IoT 부채널 분석기술 소개
(Introduction of Side Channel Analysis)

김태성
2020 07 17 한국전자통신연구원

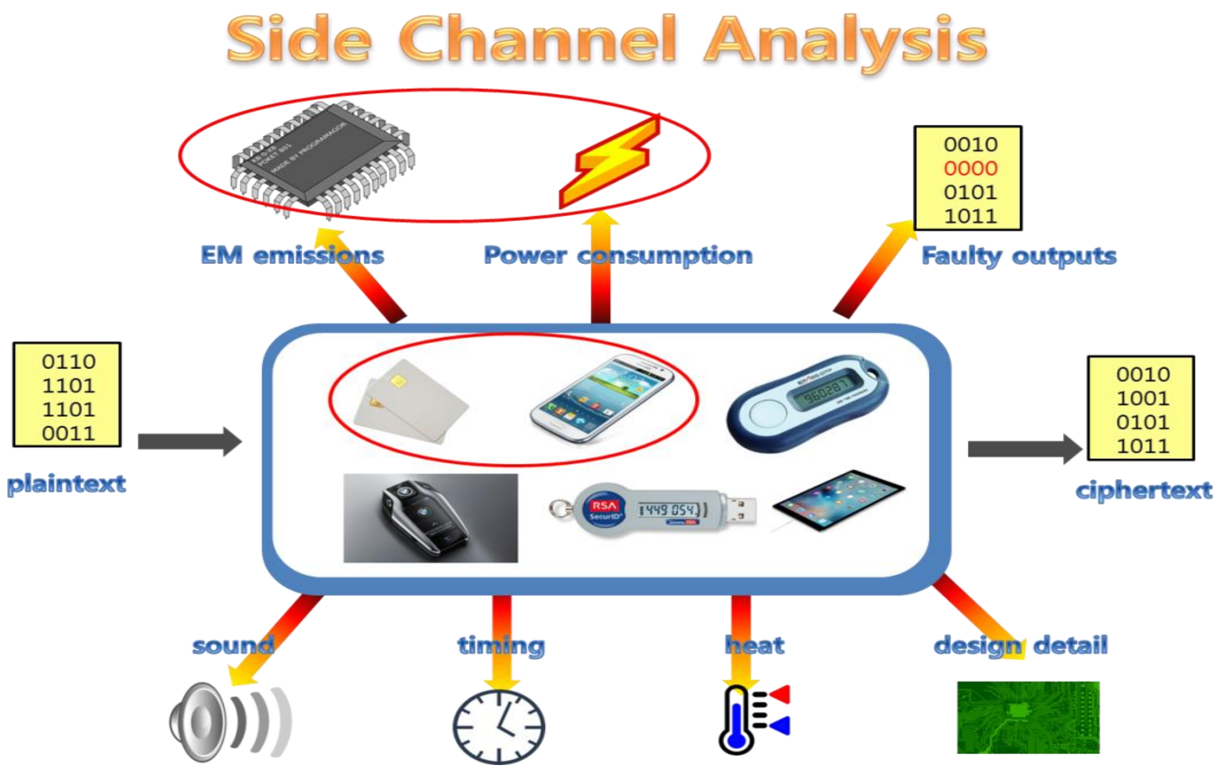
Contents

- Side Channel Analysis(Attack).
- Simple Power Analysis.
- Differential Power Analysis.
- SEED(Block Cipher).
- Open SCARF (Demo)
- Countermeasures.
- SCA Data Set.

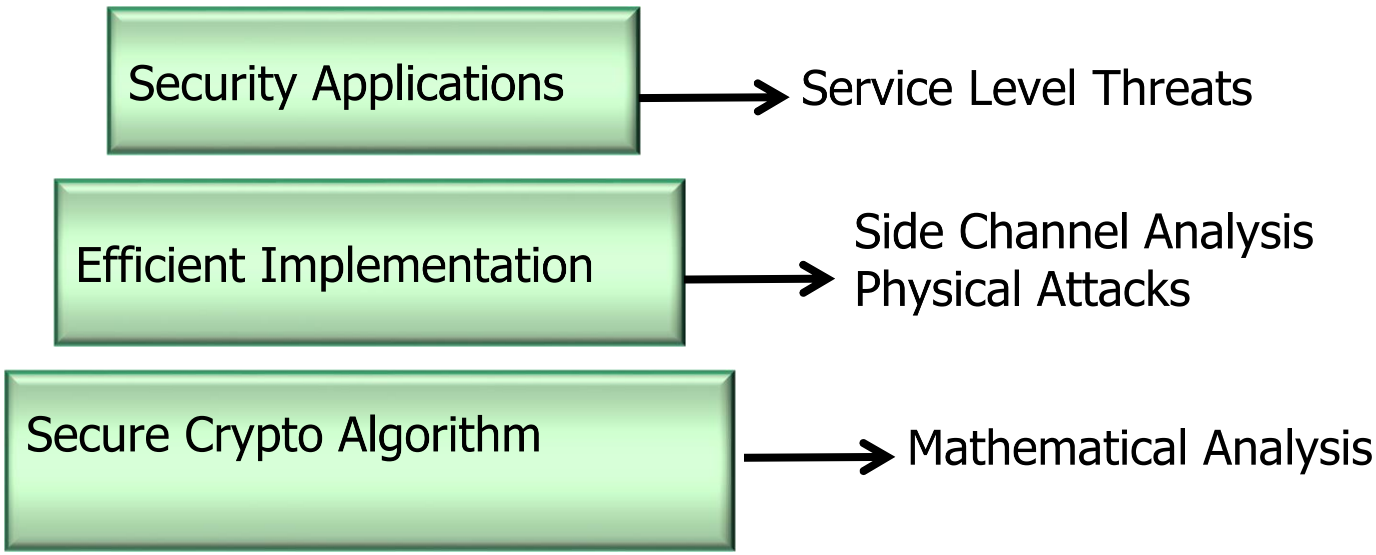
부채널 분석 – Side Channel Analysis

정의

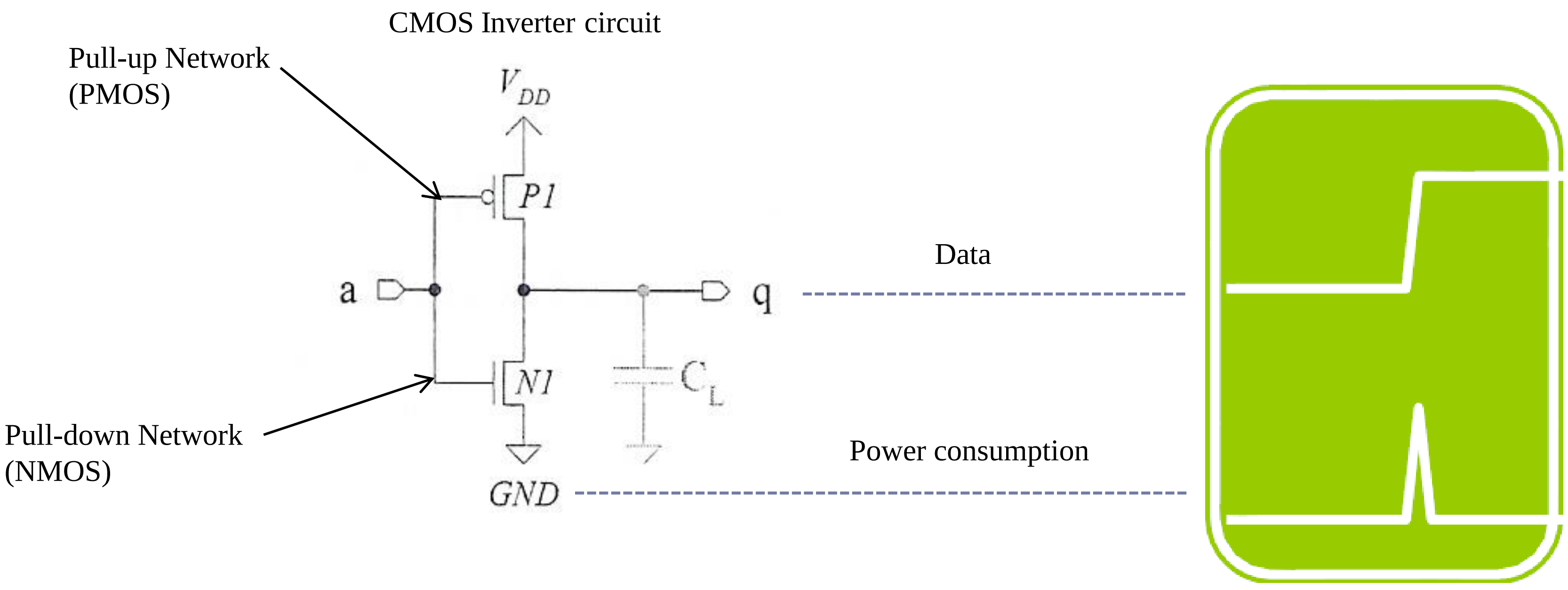
암호 모듈이 다양한 디바이스에 탑재되어 구동되는 동안 발생하는 각종 부가적인 정보(구동 시간, 발열, 소리, 전력소모량, 전자기파, 오류주입결과 등)를 이용하여 암호 모듈의 비밀 정보(비밀키, 부분키)를 크랙킹하는 공격 방법 **(Kocher, CRYPTO' 96)**



부채널 분석 – 원리


Security Applications
Service Level Threats
Efficient Implementation
Side Channel Analysis
Physical Attacks
Secure Crypto Algorithm
Mathematical Analysis

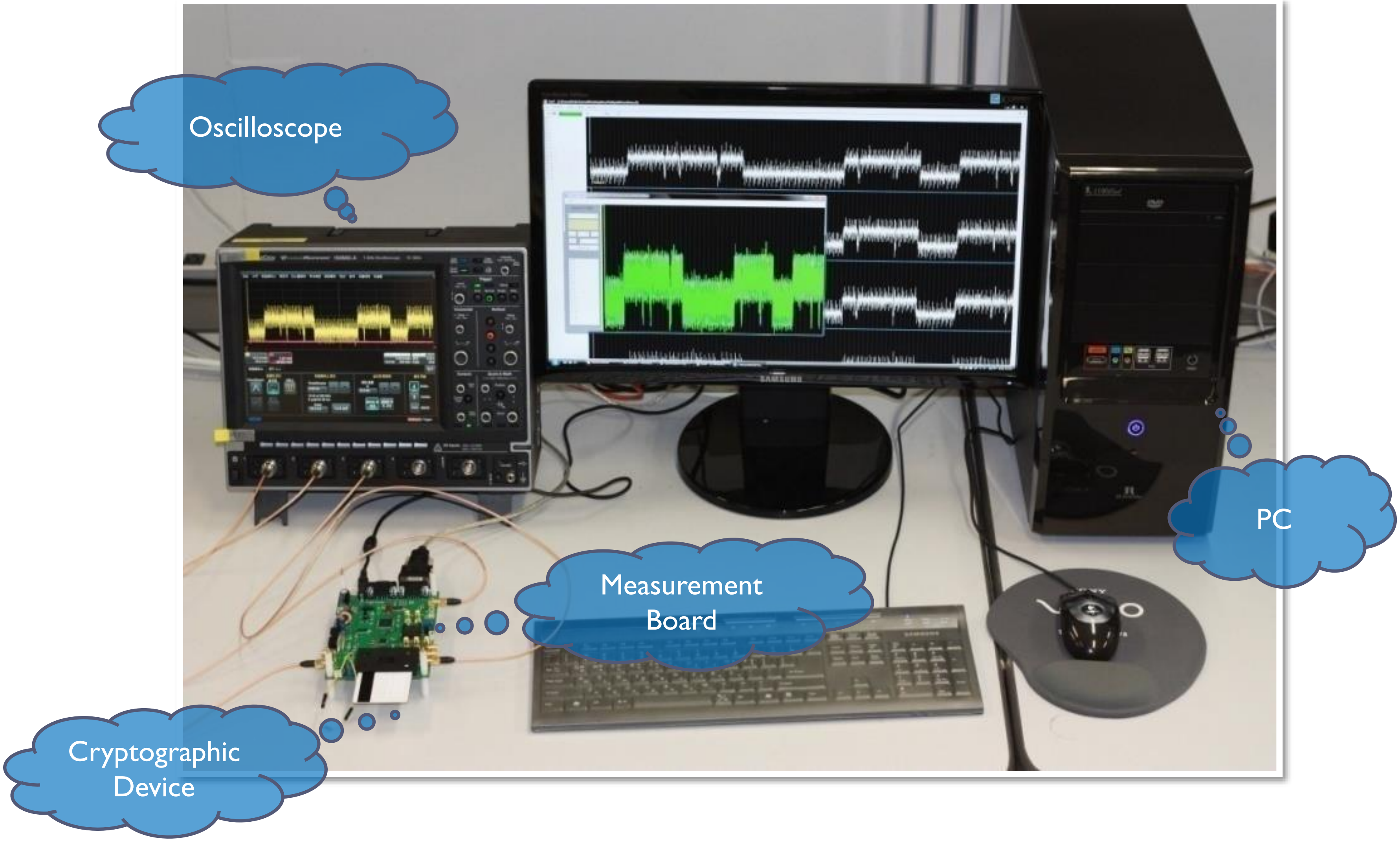
부채널 분석 – 원리



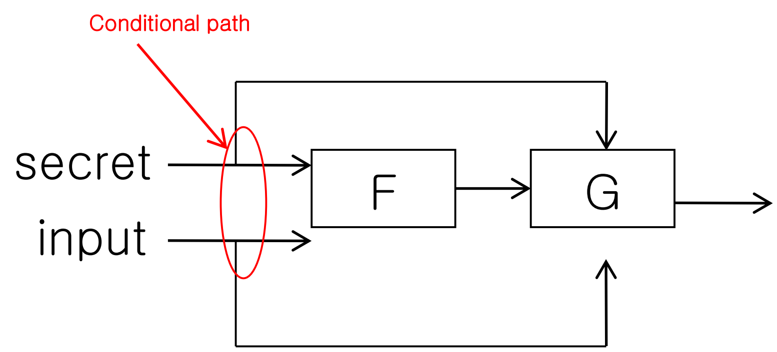
Power = P(static) + P(dynamic)

1→0 or 0→1 switch 될 때, P(dynamic) 발생함

Side Channel Analysis - Experiment

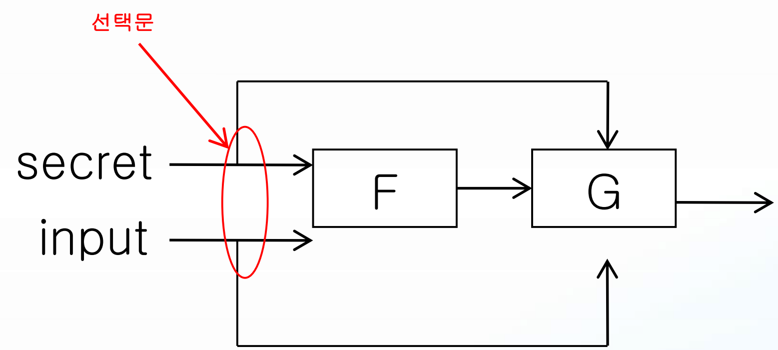


Simple Power Analysis(SPA)



A technique that involves directly interpreting power consumption measurement(i.e. traces) collected during cryptographic operations.

RSA exponentiation



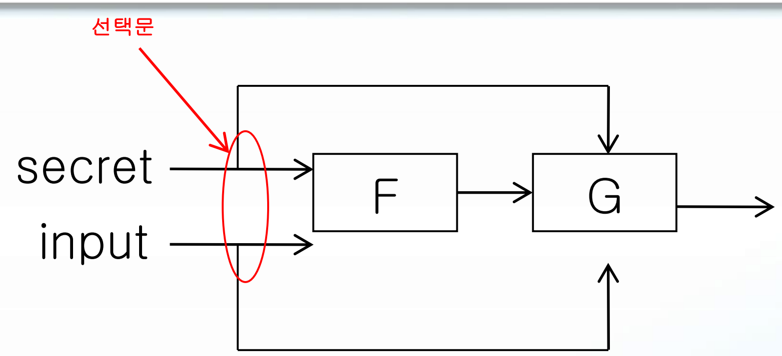
Algorithm 1 Left-to-Right Exponentiation Algorithm

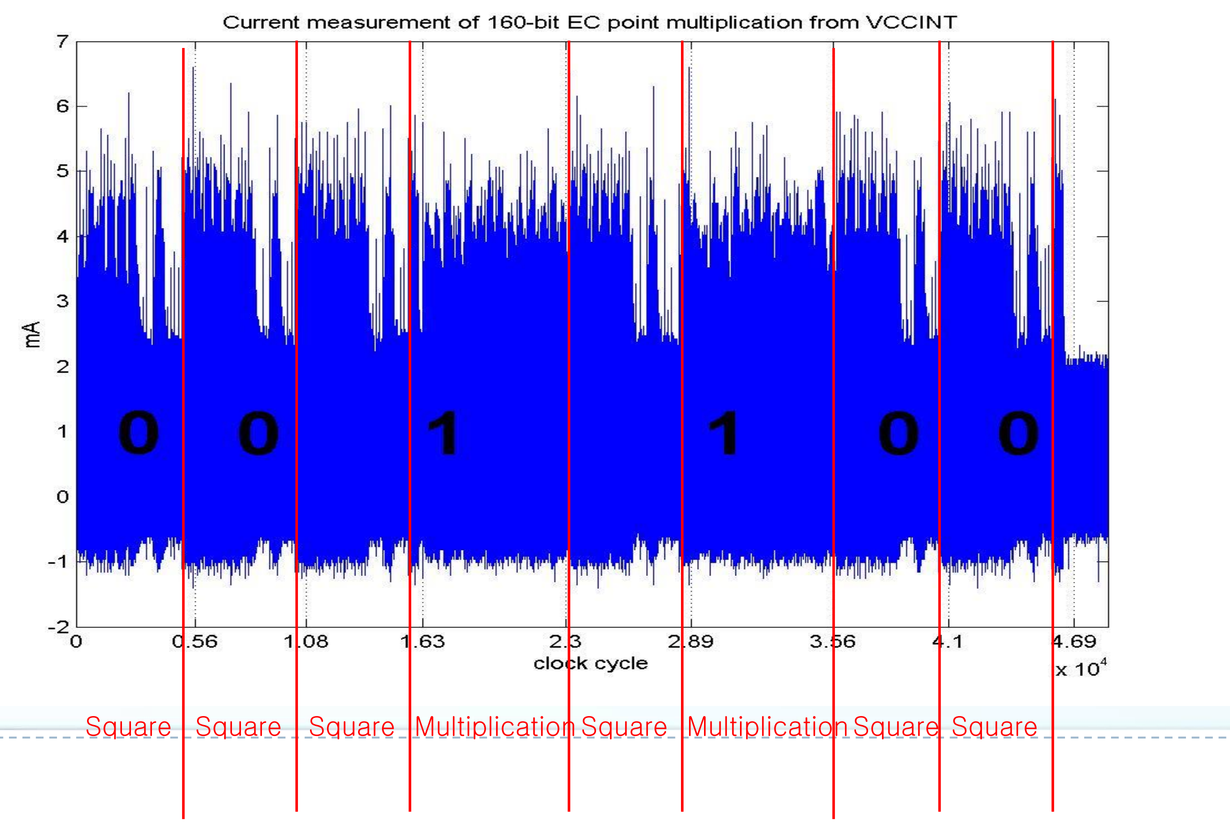
Input: a, $b = [b_{n-1}b_{n-2}b_1b_0]$ with binary expression
Output: $c = a^b \mod m$

```
c ← 1
for k = n − 1 down to 0 do
    c ← c · c  mod m
    if b[k] = 1 then
        c ← c · a  mod m
    end if
end for
return c
```

!!!!!!! 선택문 !!!!!!!

RSA exponentiation



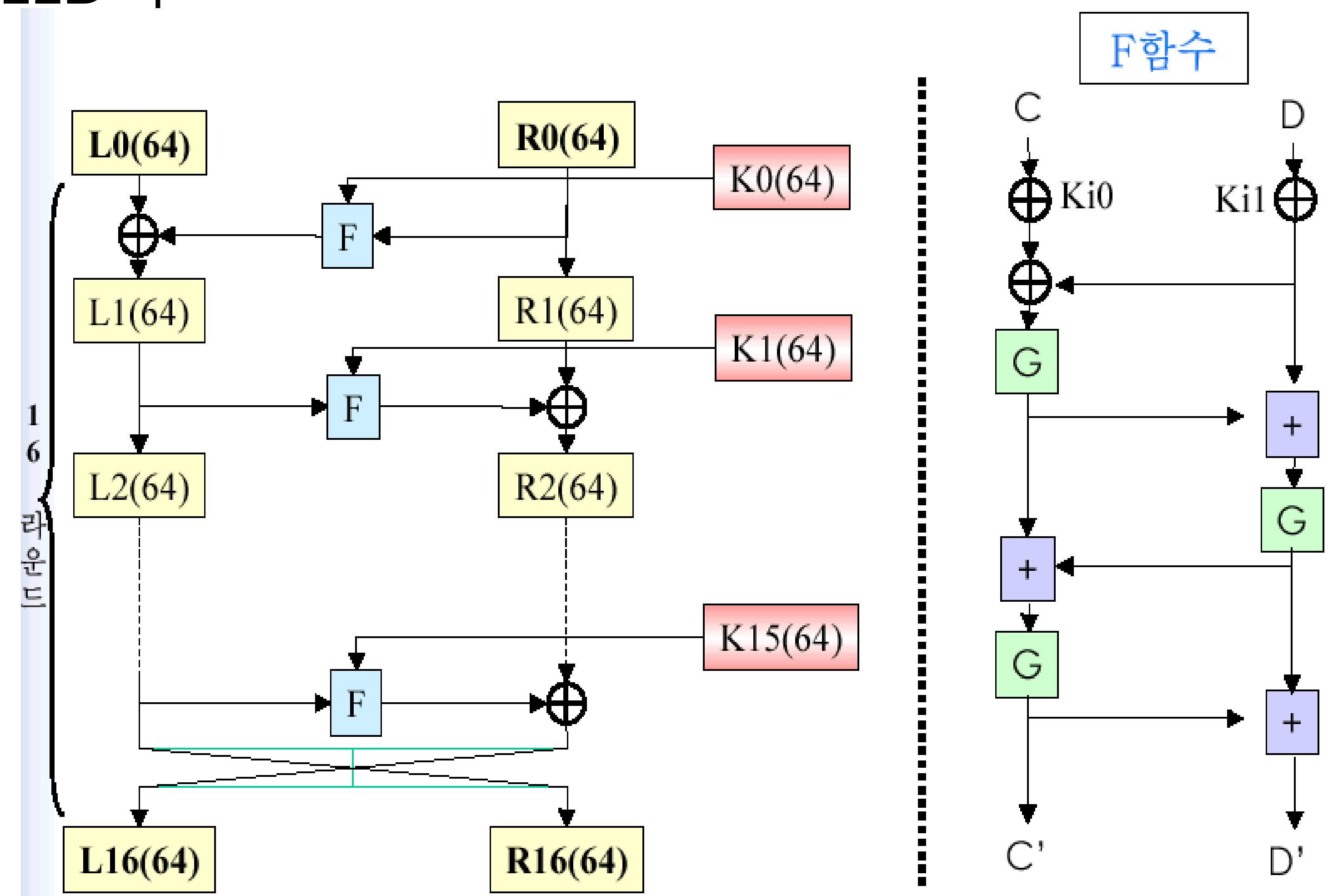


SEED

- KISA에 의해 설계
- 블록 크기:128비트, 키 크기: 128비트
- 구조: 16라운드 Feistel 구조
- DC와 LC 공격에 안전하게 설계

SEED

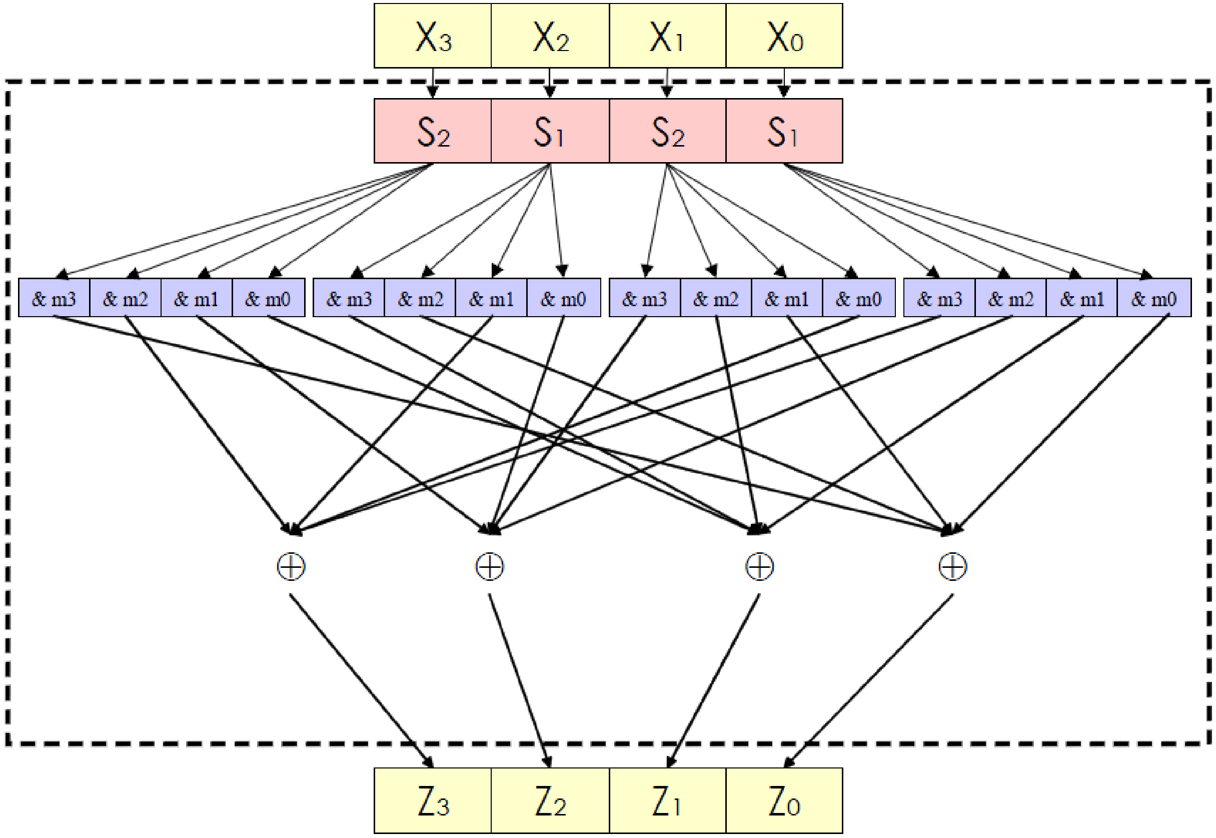
- SEED 구조


L0(64)
R0(64)
K0(64)
F
L1(64)
R1(64)
K1(64)
F
16 라운드
L2(64)
R2(64)
K15(64)
F
L16(64)
R16(64)
F함수
C
D
Ki0
Ki1
G
+
G
+
G
+
C'
D'

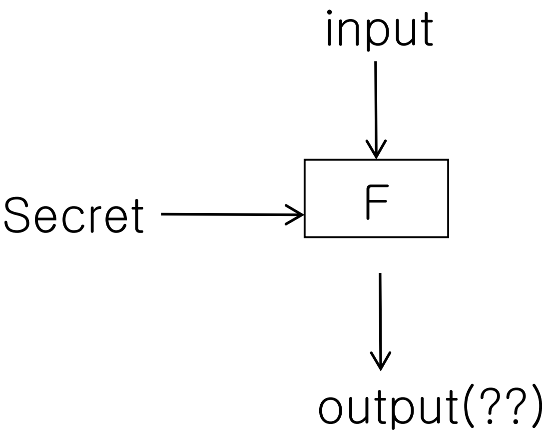
SEED

- G 함수 : 32비트 입/출력

$$(m_0 = 0\text{xfc},\quad m_1 = 0\text{xf3},\quad m_2 = 0\text{xcf},\quad m_3 = 0\text{x3f})$$

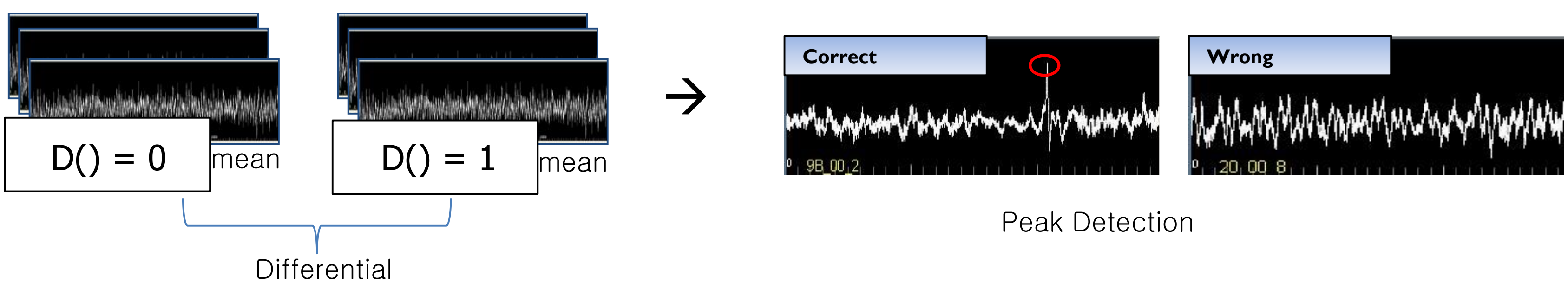


Side Channel Analysis -Differential Power Analysis



- **Attack strategy**
  - (1) For many inputs $m_1, m_2, \ldots m_k$,
    Collect power traces $T_1, T_1, \ldots T_K$
  - (2) Guessing "Secret" and calculate some value related to H(output)
  - (3) Compare and calculate some statistical value between (2) and power traces
  - (4) Find maximum(minimum) for all possible secret

$D(s_i, m_j)$ = H(the first bit of $F(s_i, m_j)$)



What is DPA?

- Gray box attack model
- Intermediate state
  - $I(p_i, k)$ p is plaint text, k is small portion of key(usually one byte)
- Power consumption
  - $L(I(p_i, k)) + \delta$
- Guess key(0 ~ 255)를 입력으로 중간값을 구하고 이를 두 개의 서브셋으로 분류한다.
- 분류된 두 서브셋의 전력소비값의 평균의 차를 구한다.
- guess key가 틀릴 경우 랜덤하게 분류가 되어 평균의 차는 0에 가까울 것이다.
- 반대로 guess key 맞다면 다른 후보키보다 높은 값을 가질 것이다.

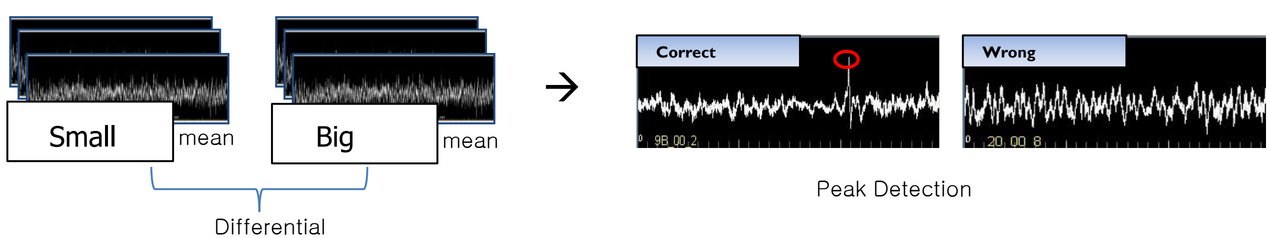
DPA Example(1)

- 파형 수집
  - Input 데이터와 Trace(파형)의 한 쌍
  - Example
    - AB (2,5,7,3,8,9,3)
    - 05 (5,7,3,0,9,8,6)
    - F2 (5,7,8,3,0,0,5)
    - …..
- DPA 분석
  - 목표로 하는 중간 값 함수
    - Intermediate = Sbox(Plain xor Key)
    - Sbox 는 input이 바이트이고 output이 바이트라고 생각하면 됨.
  - 키의 후보는 한 바이트 공격을 할 것이므로 0x00 ~ 0xFF 까지 256개임.

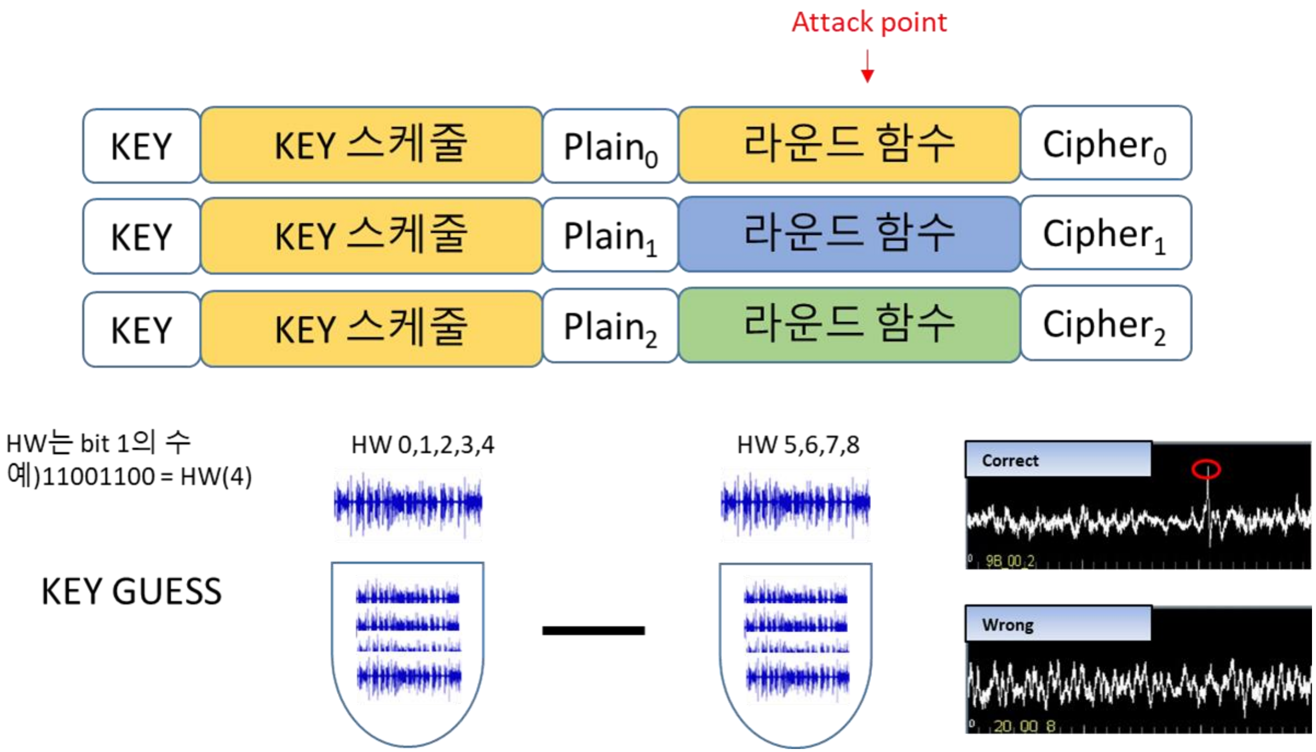
DPA Example(2)

- DPA 분석
  - 모든 후보키와 모든 전력 소모값에 대하여 다음을 반복 수행.
    - 0xC2가 현재 계산하고 있는 후보키라면 (0x00 ~ 0xFF 중의 하나)
    - 0x54 = Sbox(0xC2 xor 0xAB), 따라서 중간값은 0x54.
    - 0x54(010100100) 이므로 HW는 3
    - HW 4를 기준으로 Small과 Big 그룹으로 분류
    - 따라서 전력 소모값 2는 Small 그룹임.
    - 0x05와 0xF2가 Big 그룹이 된다면, Small{2} , Big{5,5}
    - 각 그룹의 차가 DPA 결과 값이므로 -3 = AvgSmall(2) – AvgBig(5)
    - 모든 포인트의 계산을 하면 DPA of 0xC2 = {-3, -2, 1.5, 1.5, 3.5, 5, -2.5}

- AB (2,5,7,3,8,9,3)
- 05 (5,7,3,0,9,8,6)
- F2 (5,7,8,3,0,0,5)
- .....



DPA (차분전력분석)의 원리 개념



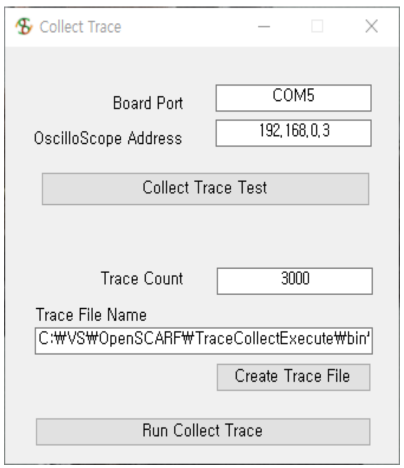
- 평문을 달리해서 입력하여 암호 알고리즘 내의 중간값 예측
- 분류된 파형 SET의 평균 파형의 차에서 피크가 있으면 공격 성공

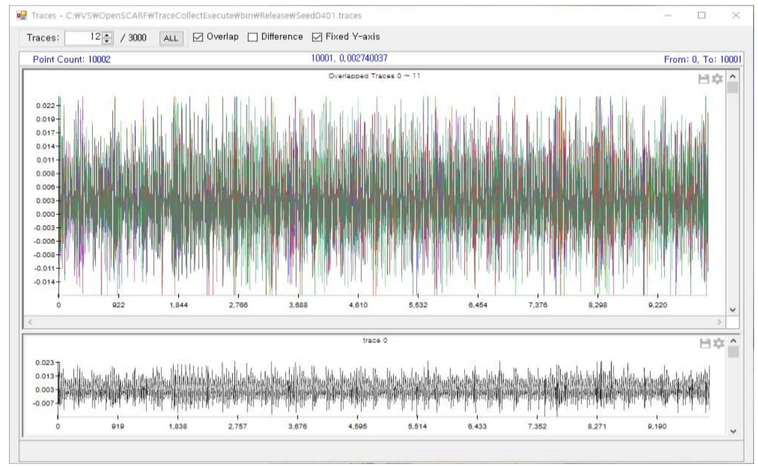
Open SCARF 소개

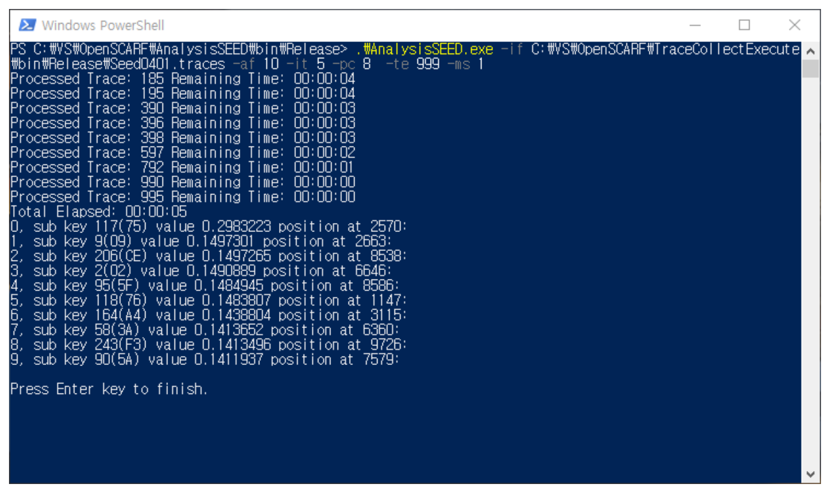
- 소프트웨어 검증 보드 및 부채널 분석 공개 소프트웨어.
  - 소프트웨어검증 보드
  - 공개 소프트웨어(구성)
    - 파형 수집 프로그램(바이너리)
    - 파형 보기 프로그램(바이너리)
    - 부채널 분석 프로그램(오픈 소스)
      - SEED 알고리즘에 대한 CPA 분석.
      - C# 언어로 구현.
- www.trusthingz.org에 공개 예정.
  - 바이너리 및 소스 다운로드
  - 튜토리얼 형식의 위키 페이지 설명

데모

Open SCARF 스크린 샷

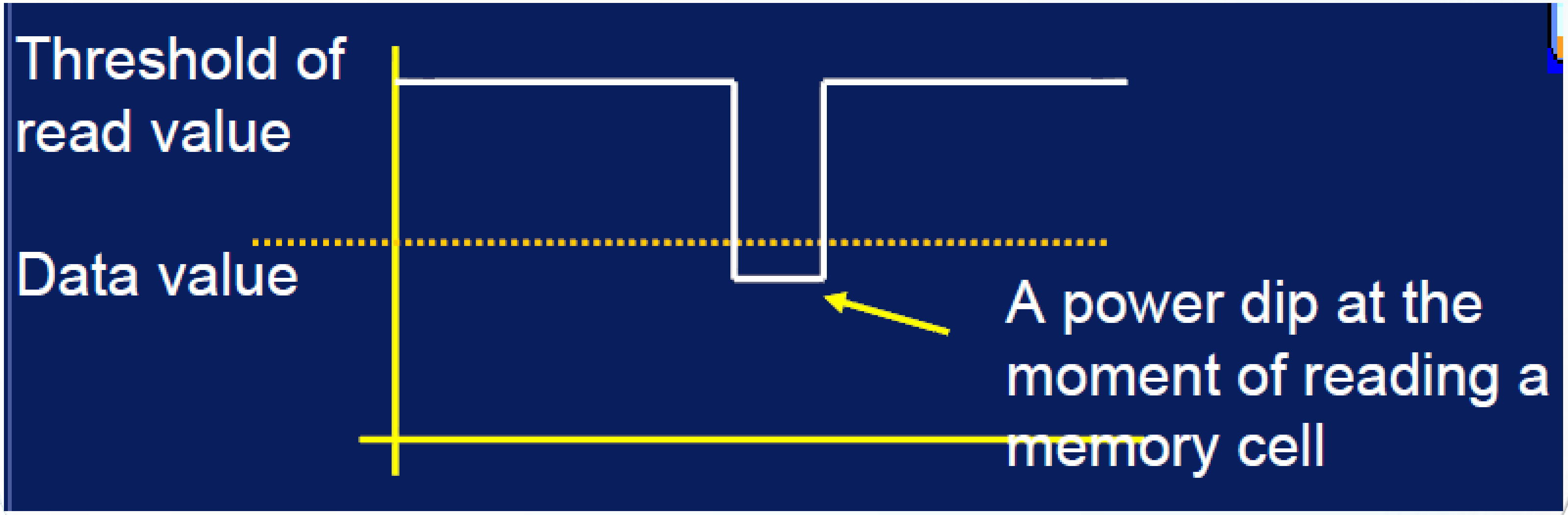




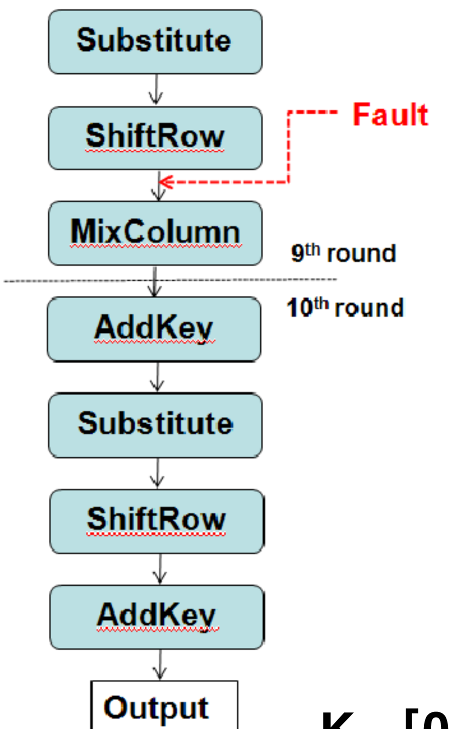


오류 주입 부채널 분석

- Voltage glitching
  - Very short glitches on the supply voltage
  - Can change the value of read data



AES 오류 주입 공격



After ShiftRows 9

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Fault injected 1E

99	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

After Mixcolumn

7B	40	A3	4C
29	D4	70	9F
8A	E4	3A	42
CF	A5	A6	BC

$\oplus$

K_9

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6E

After AddRoundKey 9

D7	59	8B	1B
5E	2E	A1	C3
EC	38	13	42
3C	84	E7	D2

After SubBytes 10

0E	CB	3D	AF
58	31	32	2E
CE	07	7D	2C
EB	5F	94	B5

After ShiftRows 10

0E	CB	3D	AF
31	32	2E	58
7D	2C	CE	07
B5	EB	5F	94

$\oplus$

value of K_{10}

D0	C9	E1	B6
14	EE	3F	63
F9	25	0C	0C
A8	89	C8	A6

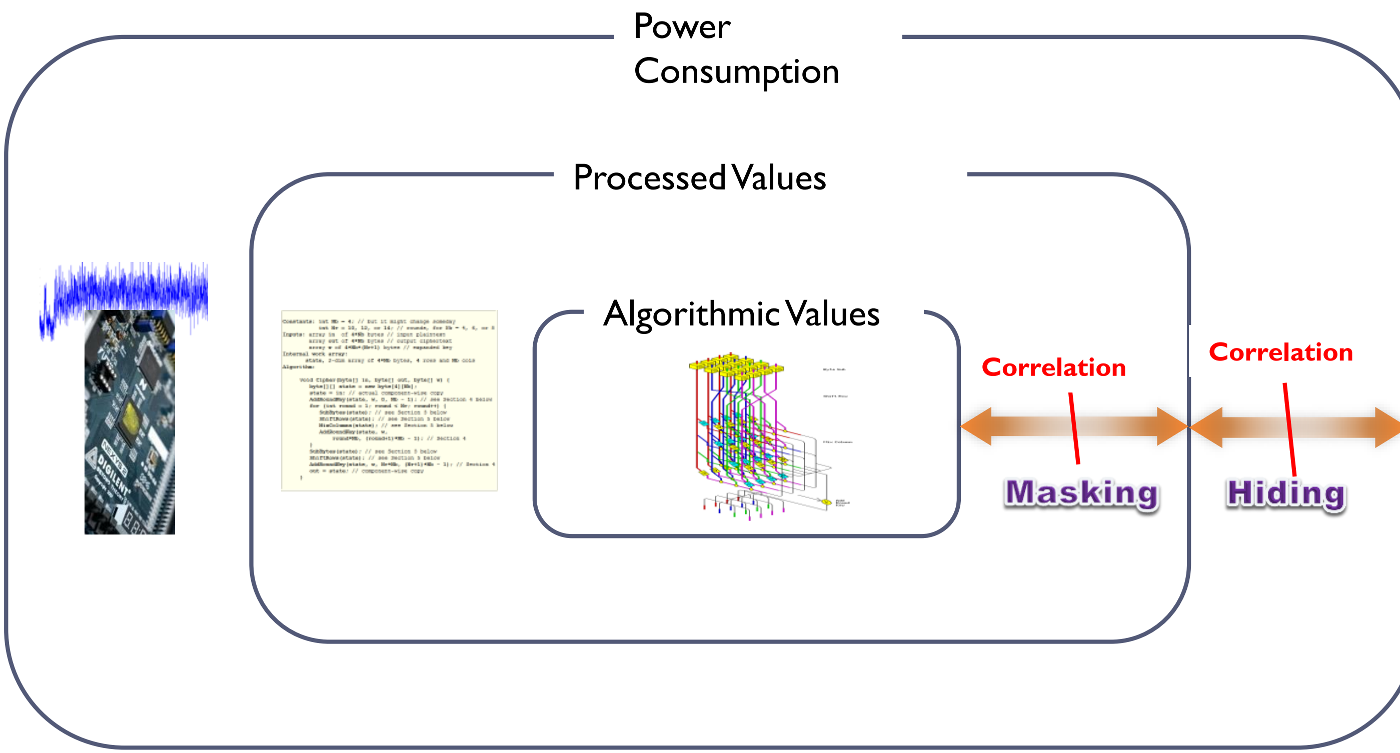
Output with Faults

DE	02	DC	19
25	DC	11	3B
84	09	C2	0B
1D	62	97	32

$K_{10}[0]$의 한번의 오류 공격으로 좁혀진 키 후보들

$K_{10}[0] \in$ {'03', '06', '09', '0C', '10', '15', '1A', '1F', '21', '24', '2B', '2E', '32', '37', '38', '3D', '43', '46', '49', '4C', '50', '55', '5F', '61', '64', '6B', '6E', '72', '77', '78', '7D', '83', '86', '89', '8C', '90', '95', '9A', '9F', 'A1', 'A4', 'AB', 'AE', 'B2', 'B7', 'B8', 'C3', 'C6', 'C9', 'CC', **'D0'**, 'D5', 'DA', 'DF', 'E1', 'E4', 'EB', 'EE', 'F2', 'F7', 'F8', 'FD'}

Countermeasures - Concept



Countermeasures

- Hardware
  - Signal reduction
  - Adding amplitude noise
  - Adding timing noise
  - Dedicated components
- Software
  - Time constant programming
  - Adding random delay or alternating paths
  - Blinding(masking) of intermediate values with random values

SCARF Data Set

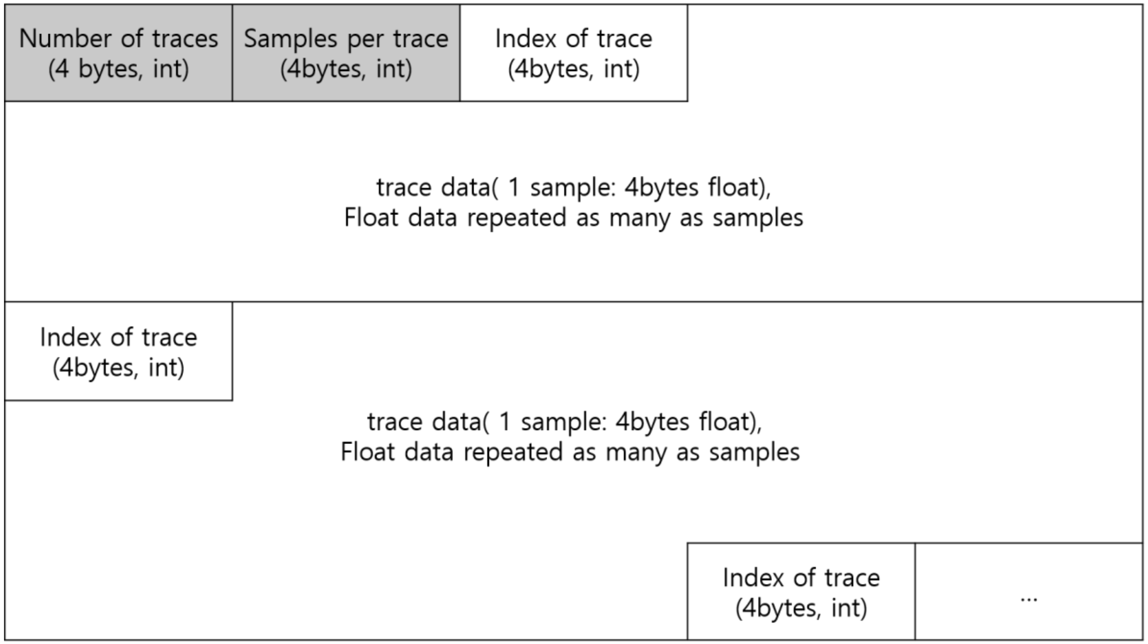
- **Acquisition Board for Data set**
  - SCARF AVR Board (8 bit MCU ATmega 128)
  - SCARF STM Board (32bit MCU STM32F411RET)
- **Cryptographic Algorithms**
  - Unprotected AES, First order masked (with shuffling) AES (full round masking)
  - Unprotected SEED, First order masked SEED (first and last 2 round masking)
  - Unprotected LEA, First order masked LEA (first and last 2 round masking)

SCARF Data Set

- **Type and number of traces provided**
  - Fixed key and 2,000 traces (Can be used DPA)
  - Variable keys and 5,000 traces (Can be used machine learning)
- **Data set file information**
  - {identifier}-info.txt : information text file
  - {identifier}.btr : trace binary file(see below for structure of the binary file)
  - {identifier}-plain.txt : plaintext expressed in hexadecimal
  - {identifier}-cipher.txt : cipher expressed in hexadecimal
  - {identifier}-key.txt : key expressed in hexadecimal

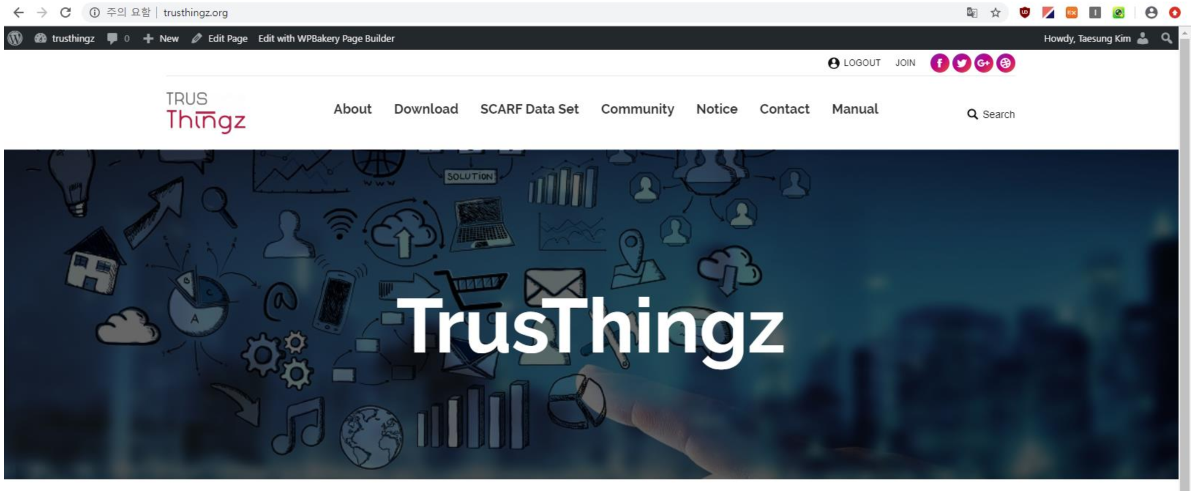
SCARF Data Set

- **Structure of the binary trace file**



SCARF Data Set

- Open Web Site
  - http://www.trusthingz.org
  - Visible only to who logged in the site.
  - We will keep updating!



감사합니다

Scarf

Side Channel Analysis Resistant Framework