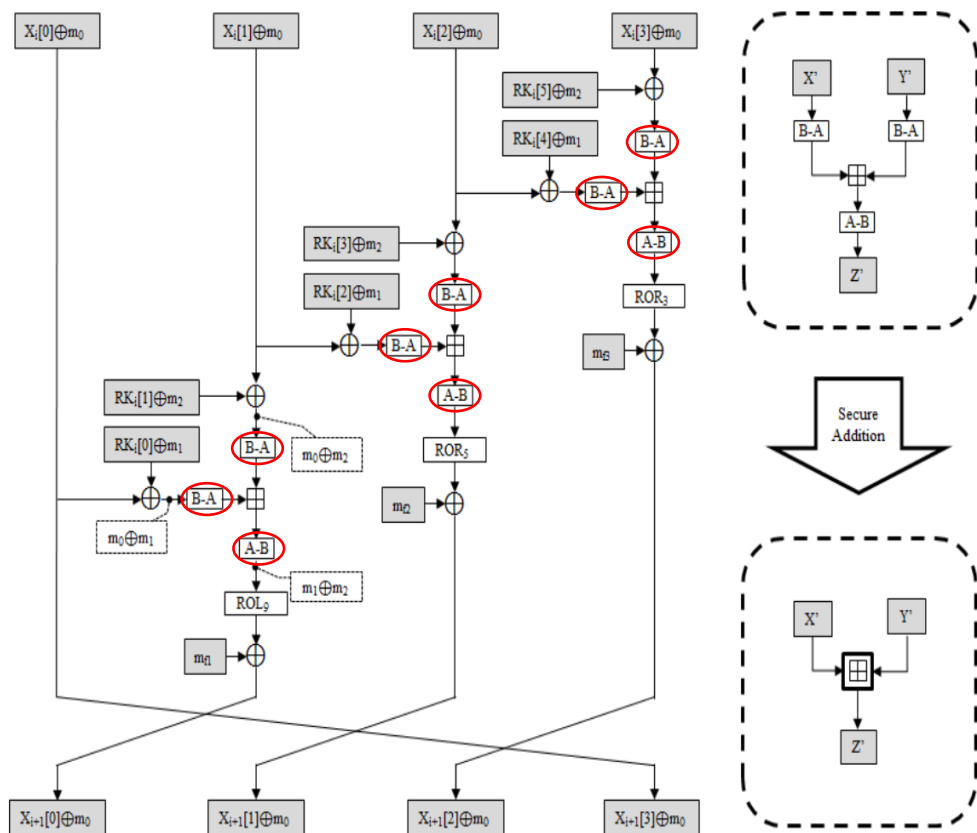


전력 분석 대응 기법 – Masking

- 비선형연산에 대한 마스킹 처리 : 상대적으로 매우 어려움
 - 덧셈 연산에 대한 마스킹 처리 : $(x \oplus m, y \oplus m')$ 으로부터 $(x+y) \oplus m''$ 을 생성하는 연산 구조 필요
 1. $x \oplus m \rightarrow x - m, y \oplus m' \rightarrow y - m'$ 으로 변환하는 연산 필요
 2. $(x-m)+(y-m')-m''+m+m' = (x+y)-m''$
 3. $(x+y)-m'' \rightarrow (x+y) \text{ xor } m''$ 으로 변환하는 연산필요
 - arithmetic 마스킹에서 Boolean 마스킹으로의 변환에 엄청난 비용 소요
- 국산 표준 블록암호가 사용하는 비선형 연산
 - ARIA : 유한체 위에서의 역원 연산 기반 S-box
 - SEED : S-box 및 덧셈연산
 - HIGHT, LEA : 덧셈연산
- S-box를 비선형 연산으로 사용하는 ARIA, SEED, AES에 비해 HIGHT, LEA 등은 부채널 대응 기술 적용 시 속도 저하 현상이 심함
- 암호 알고리즘 설계 시점부터 부채널 대응기술에 대한 효율성을 고려해야 함

▪ LEA 마스크 대응기술

- 입력 마스크 m_1 , 라운드 키 마스크 m_2 를 이용하여 산술 마스크링과 불 마스크링의 변환 알고리즘을 적용
- 라운드 마다 3번의 덧셈 마스크링 필요



Input : A, r_x

Output : x'

- $\Gamma = \gamma$
- $T = 2\Gamma$
- $x' = \Gamma \oplus r_x$
- $\Omega = \Gamma \wedge x'$
- $x' = T \oplus A$
- $\Gamma = \Gamma \oplus x'$
- $\Gamma = \Gamma \wedge r_x$
- $\Omega = \Omega \oplus \Gamma$
- $\Gamma = T \wedge A$
- $\Omega = \Omega \oplus \Gamma$
- for $i=1$ to $k-1$ do
 - $\Gamma = T \wedge r_x$
 - $\Gamma = \Gamma \oplus \Omega$
 - $T = T \wedge A$
 - $\Gamma = \Gamma \oplus x'$
 - $T = 2\Gamma$
- $x' = x' \oplus T$

Goubin AtoB 마스크 알고리즘

김창균 et al., LEA에 대한 마스크 기반
부채널분석 및 대응 방법, 정보보호학회논문지,
2015

전력 분석 대응 기법 – Hiding

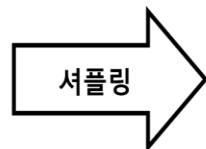
■ 셔플링 기법

- 독립적인 연산의 순서를 무작위로 배열하여 전력 파형에 SNR를 낮추어 공격에 대응

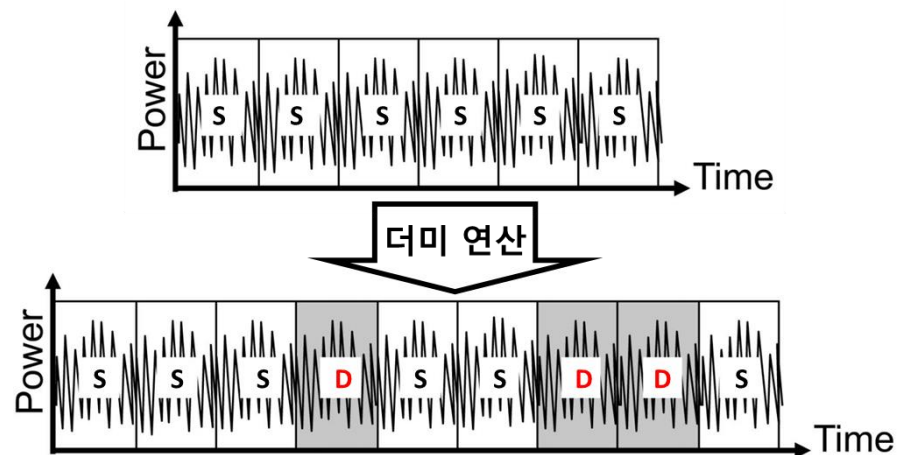
■ 더미 연산 추가

- 연산이 진행되는 도중 의미 없는 연산을 추가하여 공격자가 올바른 시점정렬을 어렵게 만드는 대응기술

1st	2nd	3rd	4th
5th	6th	7th	8th
9th	10th	11th	12th
13th	14th	15th	16th



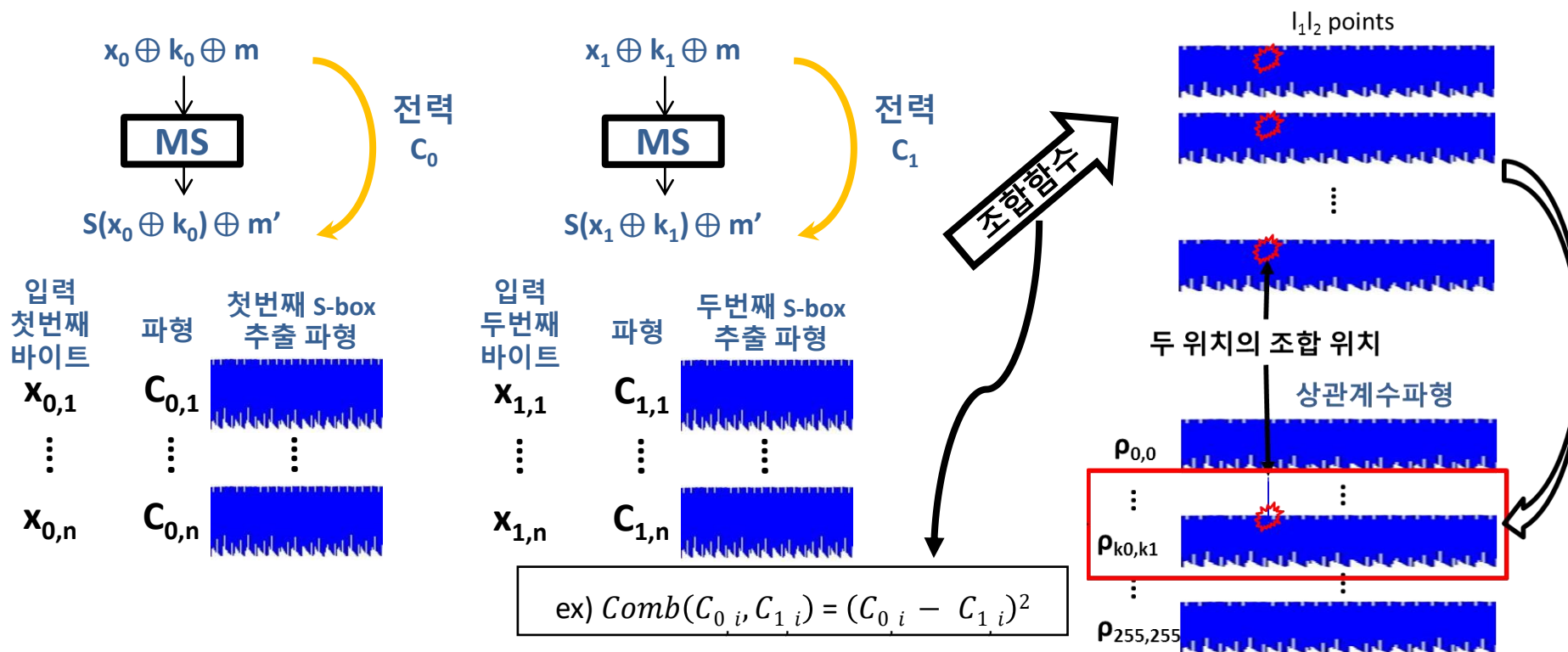
15th	12th	7th	16th
11th	4th	8th	6th
3rd	9th	1st	13th
10th	5th	14th	2nd



부채널 분석 대응기술 무력화

Second-Order DPA / CPA

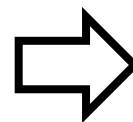
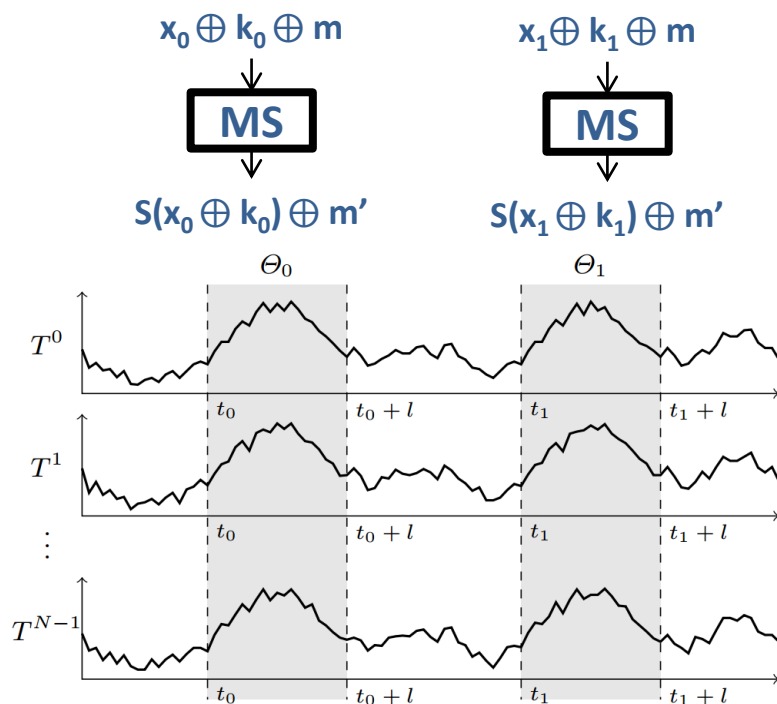
- 공격 대상 알고리즘 : 마스크 블록암호
- 공격 가정 : 다수의 임의 평문·암호문 사용가능, 공격에 사용될 두 시점에서 사용하는 마스크 값이 동일
- 공격 개요 : 마스크 값이 같은 두 시점을 조합하여 해당 시점 중간 값들의 XOR 파형을 얻음 → 조합파형으로 CPA



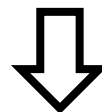
부채널 분석 대응기술 무력화

Collision Attack

- 공격 대상 알고리즘 : 일반/마스킹 블록암호
- 공격 가정 : 다수의 임의 평문·암호문 사용가능, 공격에 사용될 두 시점에서 사용하는 마스킹 값이 동일
- 공격 개요 : 서로 다른 두 연산의 파형 간에 CPA 분석을 통해 같은 데이터가 연산 되는지 아닌지 분석
- 이러한 전력사이에 충돌 발생 여부를 이용하여 두 비밀키의 차분 정보를 알 수 있음



$$\hat{\rho}_{\theta_0, \theta_1}(t) = \frac{\text{Cov}(\theta_0(t), \theta_1(t))}{\sigma_{\theta_0(t)} \sigma_{\theta_1(t)}}$$



$$\text{If } S(x_0 \oplus k_0) \oplus m' = S(x_1 \oplus k_1) \oplus m'$$

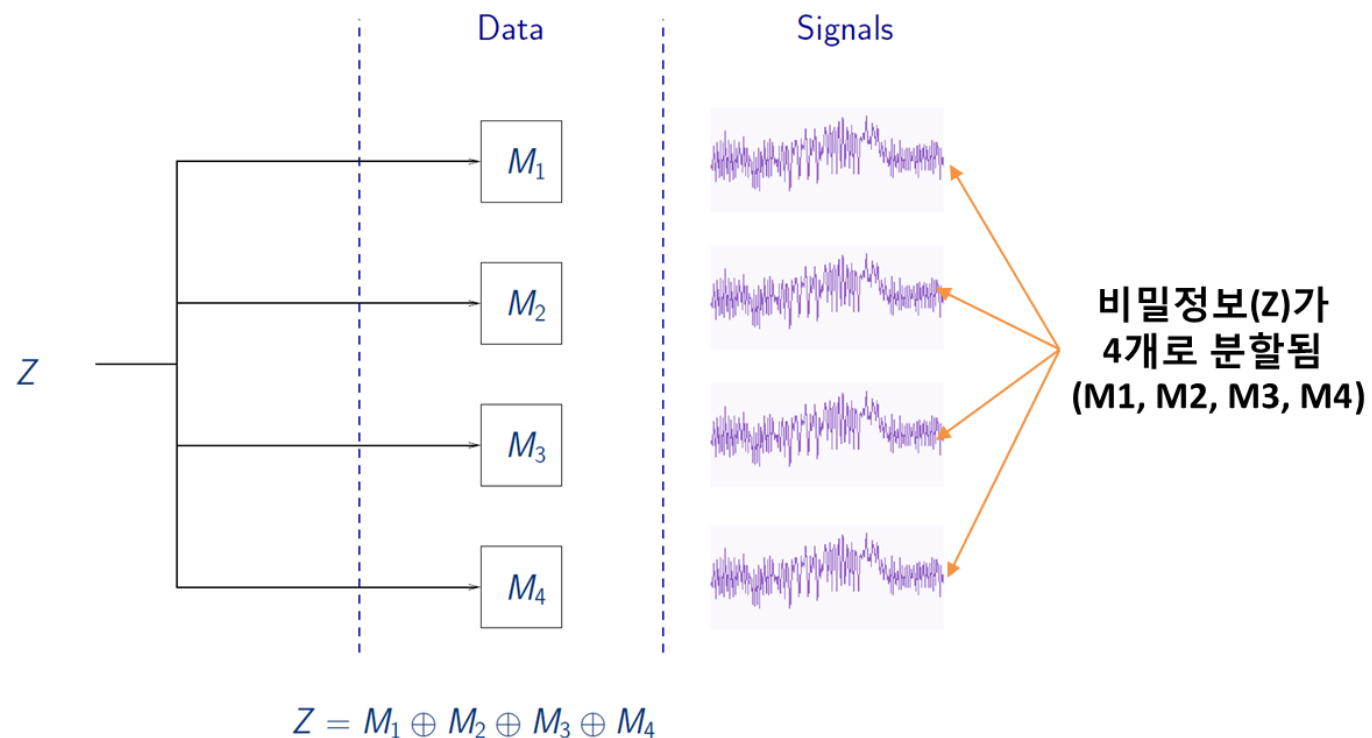
$$\Rightarrow x_0 \oplus k_0 = x_1 \oplus k_1$$

$$\Rightarrow k_0 = k_1 \oplus \underline{x_0 \oplus x_1}$$

고차 대응기술

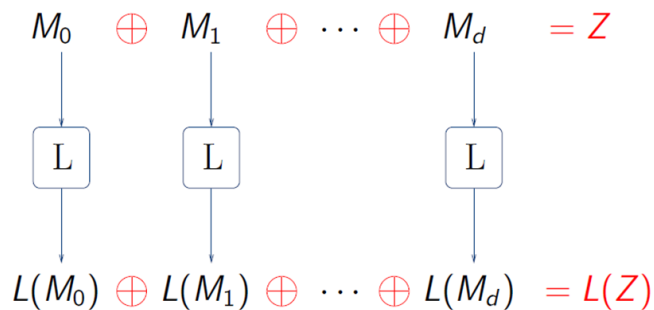
고차 마스킹 대응기술

- 비밀정보를 3개 이상으로 공유하여 서로 다른 시점에 비밀정보에 대한 부채널 신호를 분할
- 공격자는 분리된 데이터에 대한 모든 부채널 신호를 취득, 조합해 고차 전력 분석 수행
- 3차 마스킹 이상 적용 시 현실적으로 분석 불가능



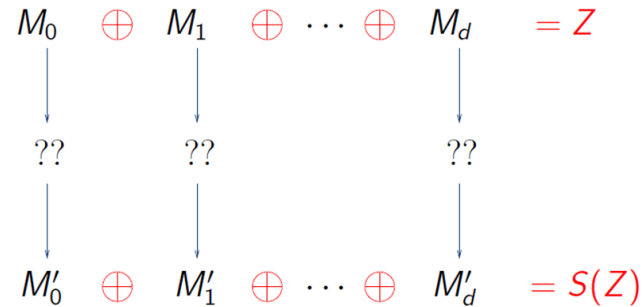
고차 마스킹 대응기술

- 비밀정보를 3개 이상으로 공유하여 서로 다른 시점에 비밀정보에 대한 부채널 신호를 분할
- 공격자는 분리된 데이터에 대한 모든 부채널 신호를 취득, 조합해 고차 전력 분석 수행
- 3차 마스킹 이상 적용 시 현실적으로 분석 불가능

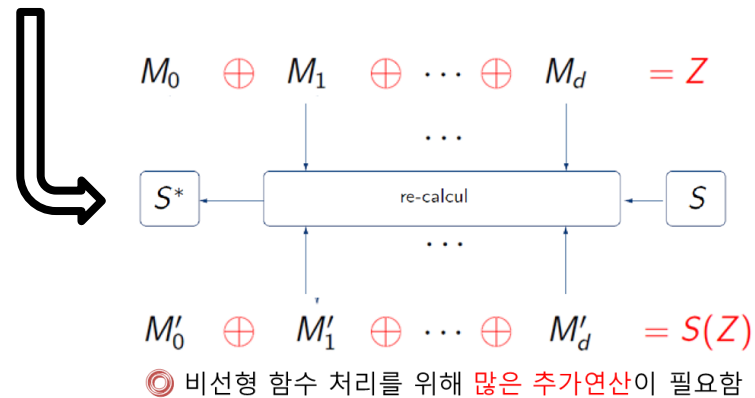


◎ 선형 함수의 경우 고차 마스킹 대응기술 적용이 간단함

VS



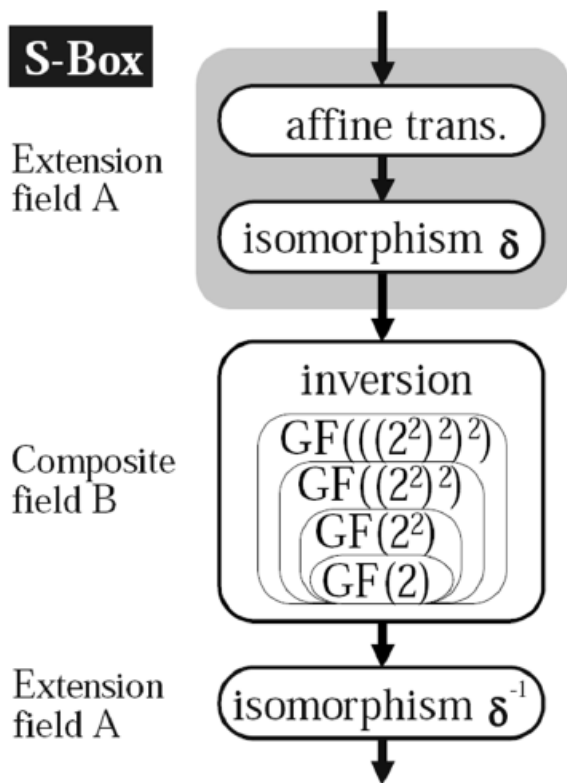
◎ 비선형 함수에 대한 고차 마스킹 대응기술의 경우 적용이 힘들



고차 대응기술

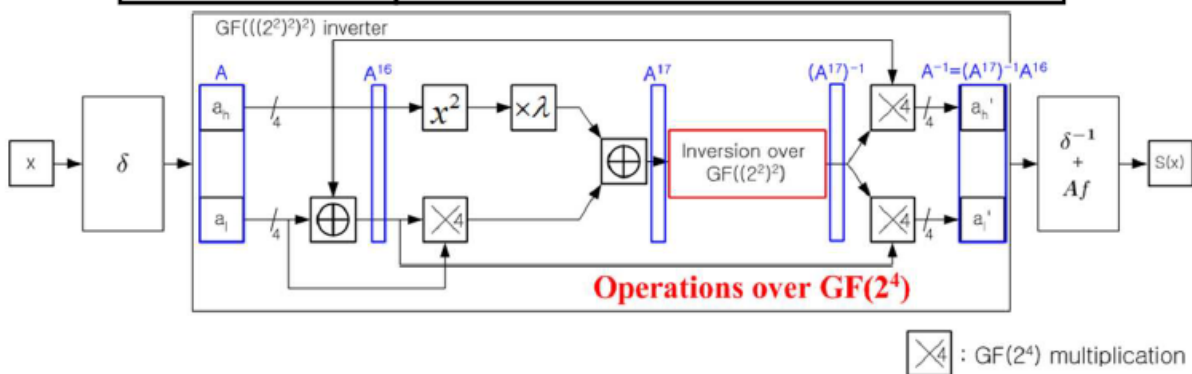
고차 마스킹 대응기술 (AES 예시)

- 합성체를 이용하여 AES S-Box의 구현이 가능



$$\begin{aligned}
 GF(2^2) &: P_0(x) = x^2 + x + 1, \text{ where } P_0(\alpha) = 0, \\
 GF((2^2)^2) &: P_1(x) = x^2 + x + \alpha, \text{ where } P_1(\beta) = 0, \\
 GF(((2^2)^2)^2) &: P_2(x) = x^2 + x + \lambda, \text{ where } \lambda = (\alpha + 1)\beta, P_2(\gamma) = 0.
 \end{aligned}$$

Inversion operation over the composite field	<ul style="list-style-type: none"> STEP 1 : $a_h\gamma + a_l = \delta(x)$ STEP 2 : $d = \lambda a_h^2 + a_l(a_h + a_l)$ STEP 3 : $d' = d^{-1}$ STEP 4 : $(a_h', a_l') = (d' a_h, d'(a_h + a_l))$ STEP 5 : $\delta^{-1}(a_h' \gamma + a_l')$
---	---



고차 대응기술

고차 마스킹 대응기술 (AES 예시)

- 합성체 AES S-Box 연산을 기반으로 고차 마스킹을 적용

Algorithm. d^{th} -order masking of AES S-box

Input : (x_0, x_1, \dots, x_d) s.t. $x = \bigoplus_{i=0}^d x_i$

Output : (y_0, y_1, \dots, y_d) s.t. $y = \text{S-box}(x) = \bigoplus_{i=0}^d y_i$

1_(a). $(H_0 || L_0, H_1 || L_1, \dots, H_d || L_d) = (T5[x_0], T5[x_1], \dots, T5[x_d])$

1_(b). $(w_0, w_1, \dots, w_d) = (T3[H_0], T3[H_1], \dots, T3[H_d])$

1_(c). $(t_0, t_1, \dots, t_d) = (H_0 \oplus L_0, H_1 \oplus L_1, \dots, H_d \oplus L_d)$

2. $(L_0, L_1, \dots, L_d) = \text{SecMult4}((t_0, t_1, \dots, t_d), (L_0, L_1, \dots, L_d))$

3. $(w_0, w_1, \dots, w_d) = (w_0 \oplus L_0, w_1 \oplus L_1, \dots, w_d \oplus L_d)$

4. $(w_0, w_1, \dots, w_d) = \text{SecInv}((w_0, w_1, \dots, w_d))$

5. $(H_0, H_1, \dots, H_d) = \text{SecMult4}((w_0, w_1, \dots, w_d), (H_0, H_1, \dots, H_d))$

6. $(L_0, L_1, \dots, L_d) = \text{SecMult4}((w_0, w_1, \dots, w_d), (L_0, L_1, \dots, L_d))$

7. $(y_0, y_1, \dots, y_d) = (T6[H_0 || L_0], T6[H_1 || L_1], \dots, T6[H_d || L_d])$

8. If d is odd, $y_0 = y_0 \oplus 0x63$

9. Return (y_0, y_1, \dots, y_d)

Algorithm. $GF(2^4)$ SecInv function

Input : (x_0, x_1, \dots, x_d) s.t. $x = \bigoplus_{i=0}^d x_i$

Output : (y_0, y_1, \dots, y_d) s.t. $y = x^{14} = \bigoplus_{i=0}^d y_i$

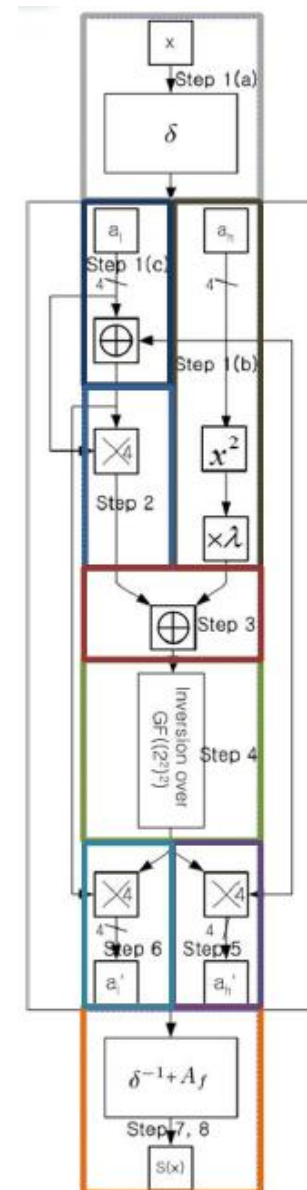
1. $(w_0, w_1, \dots, w_d) = (T1[x_0], T1[x_1], \dots, T1[x_d])$ // x^2

2. RefreshMasks((w_0, w_1, \dots, w_d)) // Eliminate the dependence between two input tuples

3. $(z_0, z_1, \dots, z_d) = \text{SecMult4}((w_0, w_1, \dots, w_d), (x_0, x_1, \dots, x_d))$ // x^3

4. $(z_0, z_1, \dots, z_d) = (T2[z_0], T2[z_1], \dots, T2[z_d])$ // x^{12}

5. $(y_0, y_1, \dots, y_d) = \text{SecMult4}((z_0, z_1, \dots, z_d), (w_0, w_1, \dots, w_d))$ // x^{14}



고차 대응기술

■ 고차 마스킹 대응기술이 적용된 AES

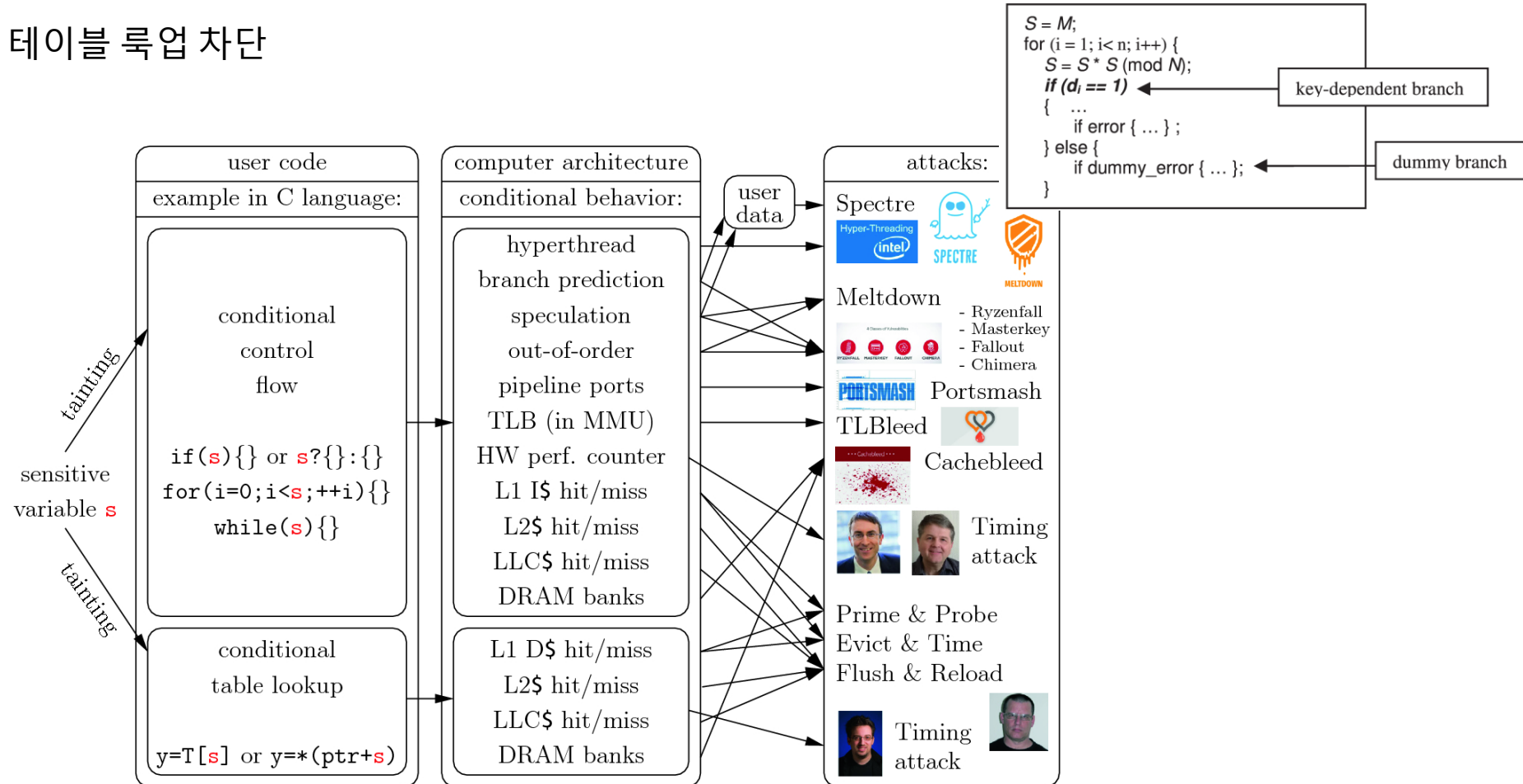
- 비밀정보를 3개 이상으로 공유하여 서로 다른 시점에 비밀정보에 대한 부채널 신호를 분할
- 공격자는 분리된 데이터에 대한 모든 부채널 신호를 취득, 조합해 고차 전력 분석 수행
- 3차 마스킹 이상 적용 시 현실적으로 분석 불가능

Method	Cycles ($\times 10^3$ cc)			Table Size (Bytes)	
	KeyExpand	Encryption	Total	RAM	ROM
Original AES					
No Masking (Straightforward AES)	2.2	9.0	11.2	0	256
First Order Masking					
ACNS'06 [9] (No dummy, No Shuffling)	4.6	14.9	19.5	256	256
Second Order Masking					
FSE'08 [17] (Complete second-order masking)	247.4	950.0	1197.4	256	256
CHES'10 [18] (Complete second-order masking)	144.1	531.2	675.4	0	768
Ours (Complete second-order masking)	66.2	199.3	265.5	0	816
Ours (KeyExpand (first) Enc. (1,2,9,10;second, 3-8:first))	5.2	90.6	95.8	256	1062
Third Order Masking					
CHES'10 [18] (Complete third-order masking)	293.4	1102.9	1396.3	0	768
Ours (Complete third-order masking)	114.6	346.8	461.3	0	816
Ours (KeyExpand (first) Enc. (1,2,9,10;third, 3-8:first))	5.5	149.6	155.1	256	1062

캐시, 타이밍 어택 대응기술

암호키에 의존한 메모리 접근 주소 차이 발생 가능성 차단

- 암호키에 의존한 조건문 차단
- 암호키에 의존한 테이블 룩업 차단



PKC 부채널 분석 및 대응기술



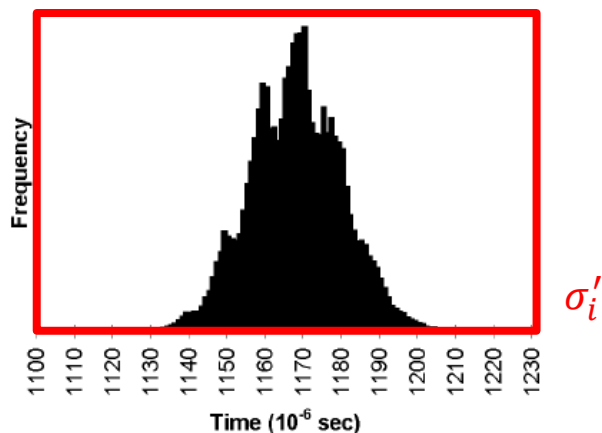


PKC 구현 시 부채널 분석 취약요소

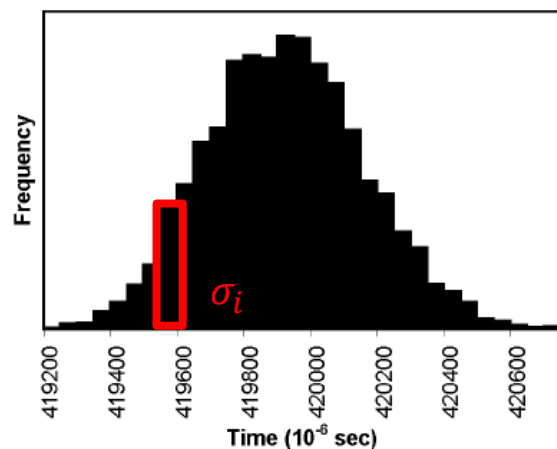
- 큰수 연산 구현에 따른 상수시간 구현 고려 필요 (Timing Attack)
- 암호키비트 처리 시 상수시간 구현 필요 (Timing Attack)
- 암호키비트에 따른 연산 차이 발생 제거 (SPA)
- 중간 연산 값의 난수 블라인딩 (DPA/CPA)
- 비밀정보에 의존한 데이터의 극심한 편중 현상 제거 (공격자는 입력값을 조절하면서 이런 현상을 활용할 수도 있음)
- 큰수 연산 설계 시, 입력 값에 따른 전력/전자 소비량 차이 제거
- 등등..

Timing Attack

- 공격 대상 알고리즘 : S&M(Square&Multiply) RSA, D&A(Double&Add) ECC
- 공격 가정 : 제곱과 곱셈이 서로 다른 알고리즘을 사용, 각 지수 bit에 해당하는 연산 타이밍 신호 정보 획득 가능
- 공격 개요 : 지수 bit별 연산 타이밍 신호 정보를 다수 수집하여 표준편차를 계산, 추측 지수 bit에 대한 정보와 비교



i 번째 빅넘버 곱셈 연산 타이밍 신호

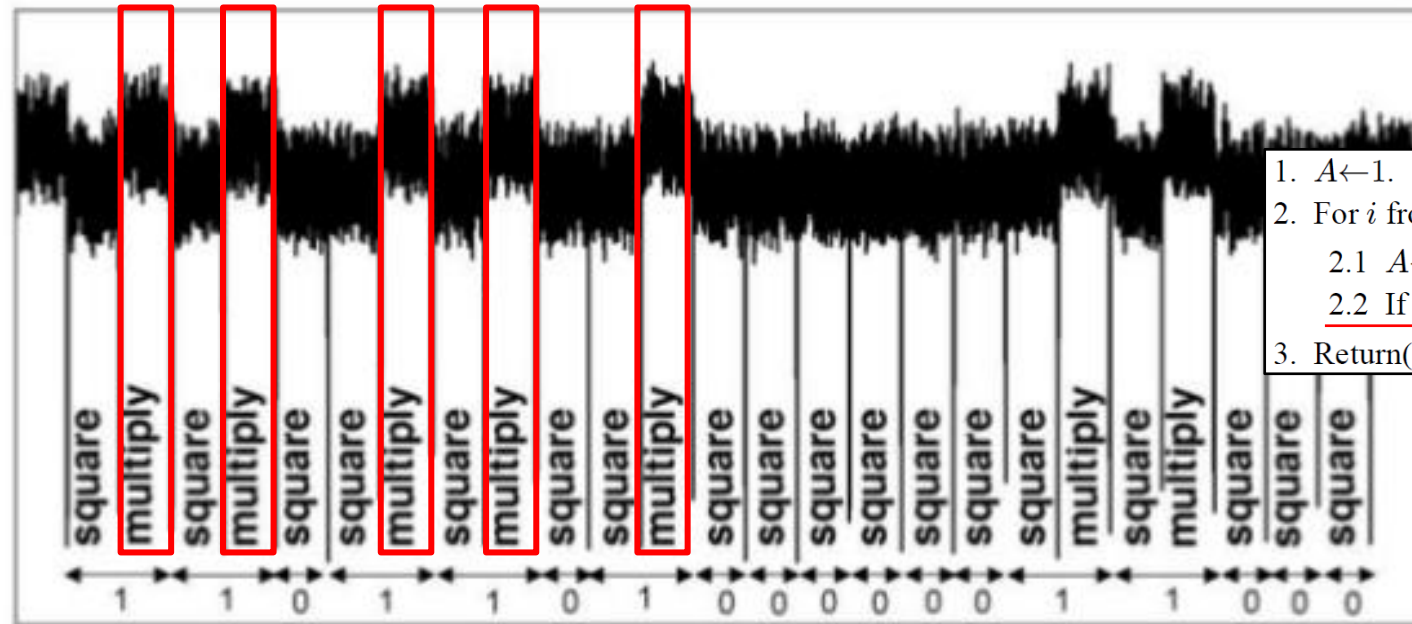


전체 빅넘버 지수승 연산 타이밍 신호

PKC 부채널 분석 및 대응기술

▪ SPA (Simple Power Attack)

- 공격 대상 알고리즘 : S&M(Square&Multiply) RSA, D&A(Double&Add) ECC
- 공격 가정 : 빅넘버 제곱 연산과 곱셈 연산이 서로 다른 알고리즘을 사용
- 공격 개요 : 암호화 연산 동안의 전력 소모량을 수집하여 각각의 연산 별 파형의 특징을 이용하여 비밀 키를 복원



1. $A \leftarrow 1$.
2. For i from t down to 0 do the following:
 - 2.1 $A \leftarrow A \cdot A$.
 - 2.2 If $e_i = 1$, then $A \leftarrow A \cdot g$.
3. Return(A).

PKC 부채널 분석 및 대응기술

■ Multiplication by School Book Method

Algorithm Multiple-precision multiplication

INPUT: positive integers x and y having $n + 1$ and $t + 1$ base b digits, respectively.

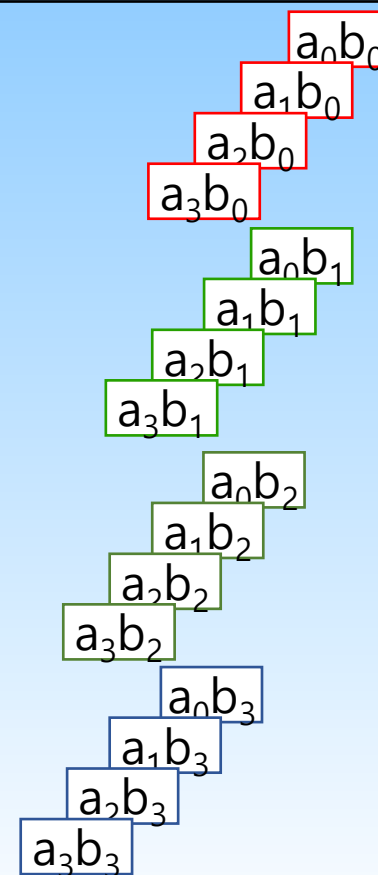
OUTPUT: the product $x \cdot y = (w_{n+t+1} \cdots w_1 w_0)_b$ in radix b representation.

1. For i from 0 to $(n + t + 1)$ do: $w_i \leftarrow 0$.
 2. For i from 0 to t do the following:
 - 2.1 $c \leftarrow 0$.
 - 2.2 For j from 0 to n do the following:

Compute $(uv)_b = w_{i+j} + x_j \cdot y_i + c$, and set $w_{i+j} \leftarrow v$, $c \leftarrow u$.
 - 2.3 $w_{i+n+1} \leftarrow c$.
 3. Return($(w_{n+t+1} \cdots w_1 w_0)$).
-

Multiplication

$$\begin{array}{r} a = a_3 a_2 a_1 a_0 \\ \times b = b_3 b_2 b_1 b_0 \\ \hline \end{array}$$

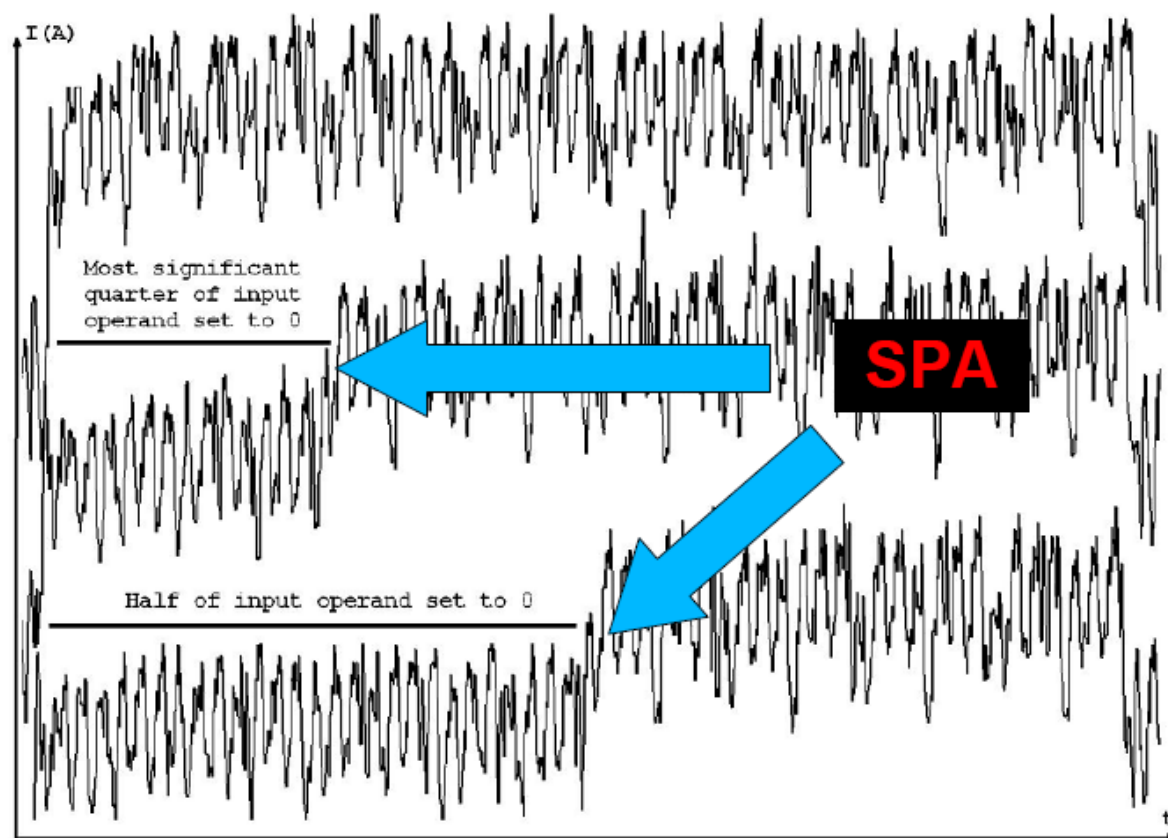


Result

PKC 부채널 분석 및 대응기술



▪ SPA on Multiplication



Square or Mult

*Random values for
Operands*

Mult $A \times B$

Quarter of A set to 0

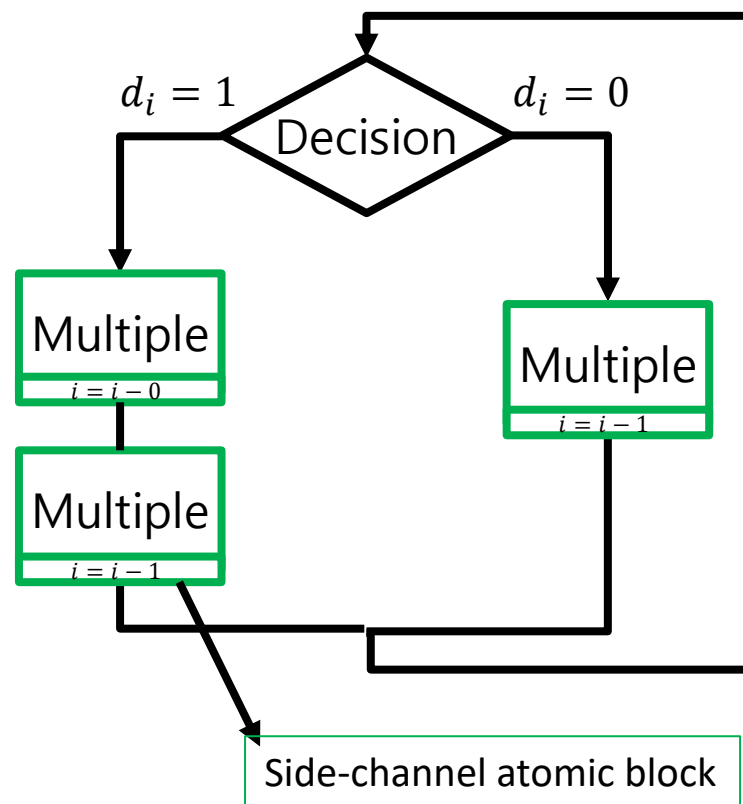
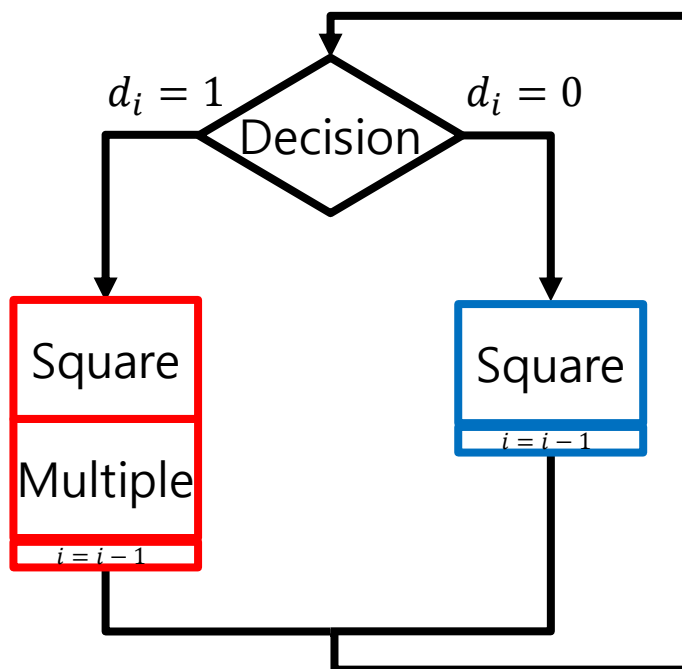
Mult $A \times B$

Half of A set to 0

PKC 부채널 분석 및 대응기술

▪ Timing, SPA에 대한 대응기술 – Atomicity 구현 기법

- Side-channel Atomicity



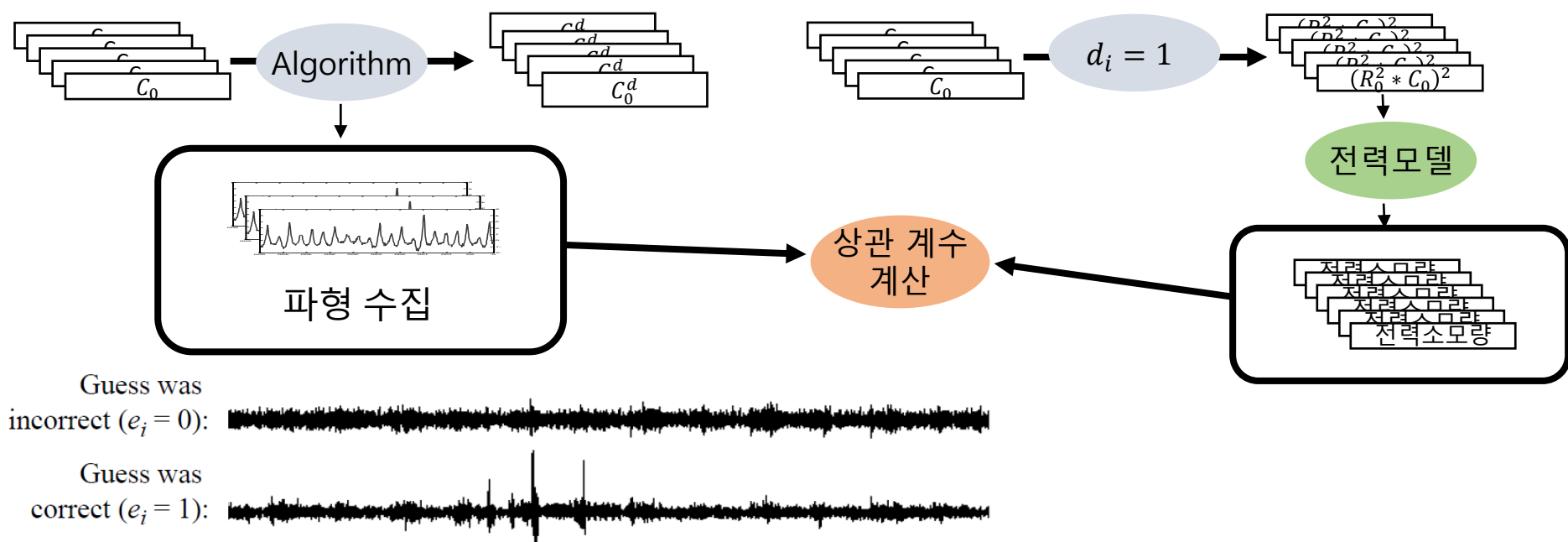
PKC 부채널 분석 및 대응기술



General DPA (Zero Exponent Multiple Data, ZEMD)

- 공격 대상 알고리즘 : Always S&M/D&A RSA/ECC
- 공격 가정 : 다수의 임의 암호문·평문 사용가능
- 공격 개요 : 비밀 키 d 의 bit를 순차적으로 1로 추측 \rightarrow 중간값 C_i 에 대해 $(C_i * M)^2$ 를 계산하여 파형과의 CPA 수행 \rightarrow peak가 발생 한다면 $d_i = 1$ 아니라면 $d_i = 0$ 으로 결정

Algorithm 1 Left-to-right Square-multiply-always algorithm
Require: $M \in G$, $d = (d_{m-1} \dots d_0)_{2,n}$
Ensure: $M^d \bmod n$
 1: $R[2] = 1$
 2: **for** i from $(m-1)$ down to 0 **do**
 3: $R[0] = R[2]^2 \bmod n$
 4: $R[1] = R[0] \times M \bmod n$
 5: $R[2] = R[d_i]$
 6: **return** $R[2]$

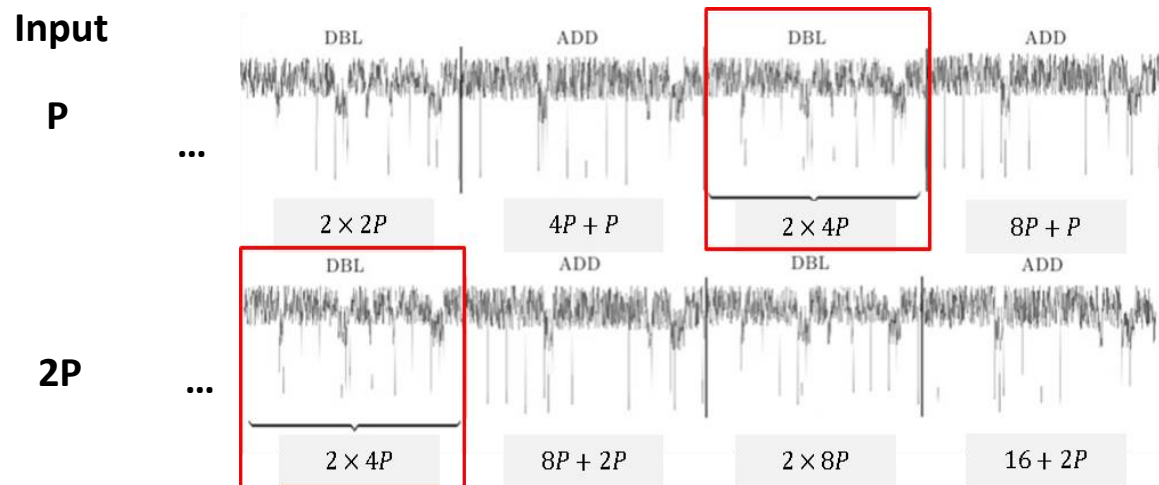


PKC 부채널 분석 및 대응기술



▪ Doubling attack

- 공격 대상 알고리즘 : Always RSA/ECC
- 공격 가정 : 2회 선택 입력문 공격
 - ECC : 타원곡선 위의 좌표 P 와 그 2배 값인 $2P$
 - RSA : 입력 m 과 $m^2 \bmod N$
- 공격 개요 : 2개의 파형들 사이에 충돌 여부를 분석



Input : P

```

S[0] = 0
for i from n down to 0
    S[0] = 2.S[0]
    S[1] = S[0] + P
    S[0] = S[d_i]
return S[0]
    
```

Input : $2P$

```

S[0] = 0
for i from n down to 0
    S[0] = 2.S[0]
    S[1] = S[0] + 2P
    S[0] = S[d_i]
return S[0]
    
```

i	d_i	comput. of dP	comput. of $d(2P)$
6	1	2×0 $0 + P$	2×0 $0 + 2P$
5	0	$2 \times P$ Dummy $2P + P$	$2 \times 2P$ Dummy $4P + 2P$
4	0	$2 \times 2P$ Dummy $4P + P$	$2 \times 4P$ Dummy $8P + 2P$
3	1	$2 \times 4P$ $8P + P$	$2 \times 8P$ $16P + 2P$
2	1	$2 \times 9P$ $18P + P$	$2 \times 18P$ $36P + 2P$
1	1	$2 \times 19P$ $38P + P$	$2 \times 38P$ $76P + 2P$
0	0	Dummy $2 \times 39P$ Dummy $78P + P$ return $78P$	Dummy $2 \times 78P$ Dummy $156P + 2P$ return $156P$

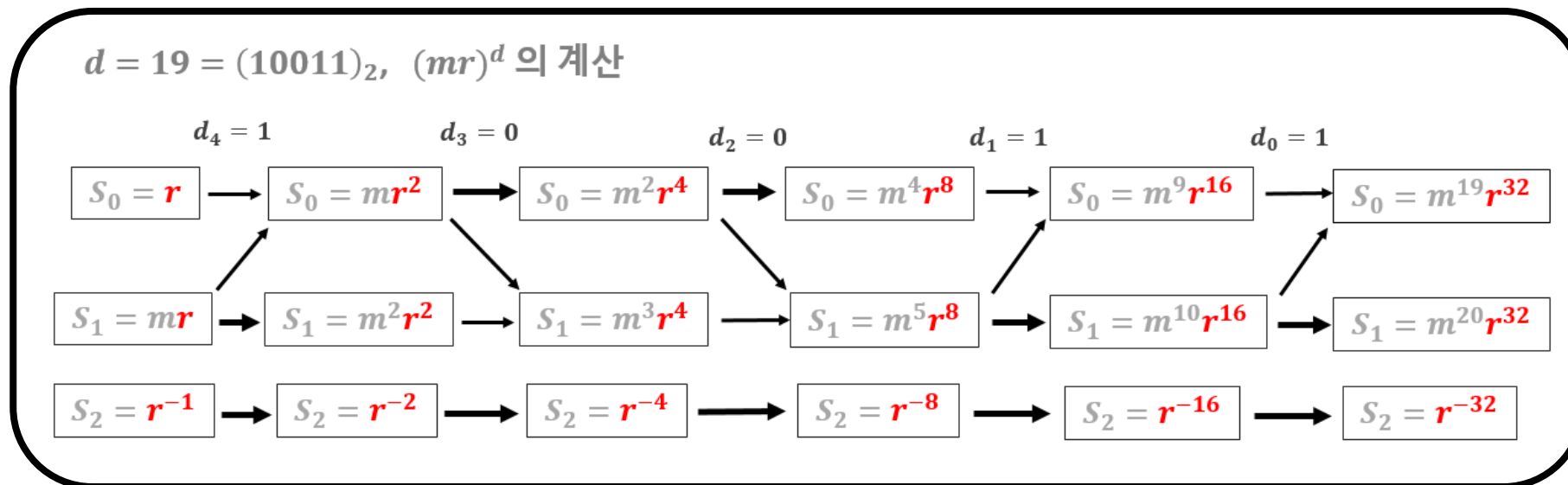
PKC 부채널 분석 및 대응기술

▪ DPA, Doubling Attack에 대한 대응기술

- 메시지 블라인딩 : Masking과 같이, 입력 값에 난수를 부여하여 중간값 추측을 불가능하게 함
- 지수 블라인딩 : 지수를 랜덤한 수로 대체하여 연산 수행 (예: $m^d = m^{d+r*\phi(n)}$)

▪ RSAES/RSA-PSS/KCDSA에 대한 Blinding 기법 적용

- Montgomery powering ladder 알고리즘에 난수 r 을 추가한 연산 수행



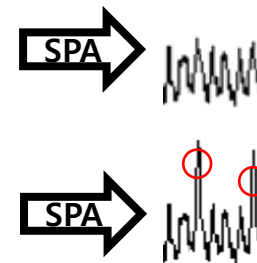
Yen's attack

- 공격 대상 알고리즘 : Always Non/CRT-RSA, ECC
- 공격 가정 : 단일/소수의 선택 입력문 공격, RSA의 공개된 모듈러 n 에 대하여 $(n-1)$ 을 입력 값으로 사용
 - 공격자는 $(1)^2 \bmod n$ 이 연산되는 파형과 $(n-1)^2 \bmod n$ 이 연산되는 파형을 SPA로 구분 할 수 있음
- 공격 개요 : Loop i 번째에 제공 연산의 전력 파형을 관측하여 d_{i-1} 을 추측 가능

Algorithm 1 Left-to-right Square-multiply-always algorithm
Require: $M \in G$, $d = (d_{m-1} \dots d_0)_2, n$
Ensure: $M^d \bmod n$
1: $R[2] = 1$
2: **for** i **from** $(m-1)$ **down to** 0 **do**
3: $R[0] = R[2]^2 \bmod n$
4: $R[1] = R[0] \times M \bmod n$
5: $R[2] = R[d_i]$
6: **return** $R[2]$

ex) Left – to – Right exponentiation

	$d_0 = 1$...	Guess $d_{i-1} = 0$	d_i
Line 3	$(1)^2 \bmod n = 1$...	$(n-1)^2 \bmod n = 1$	$(1)^2 \bmod n$
Line 4	$1(n-1) \bmod n = n-1$...	$1(n-1) \bmod n = n-1_{(Dummy)}$	
	d_0	...	Guess $d_{i-1} = 1$	d_i
Line 3	$(1)^2 \bmod n = 1$...	$(n-1)^2 \bmod n = 1$	$(n-1)^2 \bmod n$
Line 4	$1(n-1) \bmod n = n-1$...	$1(n-1) \bmod n = n-1$	



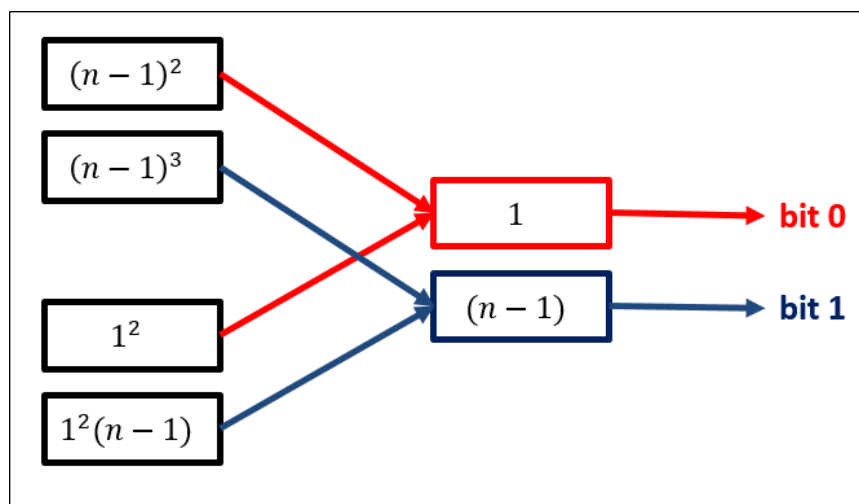
PKC 부채널 분석 및 대응기술

Yen's Attack에 대한 대응기술 – 입력 값 점검을 통해 대응 가능

- $n-1(-1)$ 입력값에 대해서는 연산 수행 없이 결과값을 리턴

RSAES/RSA-PSS에 대한 조건문 추가

- 공개 모듈러 n 에 대하여 입력 값이 $(n-1) \bmod n$ 일 경우, $(n-1)^d \bmod n = (n-1) \bmod n$
- RSA 개인키 지수 d 는 항상 홀수이므로 연산 결과는 언제나 최하위 bit 1에 대한 결과값 $n-1$

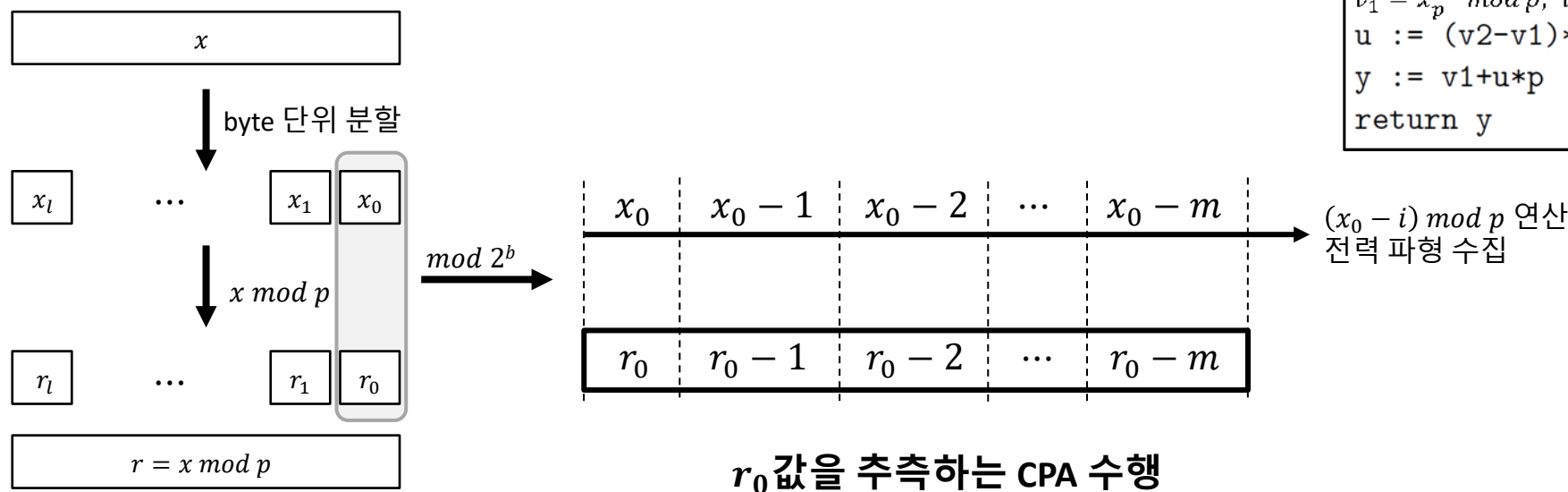


PKC 부채널 분석 및 대응기술

▪ CRT-CPA (Modular Reduction using Equidistant Data, MRED)

- 공격 대상 알고리즘 : CRT-RSA의 소수 모듈러 감산
- 공격 가정 : 다수의 선택 입력문 공격
- 공격 개요 : 연속적인 입력 값 x 을 이용하여 $(x \bmod p)$ 연산을 대상으로의 나머지 r 를 CPA로 복원

→ $r = x \bmod p$ 의 관계를 이용하여 소수 p 를 복원 ($\gcd(x-r, n)=p$)



```

 $x_p = x \bmod p, x_q = x \bmod q$ 
 $v_1 = x_p^{d_p} \bmod p, v_2 = x_q^{d_q} \bmod q$ 
 $u := (v_2 - v_1) * Pq \bmod q$ 
 $y := v_1 + u * p$ 
return y

```

PKC 부채널 분석 및 대응기술

▪ CRT-CPA에 대한 대응기술 – CRT 블라인딩 기법

- CRT 연산에 대해 감산 연산 이전 입력 값에 대한 블라인딩을 적용
- 이후 CRT 재조합 연산이 끝나기까지 중간 과정에서 블라인딩이 제거되지 않도록 유지

▪ RSA 전체 연산에 대한 CRT Blinding 기법 적용

- CRT 연산에 대한 입력 값에 난수 블라인딩을 적용할 경우, 나머지 연산 중간값을 추측할 수 없음
- 해당 난수 블라인딩이 적용된 상태로 각 소수 모듈러에 대한 지수승 연산과 CRT 결합 연산을 모두 수행

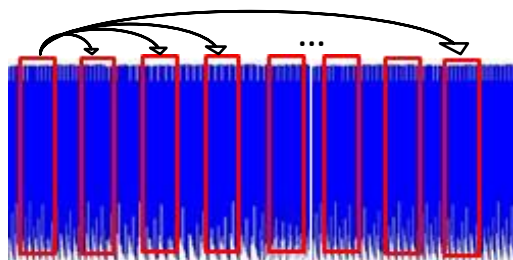
Input : c, n ($=pq$, k -bit modulus), $d(=(d_{n-1}d_{n-2}\dots d_0)_2)$, $d_p = d \bmod (p-1)$,
 $d_q = d \bmod (q-1)$, $\hat{q} = q^{-1} \bmod p$
 Output : $C^d \bmod n$

1. Generate k -bit random number r
2. $R = r^{-1} \bmod n$
3. $\tilde{C}_p = Cr \bmod p$, $\tilde{C}_q = Cr \bmod q$, $r_p = r \bmod p$, $r_q = r \bmod q$
4. $m_p = \text{Algorithm5-1}(\tilde{C}_p, r_p, d_p, p)$
5. $m_q = \text{Algorithm5-1}(\tilde{C}_q, r_q, d_q, q)$
6. $\tilde{m} = m_p + q((m_p - m_q)\hat{q} \bmod p)$
7. Return $\tilde{m}R \bmod n$

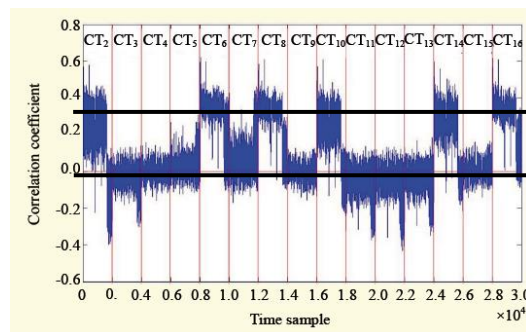
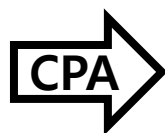
PKC 부채널 분석 및 대응기술

2nd order DPA

- 공격 대상 알고리즘 : RSA (BRIP(Binary Randomized Initial Point) 적용)
- 공격 가정 : 다수의 임의 암호문·복호문 사용가능
- 공격 개요 : 첫 번째 곱셈 연산의 파형을 참조 파형으로 설정하고, 나머지 곱셈 연산들의 파형 사이에 CPA 수행 → CPA 결과를 바탕으로 2 그룹으로 분류
→ 두 그룹 중 어떤 그룹이 지수 bit가 1인지 암호문·복호문 쌍으로 확인



- ① 첫 번째 곱셈 연산의 부분 파형을 Reference 파형으로 설정
- ② Reference 파형과 나머지 파형들의 상관관계를 계산



Algorithm 3. BRIP algorithm for RSA

Input: X, m , and $d = (d_{n-1}d_{n-2} \cdots d_1d_0)_2$ where n is the size of modulus m .

Output: $X^d \bmod m$.

1. If $X = 1$ then return 1. Else if $X = -1$ then return $1 - 2d_0$.
2. Generate a random value v .
3. Compute $U = R^v \bmod m$ and $U_0 = (\text{Mont}(1, 1))^v = R^v \bmod m$ using SPA resistant exponentiation.
4. Compute $U = \text{Mont}(U, R^2)$.
5. Compute $U_0 = \text{Mont}(U_0, R^2)$ and $U_1 = \text{Mont}(\text{Mont}(X, R^2), U_0)$.
6. For $i = n-1$ down to 0 do.
 - 6.1. $U = \text{Mont}(U, U)$.
 - 6.2. $U = \text{Mont}(U, U_{d_i})$.
7. $U = \text{Mont}(U, U_0)$
8. Return $\text{Mont}(U, 1)$.

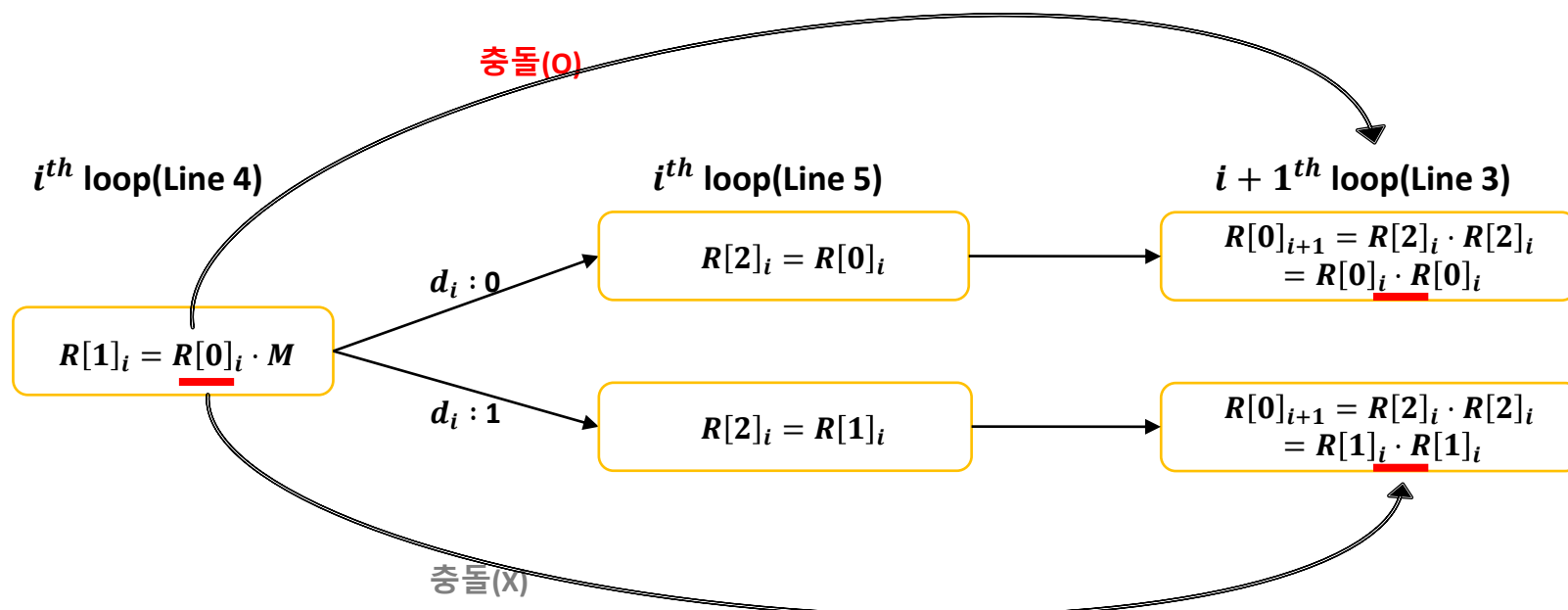
평균, 암호문 쌍을 이용하여
어떤 그룹이 0 또는 1인지 판별

PKC 부채널 분석 및 대응기술

■ HCCA (Horizontal Collision Correlation Attack)

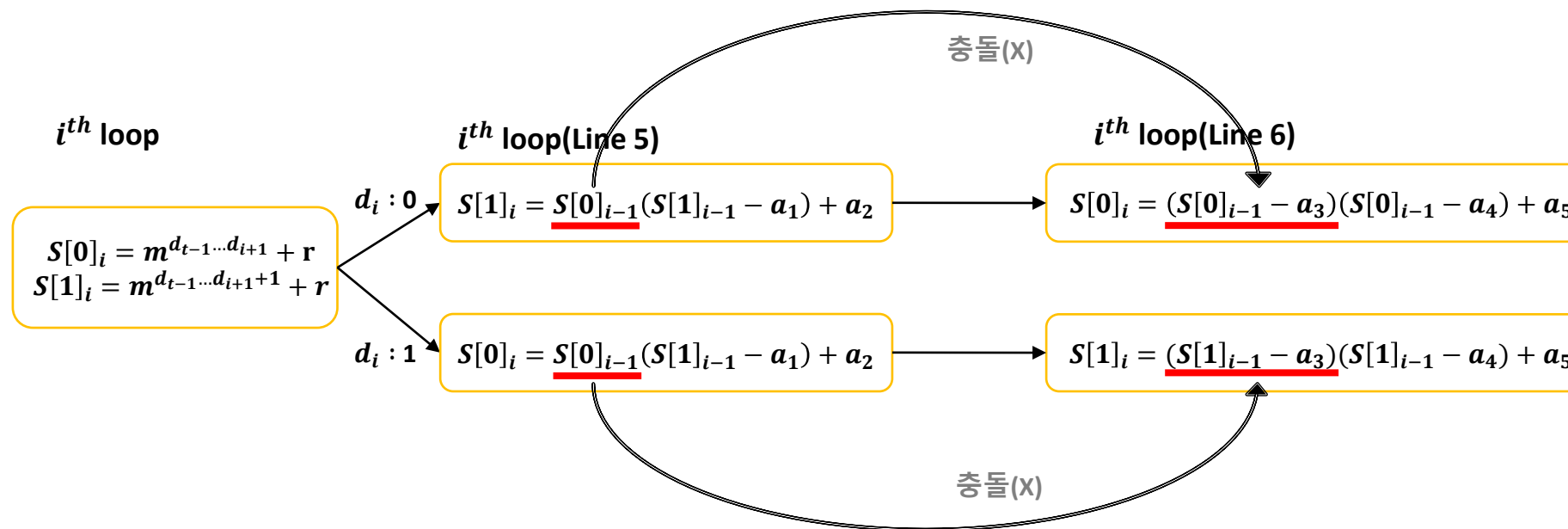
- 공격 대상 알고리즘 : always Non/CRT-RSA, ECC
- 공격 가정 : 단일 파형 공격 (암호문·복호문 정보 불필요)
- 공격 개요 : 연속한 곱셈 연산 간의 충돌 확인 → 충돌의 발생여부에 따라 지수 bit 구분 가능
 - $d_i : 0 \rightarrow R[2] = R[0]$ 을 수행하므로 i^{th} 의 $R[0]$ 과 $i + 1^{th}$ 의 $R[0]$ 와의 충돌 o
 - $d_i : 1 \rightarrow R[2] = R[1]$ 을 수행하므로 i^{th} 의 $R[0]$ 과 $i + 1^{th}$ 의 $R[1]$ 와의 충돌 x

Algorithm 1 Left-to-right Square-multiply-always algorithm
Require: $M \in G$, $d = (d_{m-1} \dots d_0)_2, n$
Ensure: $M^d \bmod n$
 1: $R[2] = 1$
 2: **for** i **from** $(m-1)$ **down to** 0 **do**
 3: $R[0] = R[2]^2 \bmod n$
 4: $R[1] = R[0] \times M \bmod n$
 5: $R[2] = R[d_i]$
 6: **return** $R[2]$



PKC 부채널 분석 및 대응기술

- HOCPA, HCCA에 대한 대응기술 – 덧셈 블라인딩 기법 + 연산 순서 조정
 - 비트 값에 의존해 곱셈 연산의 피연산자 값 사이에 나타날 수 있는 충돌 특성을 제거해야 함
- RSAES/RSA-PSS/KCDSA에 대한 덧셈 Blinding 기법 적용
 - 피연산자 값에 의한 연산적 특성이 드러나지 않도록 사전 연산 값을 통해 지수승 연산 수행



PQC 부채널 분석 및 대응기술



IA&AI SecLab
Implementation Attacks & AI Security Laboratory

NIST PQC Standardization 요구사항

▪ Security

- Against both classical and quantum attacks

▪ Performance

- Measured on various classical platforms

▪ Other properties

- Drop-in replacements
- Perfect forward secrecy
- Resistance to side-channel attacks
- Simplicity and flexibility
- etc ...

NIST PQC 부채널 분석 동향

▪ PQC 부채널 분석 동향

- PQC 부채널 분석 관련 대부분의 공격기술 연구가 Power Attack에 초점이 맞춰져 있는데 반해 대응기술 연구는 시간공격(Timing Attack)에 대한 Constant Time 구현에 편중되어 있음

부채널 분석 연구 현황			
		PQC	Others
Attacks	Timing Attack	23%	21%
	Power Attack	63%	62%
	Fault Attack	14%	17%
Counter-measures	Mask	38%	69%
	Shuffle	16%	15%
	Constant Time	46%	16%

Kyber 대상 부채널 공격

▪ KEM Decapsulation 중 PKE Decryption 부분에 대한 메시지 복구 공격

- Decryption 부분의 메시지를 복구하면(A) Session Key를 복구하는 것이 가능(B)
- PKE Decryption의 메시지를 복구했다는 것은 곧 IND-CCA가 깨졌음을 의미
→ Kyber에 대한 선택 암호문 공격으로 Secret Key 복구 가능

Algorithm 2: FO transform of a IND-CPA secure PKE into a IND-CCA secure KEM

```

1 Procedure KEM.Encaps(pk)
2    $\rho \leftarrow \mathcal{U}(\mathcal{B}^{32})$ 
3    $m = \mathcal{H}(\rho)$ 
4    $r = \mathcal{G}(m, pk)$ 
5    $ct = \text{PKE.Encrypt}(pk, m, r)$ 
6    $K = \mathcal{H}(r, ct)$ 
7   return (ct, K)
8
1 Procedure KEM.Decaps(sk, pk, ct)
2   A  $m' = \text{PKE.Decrypt}(sk, ct)$ 
3    $r' = \text{PRF}(m', pk)$ 
4    $ct' = \text{PKE.Encrypt}(pk, m', r')$ 
5   if  $ct' = ct$  then
6     B  $\text{return } K = \text{KDF}(r' || ct')$ 
7   end
8   else
9     return  $K = \text{KDF}(z || ct')$  //  $z \in \mathcal{B}^{32}$  is a random secret
10  end

```

Kyber 대상 부채널 공격

```

1 void indcpa_dec(uint8_t m[KYBER_INDCPA_MSGBYTES],
2               const uint8_t c[KYBER_INDCPA_BYTES],
3               const uint8_t sk[KYBER_INDCPA_SECRETKEYBYTES])
4 {
5     polyvec bp, skpv;
6     poly v, mp;
7
8     unpack_ciphertext(&bp, &v, c);
9     unpack_sk(&skpv, sk);
10
11     polyvec_ntt(&bp);
12     polyvec_pointwise_acc_mont(&v, &bp, &skpv);
13     poly_invntt_tomont(&mp);
14
15     poly_sub(&mp, &v, &mp);
16     poly_reduce(&mp);
17
18     poly_tomsg(m, &mp);
19 }

```

m[KYBER_INDCPA_MSGBYTES]

복호화 메시지를 담을 32byte 공간

c[KYBER_INDCPA_BYTES]

암호문 $2 \times 320 + 160$ byte (KYBER512 기준)

sk[KYBER_INDCPA_SECRETKEYBYTES]

비밀키 2×384 byte (KYBER512 기준)

1byte 변수공간 msg[i]에
복호화한 메시지를 1bit씩
8번 반복해서 담음 (1bit leakage)

```

1 void poly_tomsg(uint8_t m[KYBER_INDCPA_MSGBYTES],
2               const polyvec v)
3 {
4     unsigned int i;
5     uint16_t t;
6
7     poly_csubq(a);
8
9     for(i=0; i<KYBER_INDCPA_MSGBYTES; i++) {
10         /* init byte m[i] */
11         msg[i] = 0;
12         for(j=0; j<8; j++) {
13             /* Calculate Message Bit */
14             t = (((uint16_t)v->coeffs[8*i+j] << 1) + KYBER_Q/2)/KYBER_Q & 1;
15             /* Bit Update in Memory */
16             msg[i] |= t << j;
17         }
18     }
19 }

```

Kyber 대상 부채널 공격

```

1 void indcpa_dec(uint8_t m[KYBER_INDCPA_MSGBYTES],
2               const uint8_t c[KYBER_INDCPA_BYTES],
3               const uint8_t sk[KYBER_INDCPA_SECRETKEYBYTES])
4 {
5     polyvec bp, skpv;
6     poly v, mp;
7
8     unpack_ciphertext(&bp, &v, c);
9     unpack_sk(&skpv, sk);
10
11     polyvec_ntt(&bp);
12     polyvec_pointwise_acc_mont(&v, &bp, &skpv);
13     poly_invntt_tomont(&mp);
14
15     poly_sub(&mp, &v, &mp);
16     poly_reduce(&mp);
17
18     poly_tomsg(m, &mp);
19 }

```

j번째 for문에서
msg[i]가 가질 수 있는
해밍웨이트에 대한 template 생성

0th template,
1st template,
2nd template,

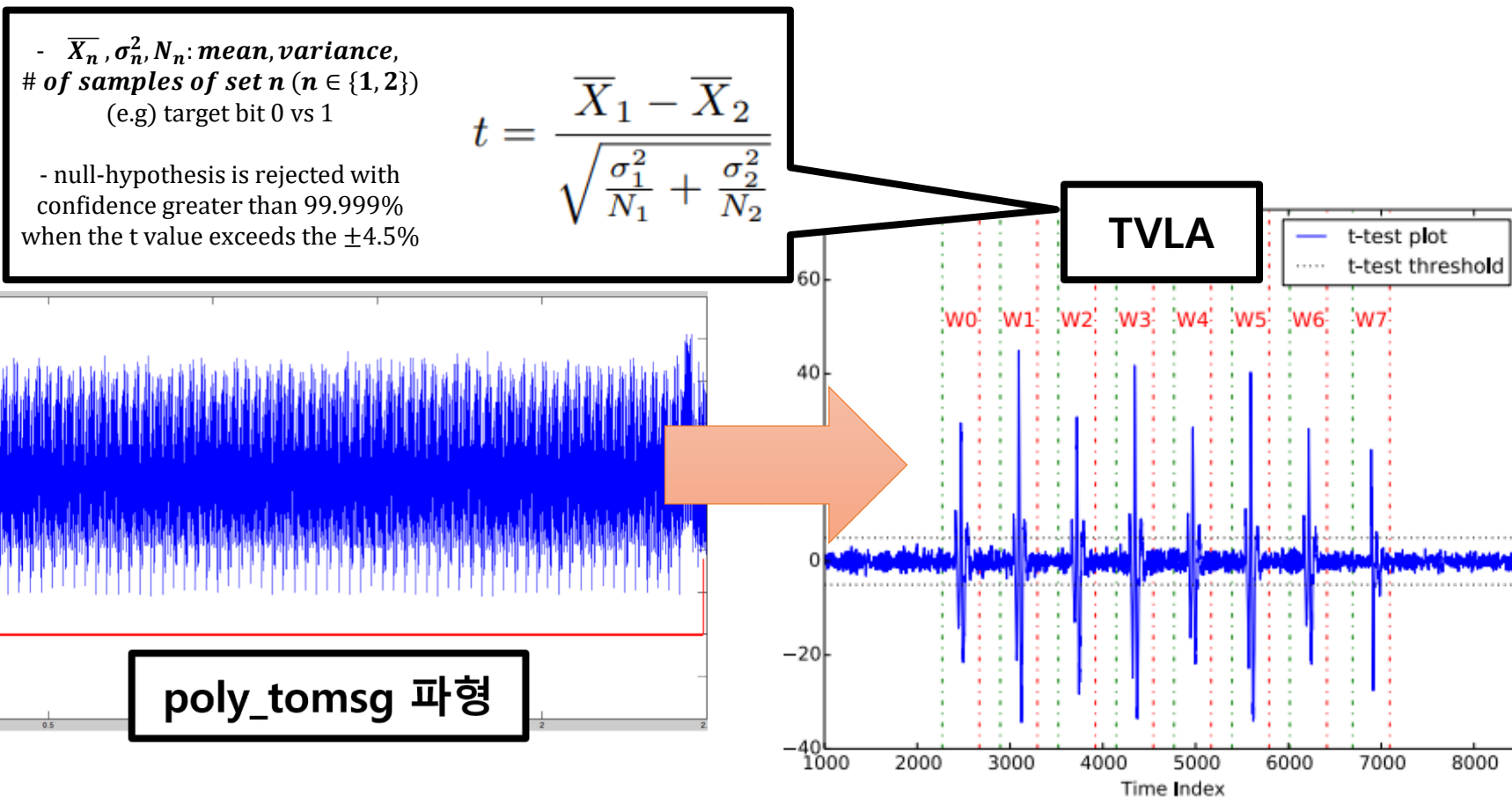
...
(j+1)th template

```

1 void poly_tomsg(
2 {
3     unsigned int i;
4     uint16_t t;
5
6     poly_csubq(a);
7
8     for(i=0; i<KYBER_INDCPA_BYTES; i++) {
9
10        /* init byte m[i] */
11        msg[i] = 0;
12
13        for(j=0; j<8; j++) {
14            /* Calculate Message Bit */
15            t = (((uint16_t)a->coeffs[8*i+j] << 1) + KYBER_Q/2)/KYBER_Q & 1;
16            /* Bit Update in Memory */
17            msg[i] |= t << j;
18        }
19    }
20 }

```

Kyber 대상 부채널 공격

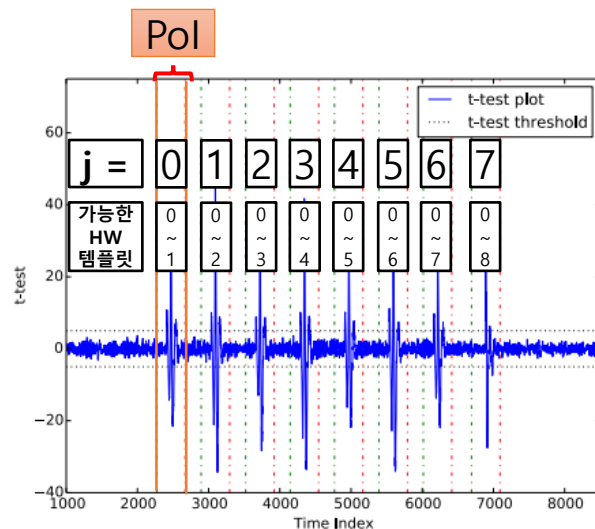


참조 문헌 : eprint 2020 “On exploiting message leakage in (few) nist pqc candidates for practical message recovery and key recovery attacks”

Kyber 대상 부채널 공격

– Pre-Processing 단계

1. 해밍웨이트에 대한 템플릿 생성을 위한 Poi 선택



2. Poi에 해당하는 파형들의 평균을 내어 해밍웨이트에 대한 템플릿 생성

– Attack 단계

3. 공격 대상 파형에 대해 복구하고 싶은 메시지 비트를 대상으로 해당 파형과 템플릿과의 거리를 계산

$$I[k] = (tr' - rt_{(i,j)}^k)^T (tr' - rt_{(i,j)}^k)$$

4. 가장 가까운 거리인 템플릿 선택

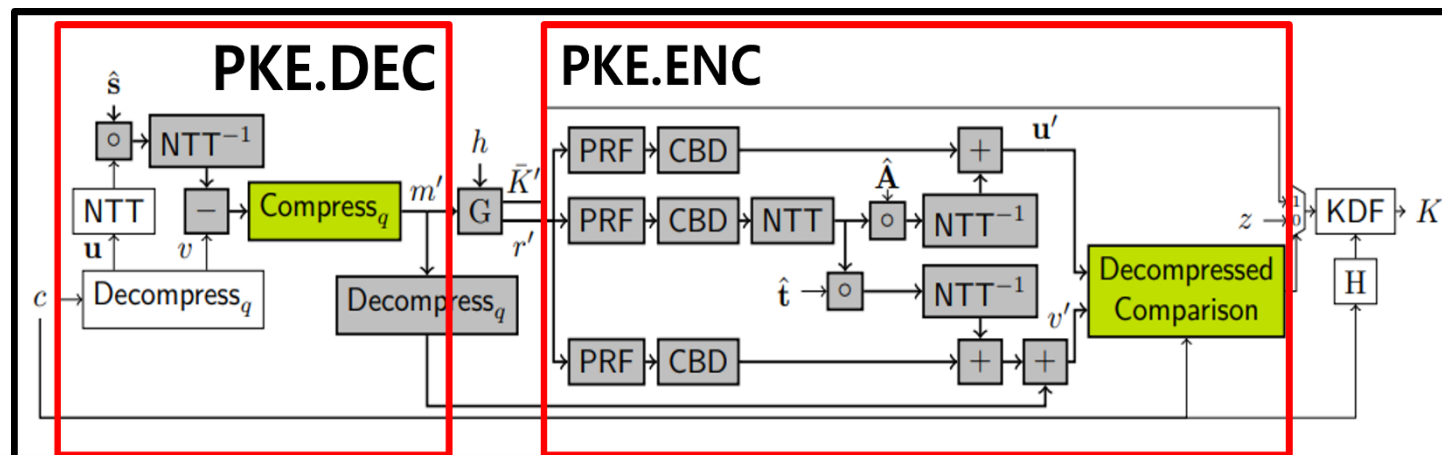
➔ 해당 템플릿을 통해 메시지 비트 복원

5. 8번 반복 후 메시지 복원 가능

Kyber 대상 부채널 공격 대응기술

▪ Kaber Decapsulation에 대한 Higher-order 마스킹 기법

- 비밀키 s 가 연산에 포함된 지점들(회색 영역)에서 비밀 정보가 누출될 가능성이 있음
- 비밀키 s 를 arithmetic masking하여 s_1, s_2, \dots, s_n 로 나눈 뒤 각각 디캡슐화 과정에 입력
- A2B / B2A 알고리즘을 이용하여 산술 연산과 비트 연산 각각 맞게 마스킹 변환



```

1 Procedure KEM.Decaps(sk, pk, ct)
2    $m' = \text{PKE.Decrypt}(sk, ct)$ 
3    $r' = \text{PRF}(m', pk)$ 
4    $ct' = \text{PKE.Encrypt}(pk, m', r')$ 
5   if  $ct' = ct$  then
6     return  $K = \text{KDF}(r' || ct')$ 
7   end

```

```

8   else
9     return  $K = \text{KDF}(z || ct') // z \in \mathcal{B}^{32}$ 
10  end

```

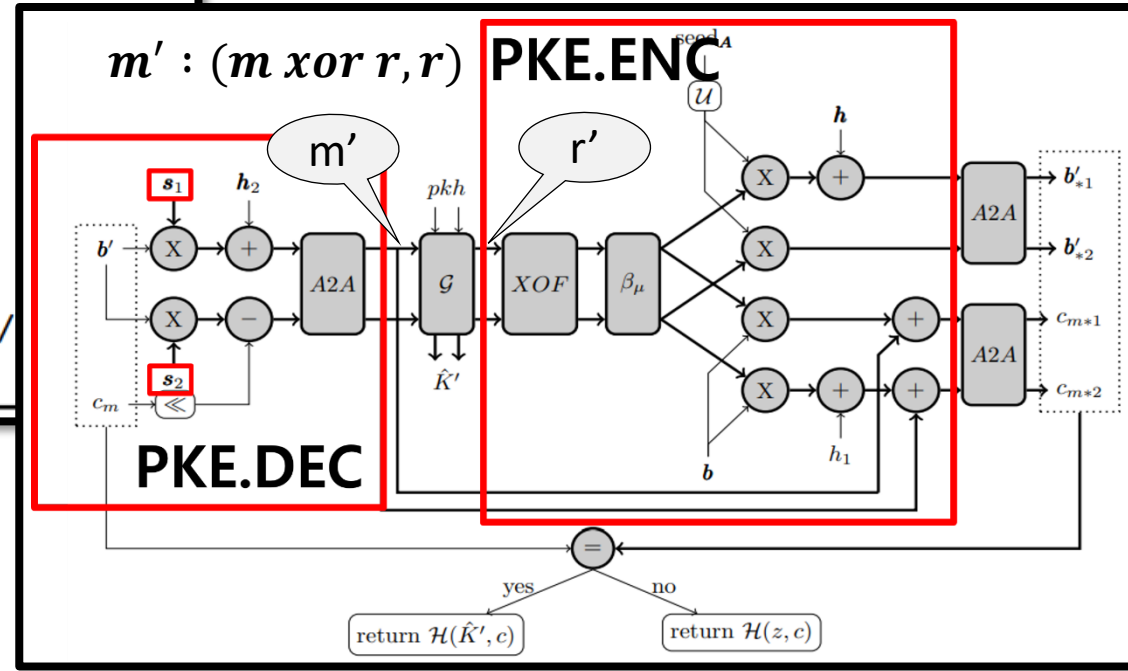
Saber 대상 부채널 공격 대응기술

▪ Saber에 대한 디캡슐화 1차 마스킹 기법

- 비밀키 s 가 연산에 포함된 지점들(회색 영역)에서 비밀 정보가 누출될 가능성이 있음
- 비밀키 s 를 arithmetic masking하여 s_1, s_2 로 나눈 뒤($s = s_1 + s_2$) 각각 디캡슐화 과정에 입력
- A2B / B2A 알고리즘을 이용하여 산술 연산과 비트 연산 각각 맞게 마스킹 변환

```

1 Procedure KEM.Decaps(sk, pk, ct)
2    $m' = \text{PKE.Decrypt}(sk, ct)$ 
3    $r' = \text{PRF}(m', pk)$ 
4    $ct' = \text{PKE.Encrypt}(pk, m', r')$ 
5   if  $ct' = ct$  then
6     return  $K = \text{KDF}(r' || ct')$ 
7   end
8   else
9     return  $K = \text{KDF}(z || ct')$  //
10  end
    
```



Saber 대상 부채널 공격

▪ KEM Decapsulation 중 PKE Decryption 부분에 대한 메시지 복구 공격

- 앞선 Kyber에 대한 Template 공격과 전반적인 과정 동일, 학습 네트워크를 이용

```

1 void indcpa_kem_dec(const uint8_t sk[SABER_INDCPA_SECRETKEYBYTES],
2                   const uint8_t ciphertext[SABER_BYTES_CCA_DEC],
3                   uint8_t m[SABER_KEYBYTES])
4 {
5
6     uint16_t s[SABER_L][SABER_N];
7     uint16_t b[SABER_L][SABER_N];
8     uint16_t v[SABER_N] = {0};
9     uint16_t cm[SABER_N];
10    int i;
11
12    BS2POLVECq(sk, s);
13    BS2POLVECP(ciphertext, b);
14    InnerProd(b, s, v);
15    BS2POLT(ciphertext, cm);
16
17    for (i = 0; i < SABER_N; i++)
18    {
19        v[i] = (v[i] + cm[i]);
20    }
21
22    POLmsg2BS(m, v);
23 }

```

`sk[SABER_INDCPA_SECRETKEYBYTES]`
비밀키 $3 \times 8 \times 32 + 4 \times 32$ byte (Saber 기준)

`ciphertext[SABER_BYTES_CCA_DEC]`
암호문 $3 \times 10 \times 32 + 1$ byte (Saber 기준)

`m[SABER_INDCPA_KEYBYTES]`
복호화된 메시지를 담을 32 byte 공간

1byte 변수공간 `bytes[i]`에
복호화한 메시지를 1bit씩
8번 반복해서 담음 (1bit leakage)

```

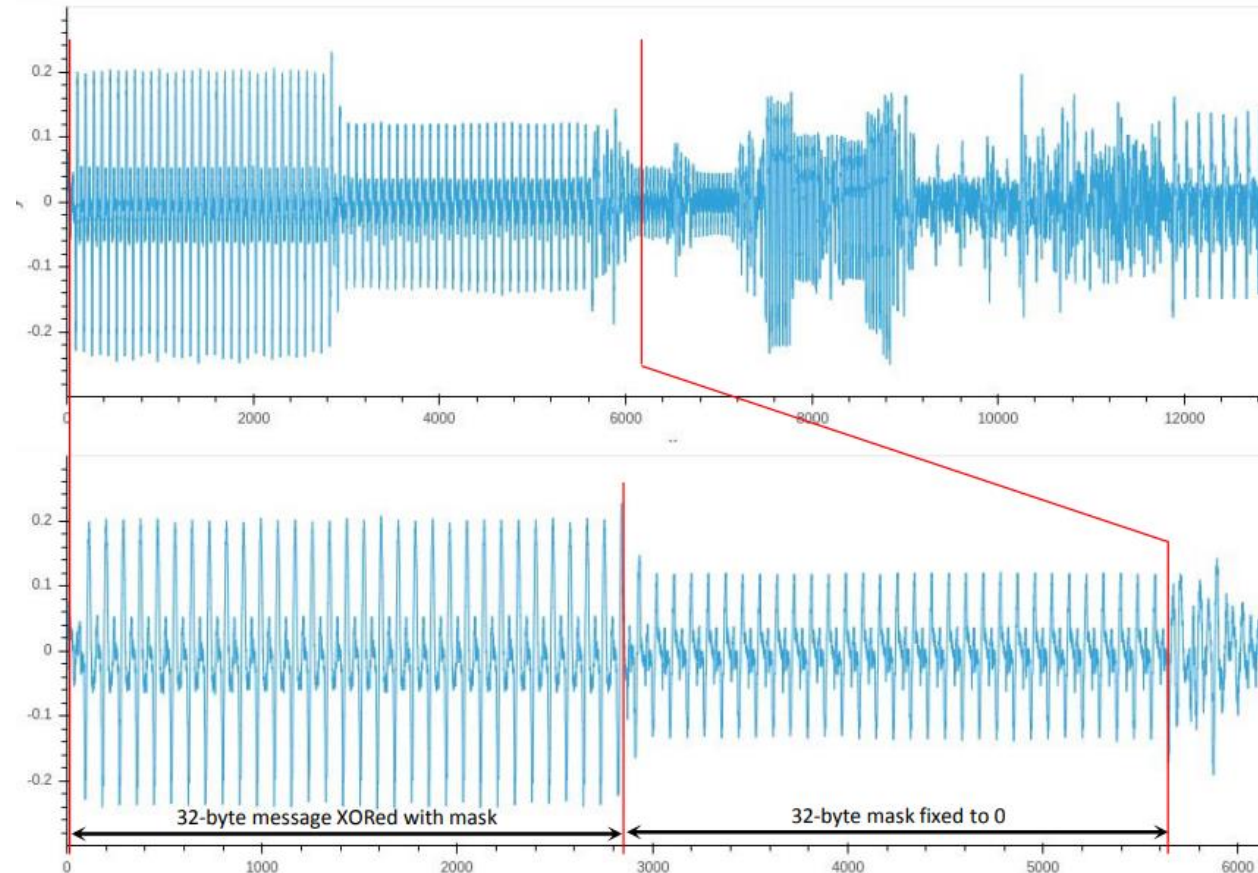
1 void POLmsg2BS(uint8_t m, uint16_t v)
2 {
3     size_t i, j;
4     memset(bytes, 0, SABER_KEYBYTES);
5
6     for (j = 0; j < SABER_KEYBYTES; j++)
7     {
8         for (i = 0; i < 8; i++)
9         {
10            bytes[j] = bytes[j] | ((v[j * 8 + i] & 0x01) << i);
11        }
12    }
13 }

```

Saber 대상 부채널 공격

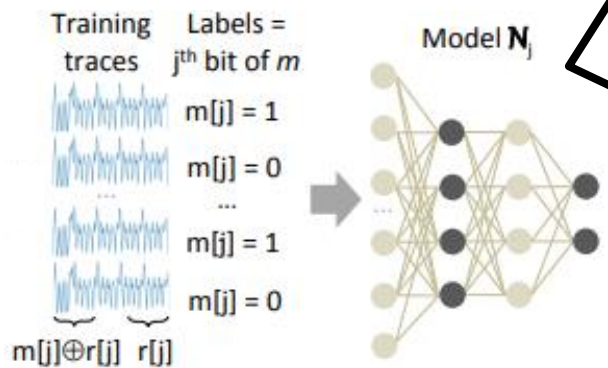
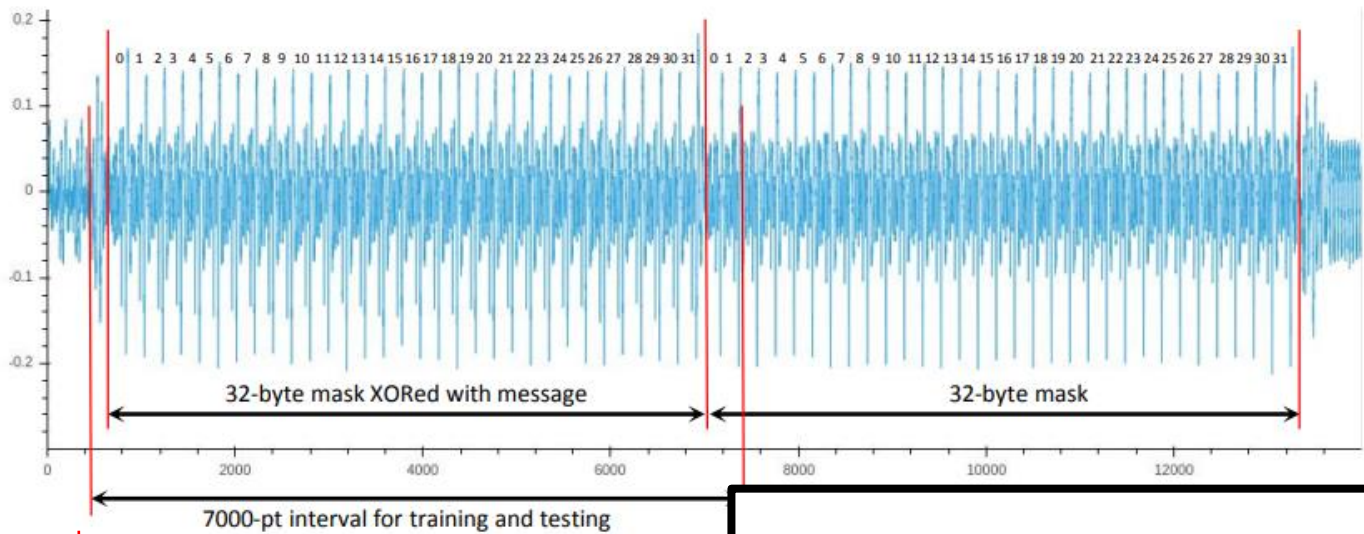


- SPA를 통해 마스킹된 POLmsg2BS의 위치와 모양을 확인



Saber 대상 부채널 공격

- 해당 구간의 파형을 입력으로 메시지 비트를 분류하는 학습 네트워크 구성



입력값 : 마스크된 $m \oplus r$ 값과 마스크 r 값이 모두 포함된 구간의 파형
라벨 : j번째 메시지 비트로 라벨링

참조 문헌 : TCHES 2021 “A side-channel attack on a masked IND-CCA secure Saber KEM”

Saber 대상 부채널 공격

- 실험 결과



D_1, D_2
from same chip vender

D_3
from different chip vender

p_j : 1000개 파형 test set에 대한 메시지 비트 $m[j]$ 의 복구율을 Device 별로 나타낸 것임

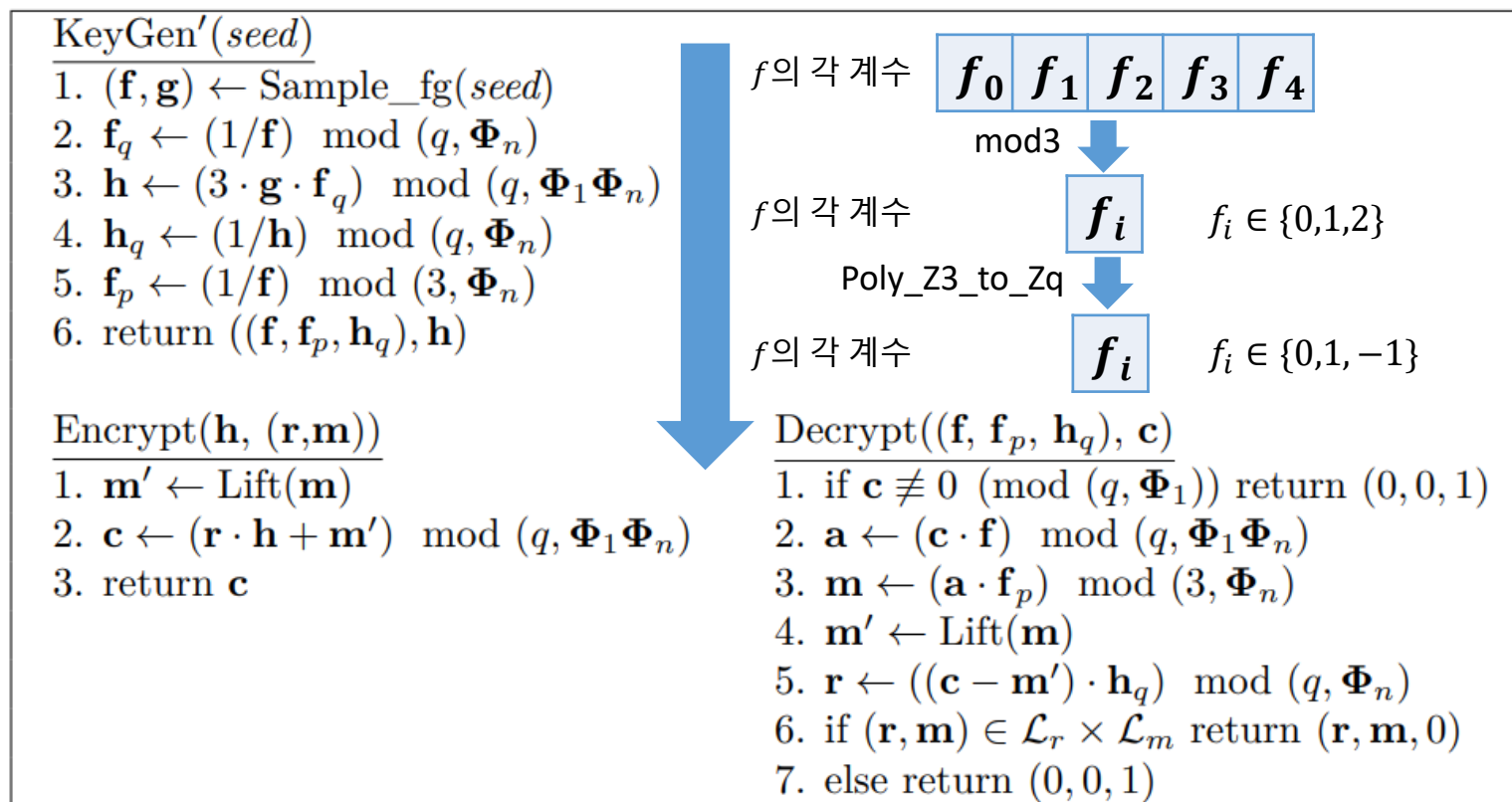
Device	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	average
D_1	0.998	0.998	0.993	0.992	0.989	0.988	0.985	0.953	0.987
D_2	0.994	0.989	0.986	0.959	0.978	0.962	0.985	0.945	0.975
D_3	0.984	0.985	0.988	0.963	0.972	0.991	0.975	0.819	0.960
average	0.992	0.990	0.989	0.971	0.979	0.980	0.982	0.906	0.974

참조 문헌 : TCHES 2021 “A side-channel attack on a masked IND-CCA secure Saber KEM”

NTRU 대상 부채널 공격

■ 변수의 해밍 웨이트값을 이용한 SPA 공격

- NTRU의 Decrypt에서의 비밀키 f 의 디코딩 과정 및 $\text{mod } q$ 연산을 위한 $\text{mod}3$, poly_Z3_to_Zq 알고리즘 공격



NTRU 대상 부채널 공격

■ 해밍 웨이트의 차이가 큰 두 집합은 파형을 통해 구분 가능

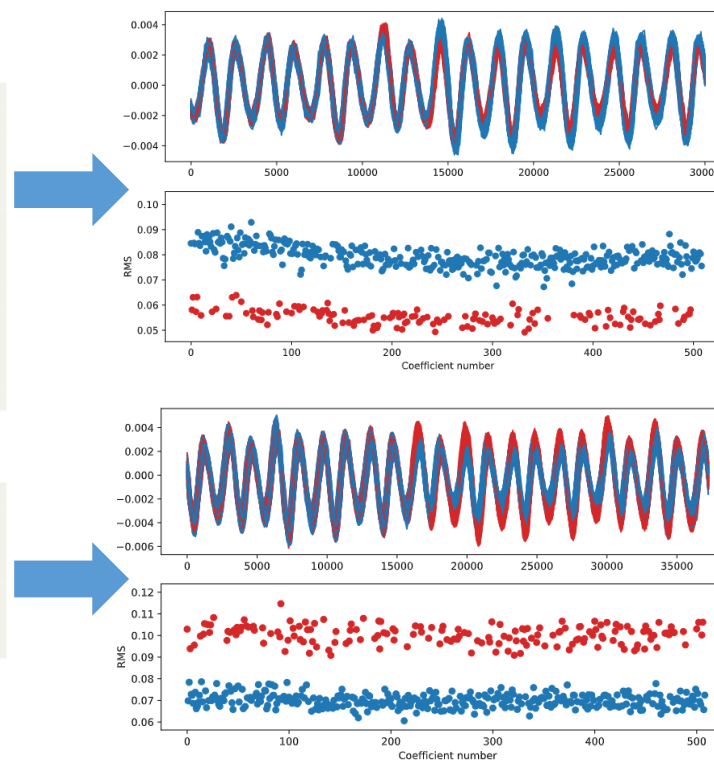
- Mod3, poly_Z3_to_Zq 알고리즘의 입력 값은 각각 비밀키 f 의 계수와 비밀키 f 가 들어감
- $\{-3, -2, -1\}$ 들은 해밍 웨이트가 15이상, $\{0, 1\}$ 들은 해밍 웨이트가 1이하
- mod3 : t 값에 따라 $\{-3, -2, -1\}$ 과 $\{0, 1\}$ 로 구분
- poly_Z3_to_Zq : r 의 계수에 따라 $\{0, 1\}$ 과 $\{-1\}$ 로 구분

```
1 static uint16_t mod3(uint16_t a) {
2     uint16_t r;
3     int16_t t, c;
4
5     r = (a >> 8) + (a & 0xff);
6     r = (r >> 4) + (r & 0xf);
7     r = (r >> 2) + (r & 0x3);
8     r = (r >> 2) + (r & 0x3);
9
10    t = r - 3;
11    c = t >> 15;
12    return (c&r) ^ (~c&t);
13 }
```

Listing 1: mod3() implementation

```
1 /* Map {0, 1, 2} -> {0,1,q-1} in place */
2 void poly_Z3_to_Zq(poly *r) {
3     int i;
4     for(i=0; i<NTRU_N; i++) {
5         r->coeffs[i] = r->coeffs[i] | (((r->coeffs[i]>>1)) & (NTRU_Q-1));
6     }
7 }
```

Listing 2: poly_Z3_to_Zq() implementation

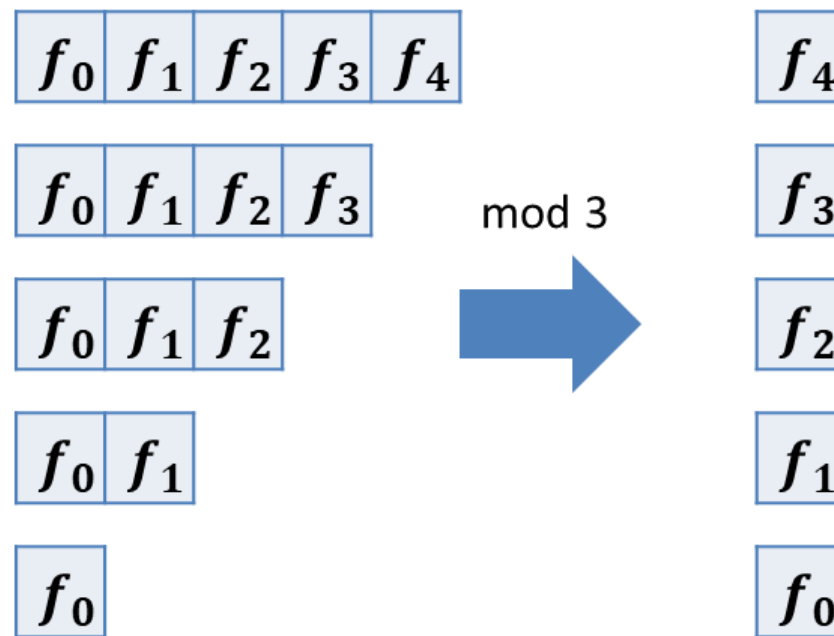


NTRU 대상 부채널 공격

■ 그 이후 공격

- f 의 계수가 -1인 경우는 poly_Z3_to_Zq 을 통해 알 수 있음
- $\text{mod}3$ 의 t 값이 -3이려면 입력 값이 0
- $\text{mod}3$ 의 t 값이 1이려면 입력 값이 79이상(f_3, f_4 를 알아내는 경우에만 사용 가능)
- f 의 계수들 중 75.3% 복원 가능
- 그 이후 BKZ algorithm을 통한 키 복구

3진법



NTRU 대상 부채널 공격 대응기술

■ 변수의 해밍 웨이트값을 이용한 SPA 공격에 대한 대응기술 설계

- r 값이 음수가 되는 부분을 삭제 \rightarrow 해밍 웨이트 차이가 작음
- 8번줄까지 진행되면 $r \in \{0,1,2,3,4\}$ 이며 9,10번 줄을 통해 각각 4,3을 1,0으로 바꾸며 나머지 값들은 고정

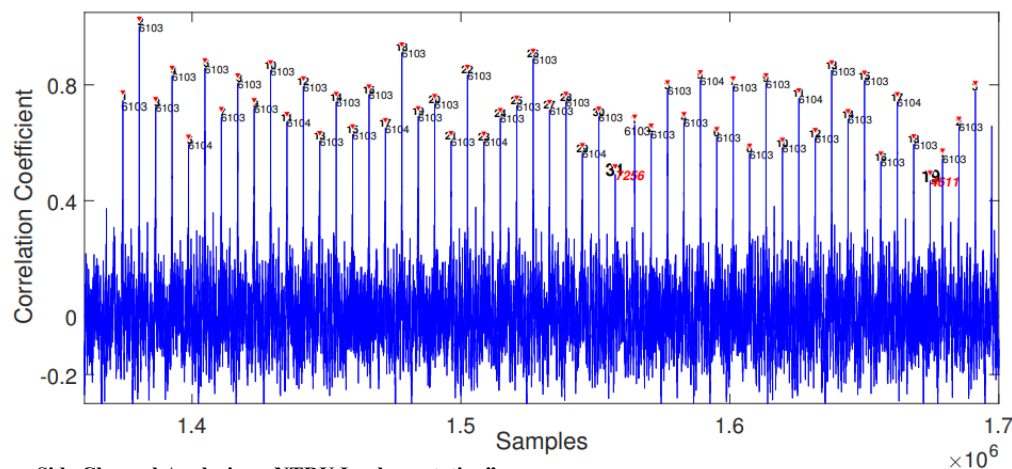
```
1 static uint16_t mod3_alt(uint16_t a)
2 {
3     uint16_t r;
4
5     r = (a >> 8) + (a & 0xff);
6     r = (r >> 4) + (r & 0xf);
7     r = (r >> 2) + (r & 0x3);
8     r = (r >> 2) + (r & 0x3);
9     r = ((r >> 2) + (r)) & 0x3; // Map 4 -> 1
10    return (r + ((r+1)>>2)) & 0x3; // Map 3 -> 0
11 }
```

Listing 3: Alternative mod3() implementation

NTRU 대상 부채널 공격

■ NTRU 곱셈 연산 대상 Single Trace Attack

- NTRU Open Source 곱셈 연산
 - 7번 줄 : $N - b_j$ 번 반복
 - 10번 줄 : b_j 번 반복
 - 반복문 중간과 반복문 끝의 파형이 다름을 이용
 - 상관계수가 낮은 두 파형 사이 길이가 b_j



NTRU Open Source

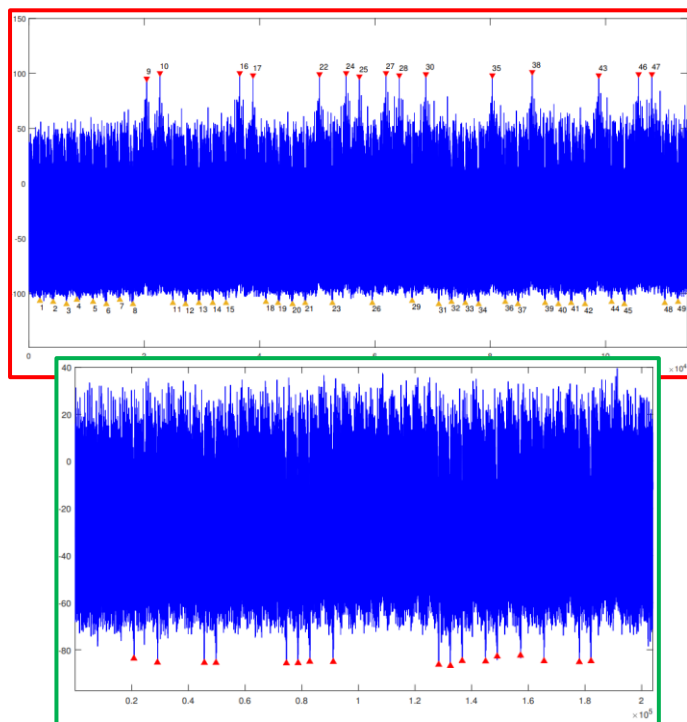
```

1: for  $i = 0; i < N; i++$  do
2:    $t_i \leftarrow 0$ 
3: end for
4: for  $j = d_F + 1; j < 2d_F + 1; j++$  do
5:    $k \leftarrow b_j$ 
6:   for  $i = 0; k < N; i++, k++$  do
7:      $t_k \leftarrow t_k + e_i$   $N - b_j$ 
8:   end for
9:   for  $k = 0; i < N; i++, k++$  do
10:     $t_k = t_k + e_i$   $b_j$ 
11:   end for
12: end for
    
```


NTRU 대상 부채널 공격

■ NTRU 곱셈 연산 대상 Single Trace Attack

- 빨간색 박스를 통해 비밀키 계수 각각을 $\{0,1\}$ 과 $\{-1\}$ 로 구분
- 초록색 박스를 통해 비밀키 계수 0과 1을 구분



참조 문헌 : “Single Trace Side Channel Analysis on NTRU Implementation”

NTRUencrypt

```

1: for  $0 \leq i < N$  do
2:    $f_i \leftarrow F_i \times p$ 
3: end for
4:  $f_0 \leftarrow f_0 + 1$ 
5: for  $0 \leq j < N$  do
6:    $t_j \leftarrow e_0 \times f_j$ 
7: end for
8: for  $1 \leq i < N$  do
9:    $t_{i+N-1} \leftarrow 0$ 
10:  for  $0 \leq j < N$  do
11:     $t_{i+j} \leftarrow t_{i+j} + e_i \times f_j$ 
12:  end for
13: end for      :
```


FALCON 대상 부채널 공격

▪ FFT 상태에서의 곱셈에 대한 CPA 공격

- 키 생성 알고리즘 및 서명 생성 알고리즘 내부 FFT 곱 연산 존재
- FFT 곱 연산은 다항식의 각 계수별 곱셈으로 이루어져 있음
- 서명 생성과정에서 $FFT(c) \odot FFT(f)$ 연산을 공격하며 c 는 아는 값으로 가정

Algorithm 1 FALCON Key Generation Algorithm [5]

Input: A monic polynomial $\phi \in \mathbb{Z}[x]$, a modulus q

Output: A secret key sk and a public key h

```

1:  $f, g, F, G \leftarrow \text{NTRUGen}(\phi, q)$ 
2:  $B \leftarrow \begin{bmatrix} g & -f \\ G & F \end{bmatrix}$ 
3:  $\hat{B} \leftarrow FFT(B)$ 
4:  $G \leftarrow \hat{B} \times \hat{B}^*$   $\triangleright \times$  represents matrix multiplication
5:  $T \leftarrow \text{ffLDL}^*(G)$ 
6: for each leaf of  $T$  do
7:    $leaf.value \leftarrow \sigma / \sqrt{leaf.value}$ 
8: end for
9:  $sk \leftarrow (\hat{B}, T)$ 
10:  $h \leftarrow gf^{-1} \bmod(q)$ 
11: return  $sk, h$ 
```

Algorithm 2 FALCON Signature Generation Algorithm [5]

Input: a message m , a secret key sk , a bound β^2

Output: a signature sig of m

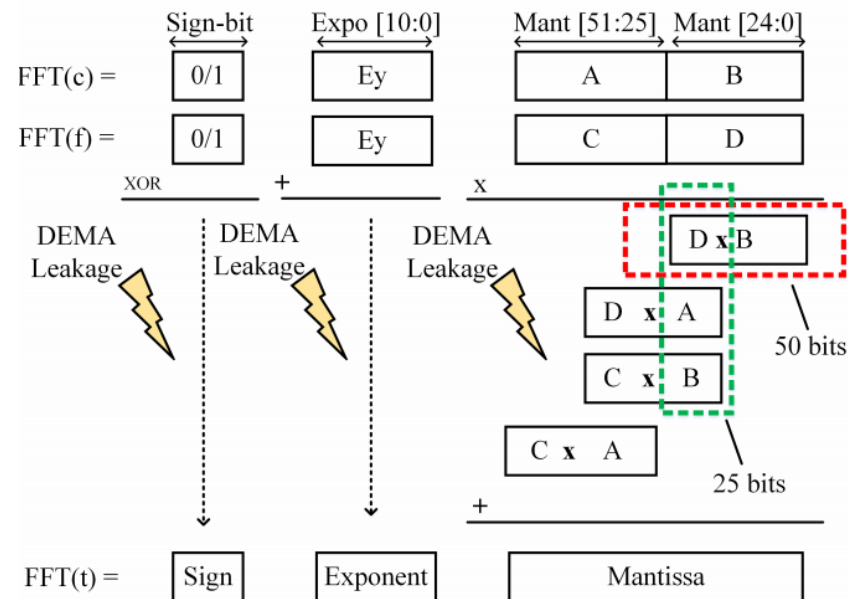
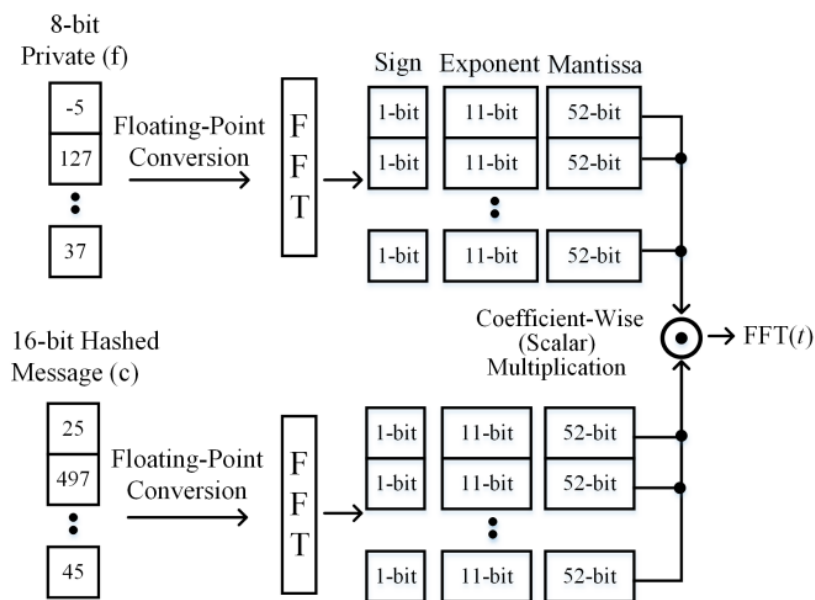
```

1:  $r \leftarrow \{0, 1\}^{320}$  uniformly
2:  $c \leftarrow \text{HashToPoint}(r || m)$ 
3:  $t \leftarrow (\frac{-1}{q} FFT(c) \odot FFT(F), \frac{1}{q} FFT(c) \odot FFT(f))$ 
4: do  $\triangleright \odot$  represents FFT multiplication
5:   do
6:      $z \leftarrow \text{ffSampling}(t, T)$ 
7:      $s \leftarrow (t - z) \begin{bmatrix} FFT(g) & -FFT(f) \\ FFT(G) & -FFT(F) \end{bmatrix}$ 
8:     while  $s^2 > [\beta^2]$ 
9:        $(s_1, s_2) \leftarrow \text{invFFT}(s)$ 
10:       $s \leftarrow \text{Compress}(s_2, 8 \cdot \text{sbytelen} - 328)$ 
11:   while  $s = \perp$ 
12: return  $sig = (r, s)$ 
```

FALCON 대상 부채널 공격

■ 실수의 자료형

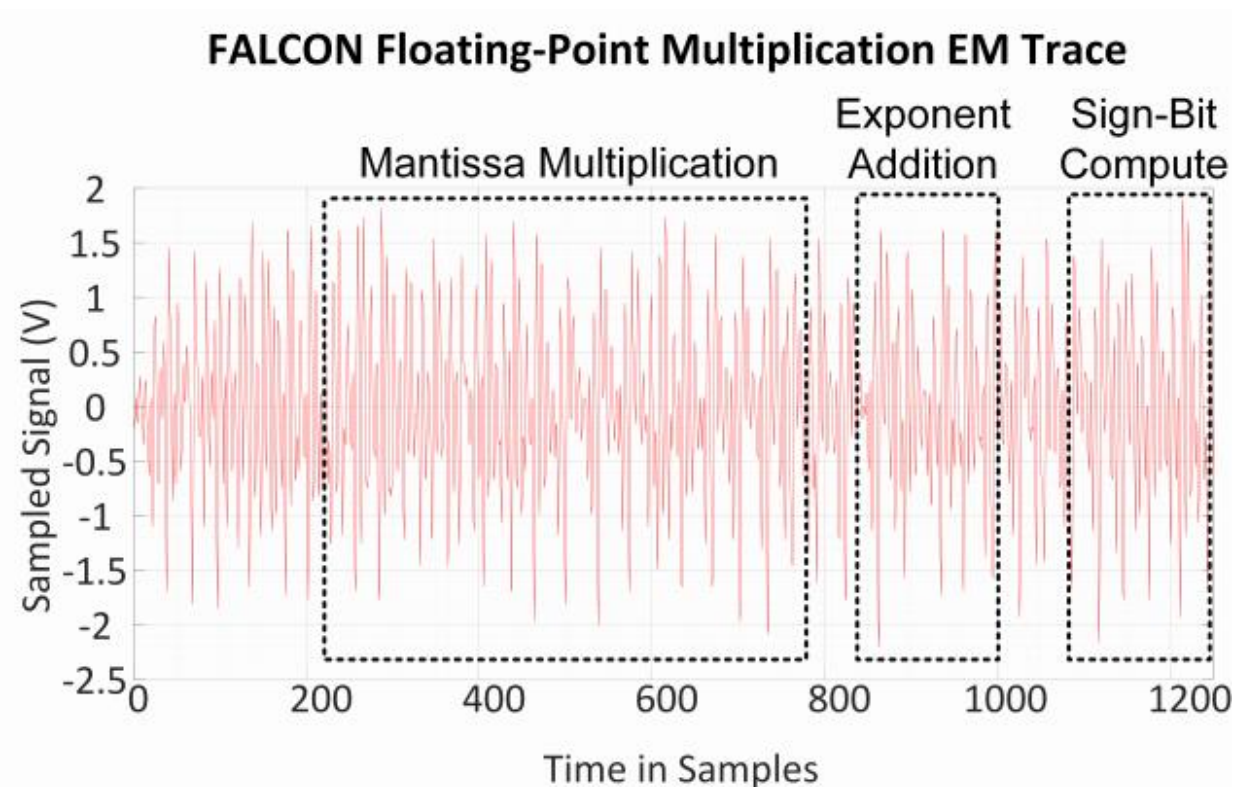
- 비밀키 f 와 중간 값 c 는 기존 정수계수 다항식에서 FFT를 통해 실수계수 다항식으로 변환
- 실수는 컴퓨터에서 부호, 지수, 가수를 각각 1, 11, 52 bit로 구성
- 부호, 지수, 가수를 각각 s, e, m 라 한다면, $(-1)^s \cdot 2^{e-1023} \cdot 1.m$
- 가수들의 네 번의 곱셈 및 덧셈연산, 지수들의 덧셈연산, 부호들의 xor연산 나타남



FALCON 대상 부채널 공격

■ 파형의 형태

- 가수 연산, 지수 연산, 부호 연산 순서로 나타남
- 가수의 곱셈 과정에서의 false positive는 이후의 덧셈 과정에서 찾을 수 있음

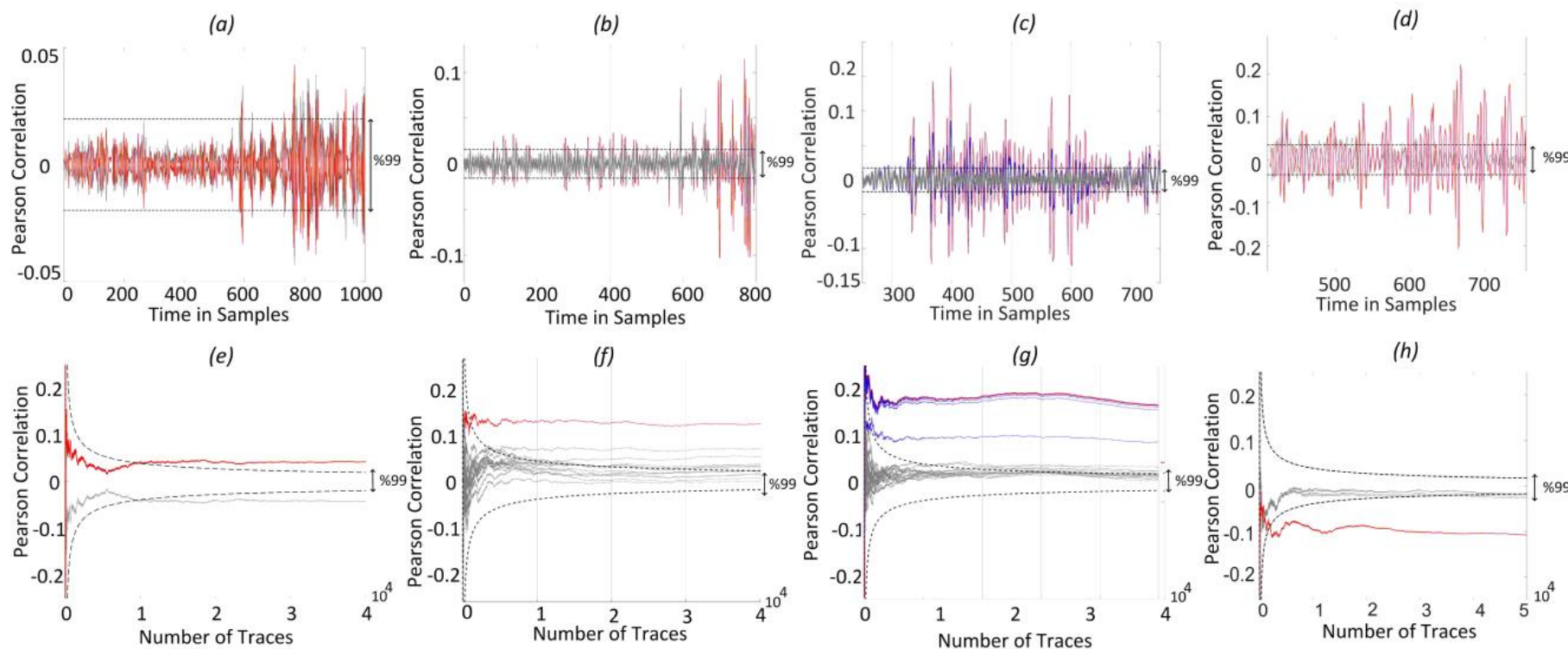


참조 문헌 : "Falcon Down: Breaking Falcon Post-Quantum Signature Scheme through Side-Channel Attacks"

FALCON 대상 부채널 공격

■ 각 과정에 대한 CEMA 공격

- (a)~(d)는 시간별로 계산한 상관계수 값(색은 정답)
- (e)~(h)는 사용한 파형 수에 따른 상관 계수 값(색은 정답)



Attacking the Sign-bit

Attacking the Exponent

Attacking the Mantissa Multiplication

Attacking the Mantissa Addition

Q&A

Thanks



IA&AI SecLab
Implementation Attacks & AI Security Laboratory