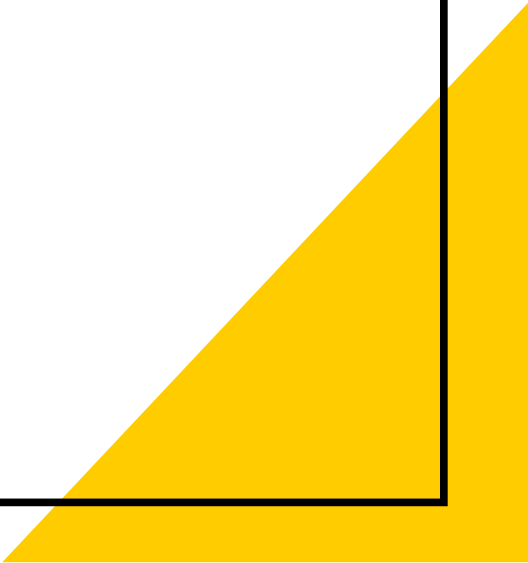


전력 분석 공격 2

IT정보공학과 신명수

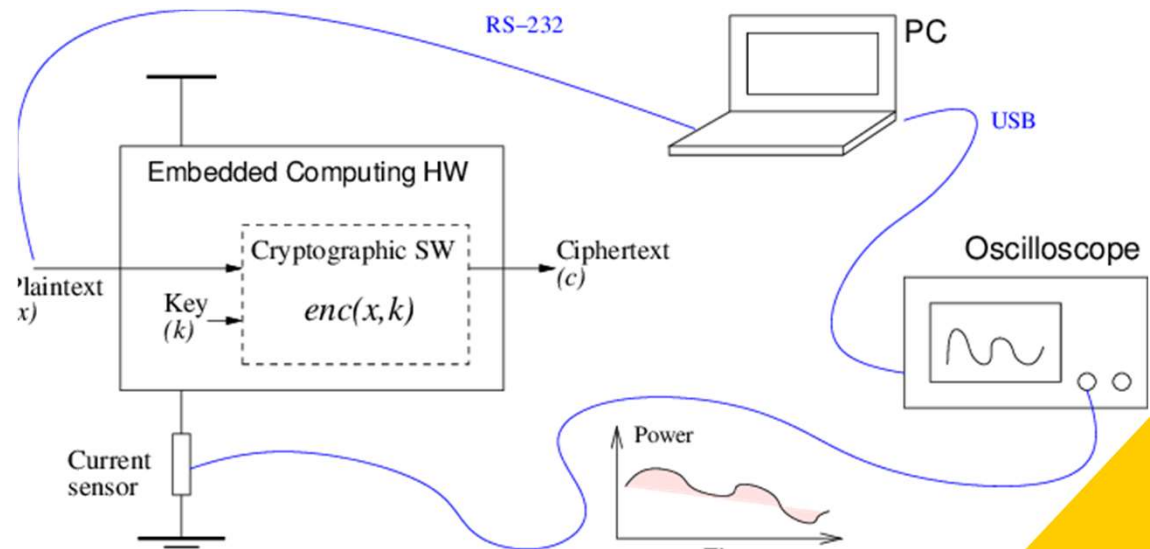
목차

1. About Side Channel Attack
 2. DPA (Differential Power Analysis)
 3. CPA (Correlation Power Analysis)
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

1. About Side Channel Attack

Power Analysis

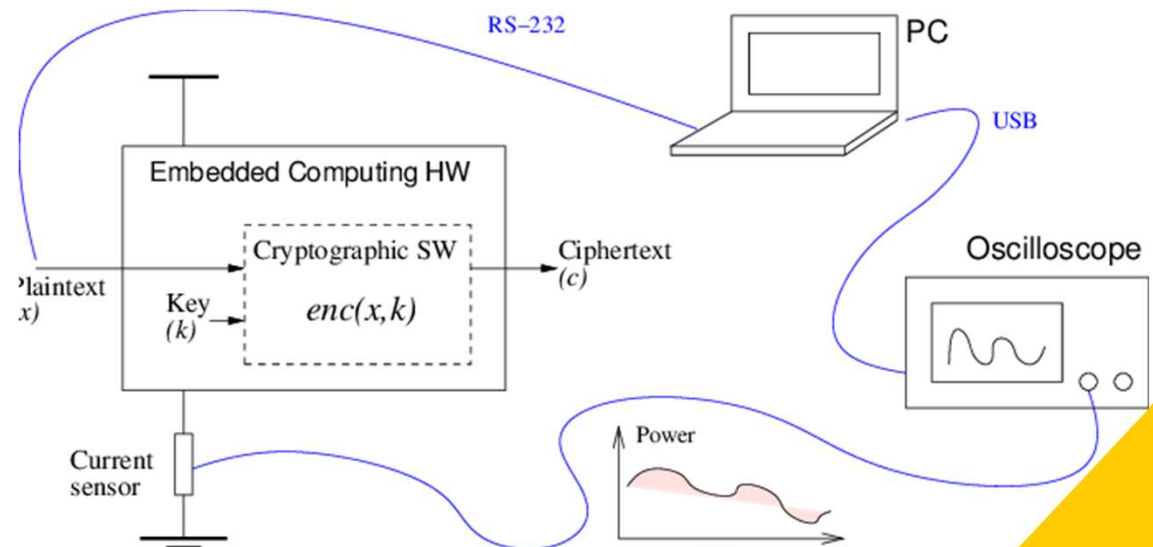
- 암호 모듈이 처리하는 데이터와 연산에 따라서 순간 소비 전력의 차이를 사용하는 공격 기법.
- 암호 모듈이 설치된 기기에 저항과 오실로스코프를 연결해 파형을 수집.



1. About Side Channel Attack

Power Analysis

- 소비 전력은 암호 모듈이 동작할 때 입력받은 정보에 따라서 약간의 차이를 보이게 된다.
- 전력 분석 공격은 데이터와 소비전력의 연관성을 바탕으로 공격

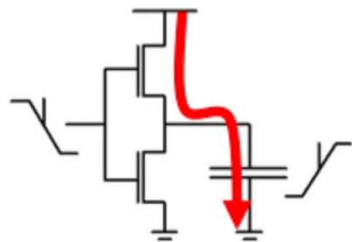


1. About Side Channel Attack – Power Analysis

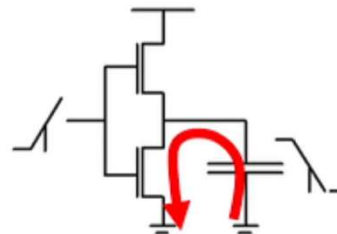
내부 연산 데이터와 소비 전력 사이 연관성

데이터가 0에서 1로 바뀔 때, 1에서 0으로 바뀔 때 전력 소비가 발생한다.

- 전력 분석은 데이터가 바뀔 때 소비 전력의 차이를 바탕으로 공격을 한다.



0-1 transition



1-0 transition

1. About Side Channel Attack – Power Analysis

데이터 상태 변화 모델링

- Hamming Weight Model (the number of bits set to 1)

$$HW(x) = \sum x[i], \quad x = (x[0], \dots, x[n-1]) \in \{0, 1\}^n$$

$$ex. HW(\textcolor{red}{1}10\textcolor{red}{1}00\textcolor{red}{1}0\textcolor{red}{1}) = 5.$$

- Hamming Distance Model

$$HD(x_0, x_1) = HW(x_0 \oplus x_1)$$

$$ex. HD(0010, 0001) = 2.$$

1. About Side Channel Attack – Power Analysis

데이터 상태 변화 모델링

- 데이터가 변경될 때 전력 소비량이 발생
-> 비트가 바뀌는 만큼 전력 소비가 일어나는 것을 모델링함.

2. DPA (Differential Power Analysis)

- 다수의 파형을 통계적으로 분석하여 암호 알고리즘 비밀키를 추출하는 방법.
- 블록암호 비밀키 추출에 효과적임.

2. DPA (Differential Power Analysis)

- 다수의 전력 파형을 사용함.
- 파형 수집과 비밀키 추출 단계를 구분한다.
- 연산 데이터의 소비 전력 모델 정보를 활용한다.
- 통계적 기법을 활용해 신호 노이즈에 내성이 있다.

2. DPA (Differential Power Analysis)

DPA 공격 조건

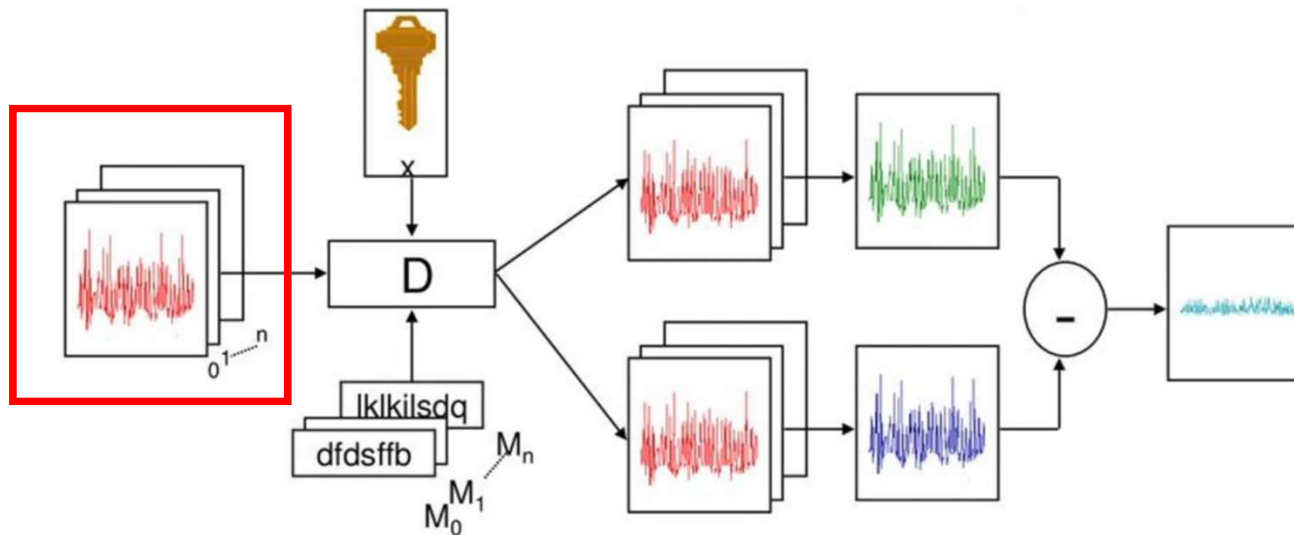
1. 부채널 신호는 연산 데이터에 의존한다.
2. 암호 알고리즘의 동작 방식은 공개되어있다.
3. 공격자는 충분한 수의 부채널 신호를 수집할 수 있다.
4. 공격자는 암호 알고리즘의 입력 또는 출력을 알 수 있다.

2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

1. 다수의 임의의 평문을 입력하여 소비전력을 측정

Input 데이터 – Trace(파형)의 한쌍 ex. AB – (2, 5, 7, 3, 8, 9, 3), 05(5, 7, 3, 0, 9, 8, 6)



2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

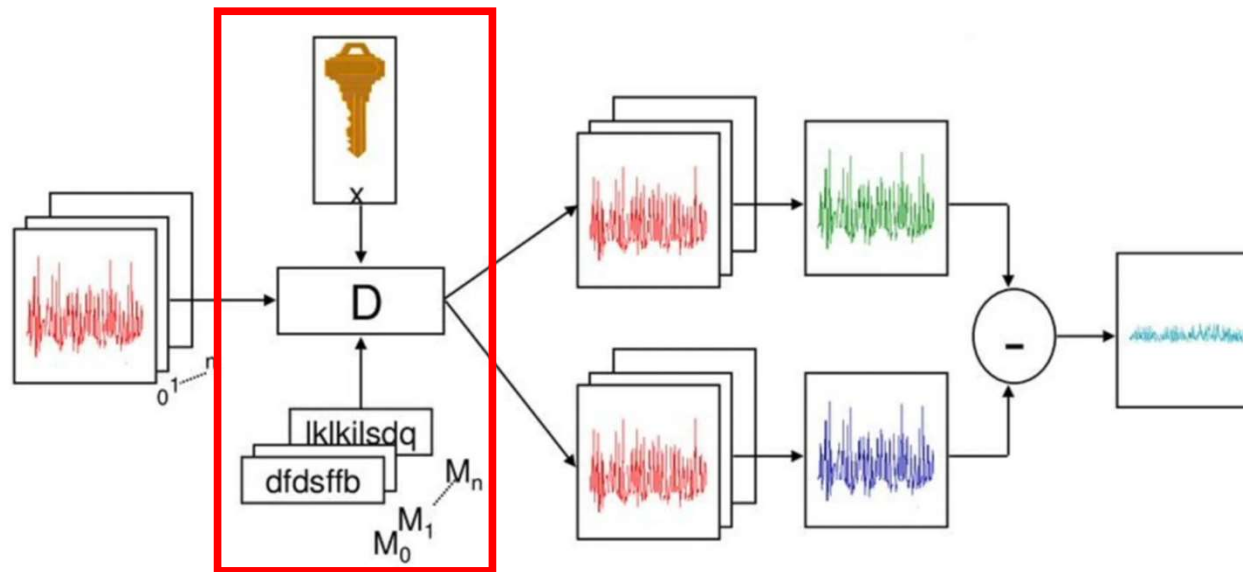
1. 다수의 임의 평문을 입력하여 소비전력을 측정

128-bit input	0	1	2	3	4	5	6	...	Sample size
99D56B ...	-0.0869	0.00097	0.02636	-0.1484	-0.0234	-0.0263	-0.0751		-0.1484
B6B6BE ...	-0.0957	-0.0009	-0.0234	-0.1474	-0.1435	-0.0361	-0.0810		-0.1298
2A5626 ...	-0.0820	-0.0097	0.04003	-0.1367	-0.0556	-0.0302	-0.0839		-0.1406

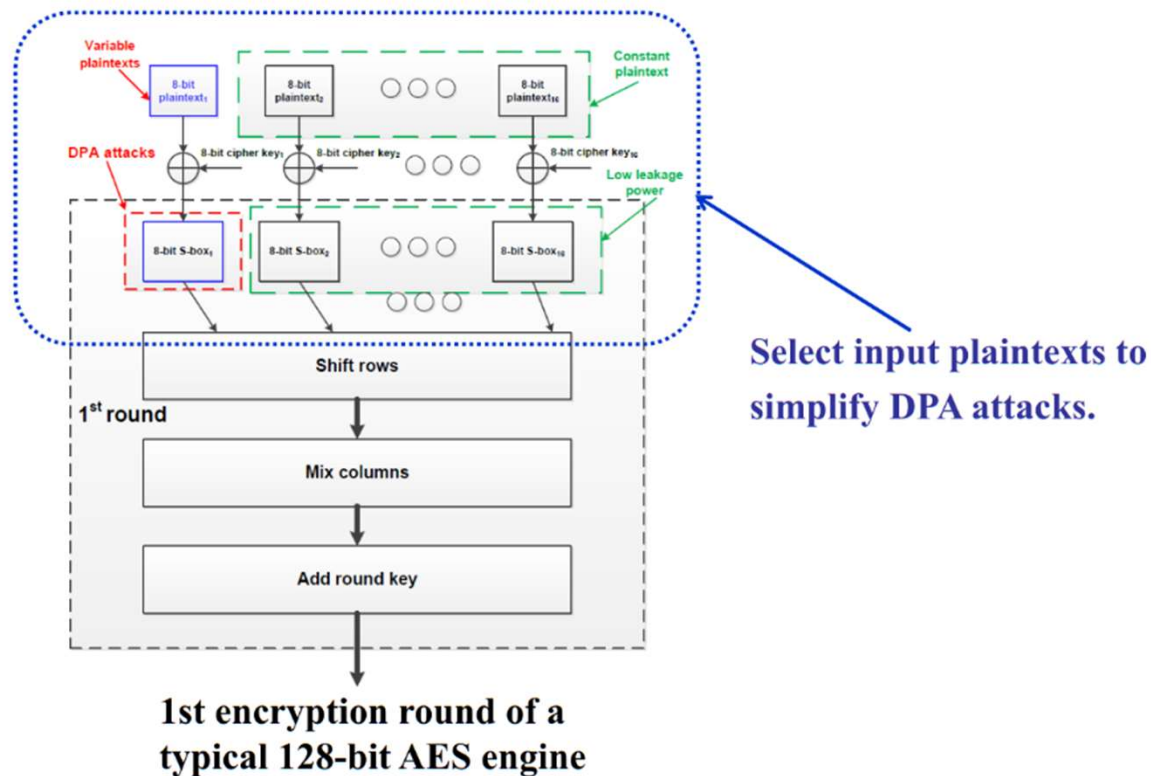
2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

2. 추측 키(1 byte)와 평문을 이용하여 중간값 연산 ex. $Sbox(p[k][i] \oplus key[i])$



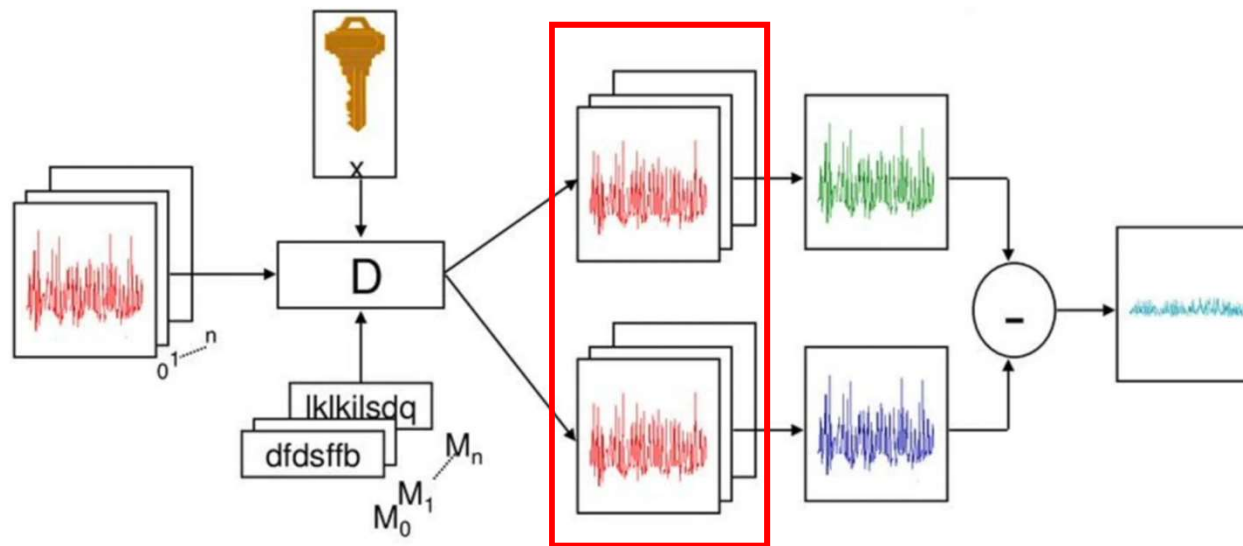
2. DPA (Differential Power Analysis)



2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

3. 중간값의 hamming weight를 계산하여 값에 따라 분류



2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정 예시 (평문 1byte = 0xAB)

2. 추측 키(1 byte)와 평문을 이용하여 중간값 연산 ex. $Sbox(p \oplus key)$

현재 계산하고 있는 추측 키가 0xC2 이고, $Sbox(0xAB \oplus 0xC2) = 0x54$ 라 하자.

3. 중간값의 hamming weight를 계산하여 값에 따라 분류

0x54(01010100) 이므로 $HW(0x54) = 3$ 이다.

$HW(Sbox()) = 4$ 를 기준으로 Small group 과 Big group 로 분류

2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

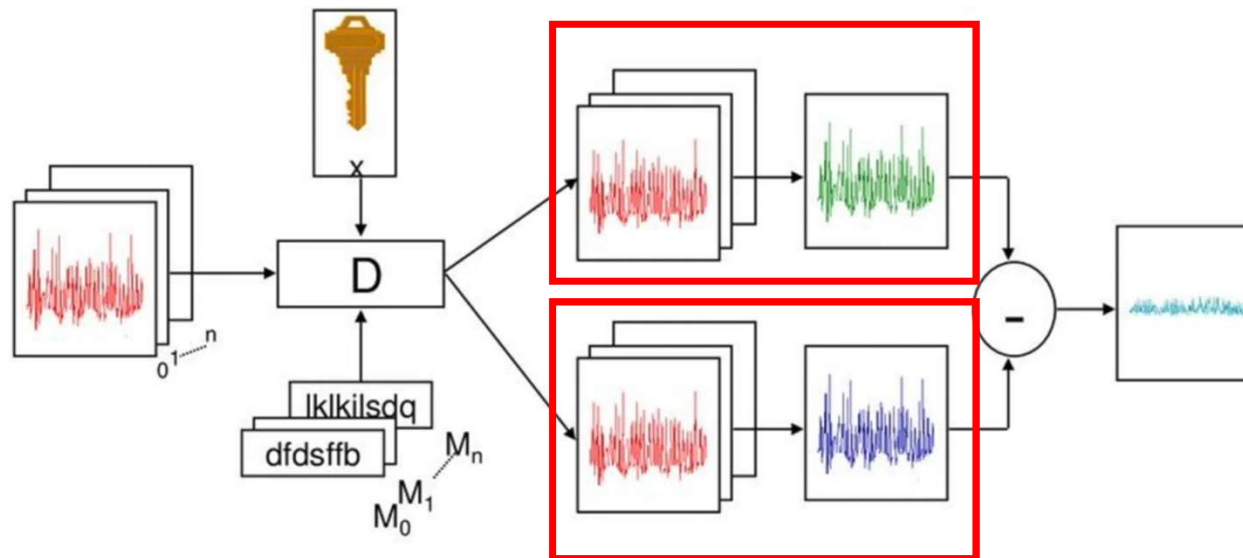
3. 중간값의 hamming weight를 계산하여 값에 따라 분류

```
for i in range(NB_TRACES):
    hw = getHammingWeight(sbox[possible_keys[possiblekey] ^ plaintexts[i][key_idx]])
    if hw >= 4:
        #addTrace(g1, traces[i])
        for j in range(NB_SAMPLES):
            g1[j] += traces[i][j]
        big += 1
    else :
        #addTrace(g2, traces[i])
        for j in range(NB_SAMPLES):
            g2[j] += traces[i][j]
        small += 1
```

2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

4. 양분한 데이터 그룹 각각 평균 소비전력을 구한다.



2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

4. 양분한 데이터 그룹 각각 평균 소비전력을 구한다.

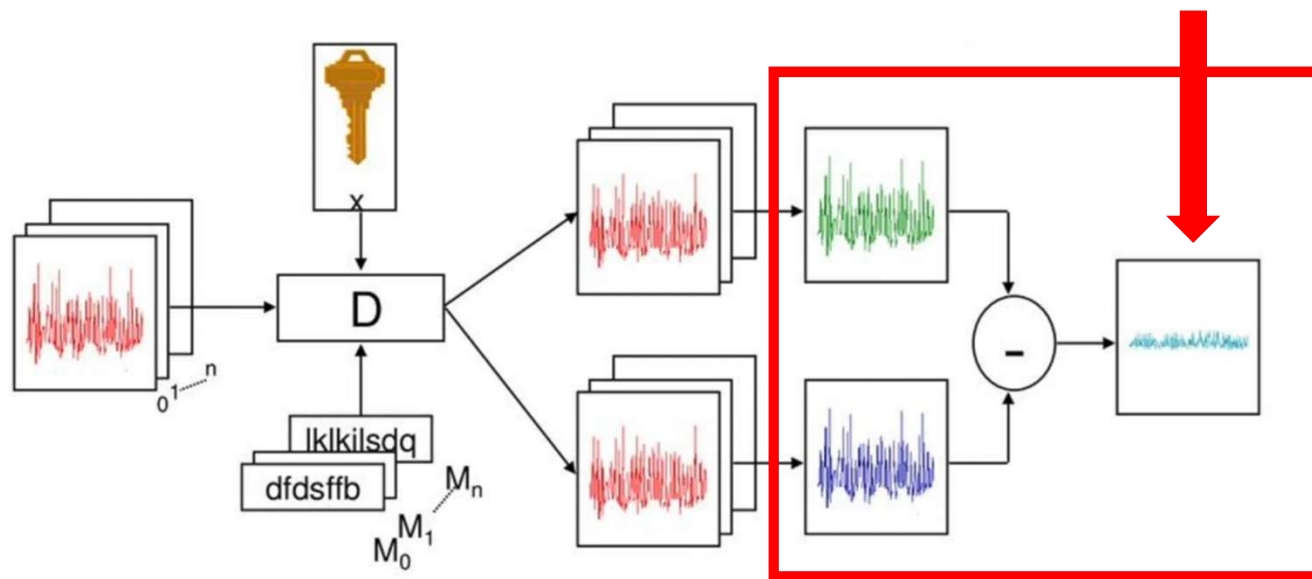
```
if big > 0:
    #calAverage(g1, big)
    for i in range(NB_SAMPLES):
        g1[i] /= big
if small > 0:
    #calAverage(g2, small)
    for i in range(NB_SAMPLES):
        g2[i] /= small
```

2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

5. 차분 신호를 계산

추측 키 0xC2에 대한 차분 신호



2. DPA (Differential Power Analysis)

블록암호에 대한 DPA 공격 과정

5. 차분 신호를 계산

```
for i in range(NB_SAMPLES):  
    res[i] = math.fabs(g1[i] - g2[i])  
  
diffs[key_idx][possiblekey] = max(res)
```

추측 키 0xC2

차분신호의 최대값

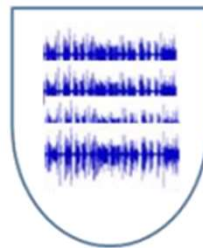
2. DPA (Differential Power Analysis)

여러개의 추측키에 대해 차분파형을 생성하여 비교

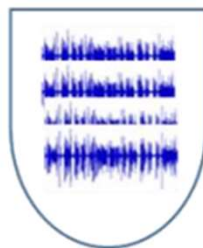
HW는 bit 1의 수
예) 11001100 = HW(4)

KEY GUESS

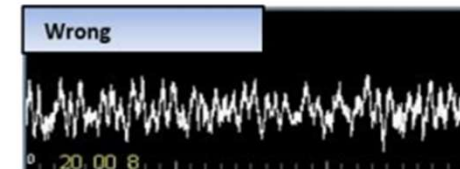
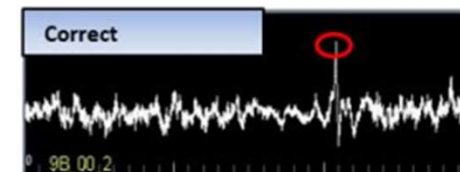
HW 0,1,2,3,4



HW 5,6,7,8



—



2. DPA (Differential Power Analysis)

여러개의 추측키에 대해 차분파형을 생성하여 비교

Key	0x00	0x01	0x02	0x03	0x04	...	0xFD	0xFE	0xFF
max	0.03	0.01	0.07	0.08	0.05	...	0.73	0.02	0.05

2. DPA (Differential Power Analysis)

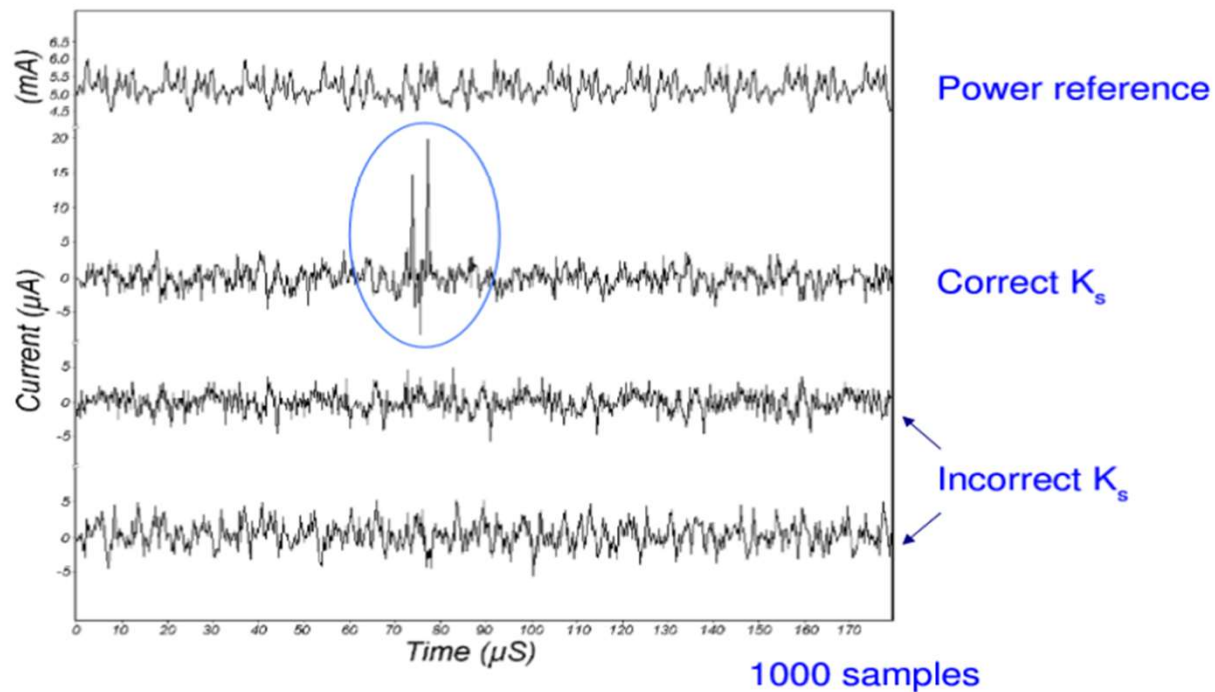


Figure 8. DPA traces, one correct and two incorrect, with power reference [7].

3. CPA (Correlation Power Analysis)

- 다수의 파형과 Hamming Weight 간의 상관 관계를 계산하여 비밀 키를 추출하는 방법.
- Hamming Weight 집단과 소비 전력 집단 간 가지는 **선형적 관계**를 기반으로 공격.

3. CPA (Correlation Power Analysis)

- 다수의 임의 평문을 입력하여 소비 전력 측정
- 추측한 키와 평문을 이용하여 중간값을 Hamming Weight 계산
- 측정한 소비 전력과 Hamming Weight 간의 상관 관계 계산
- 상관도가 가장 높게 나오는 추측키 = 올바른 키

3. CPA (Correlation Power Analysis)

CPA 에서 사용되는 Pearson 상관 계수

$$\rho(X, Y) = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E(X)^2} \cdot \sqrt{E(Y^2) - E(Y)^2}}$$

ρ 의 범위는 $-1 \leq \rho \leq 1$

ρ 가 1에 가까우면 양의 선형관계

ρ 가 -1에 가까우면 음의 선형관계

ρ 가 0에 수렴하면 선형관계를 파악하기 힘들.

3. CPA (Correlation Power Analysis)

Pearson 상관 계수

$$\rho(X, Y) = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E(X)^2} \cdot \sqrt{E(Y^2) - E(Y)^2}}$$

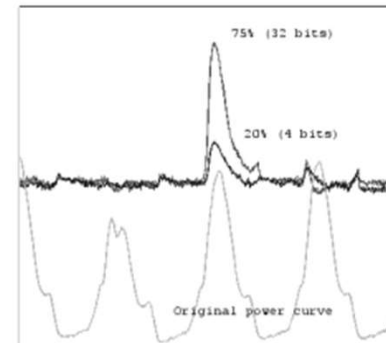


Fig. 2. Two correlation peaks for full word (32 bits) and partial (4 bits) predictions. According to theory the 20% peak should rather be around 26%.

3. CPA (Correlation Power Analysis)

128-bit input	0	1	2	3	4	5	6	...	Sample size
99 D56B	-0.0869	0.00097	0.02636	-0.1484	-0.0234	-0.0263	-0.0751		-0.1484
... B6 B6BE	-0.0957	-0.0009	-0.0234	-0.1474	-0.1435	-0.0361	-0.0810		-0.1298
... 2A 5626	-0.0820	-0.0097	0.04003	-0.1367	-0.0556	-0.0302	-0.0839		-0.1406
...
58 A845	-0.0889	-0.0063	0.02001	-0.1562	-0.1328	-0.0328	-0.0821		-0.1539
...									
Input of 1byte	0x99	0xB6	0x2A	0x58
HW(sbox)	0	1	3	6

3. CPA (Correlation Power Analysis)

```
def correlationMaxKeyByte(possible_key, plaintexts, idx, traces):  
    distance = [0 for _ in range(NB_TRACES)]  
    for i in range(NB_TRACES):  
        distance[i] = getHammingWeight(sbox[possible_key ^ plaintexts[i][idx]])  
  
    corrcoeffs = [0.0 for _ in range(NB_SAMPLES)]  
    for i in range(NB_SAMPLES):  
        corrcoeffs[i] = correlationPearson(distance, traces, i)  
    #print(possible_key, max(corrcoeffs))  
  
    return max(corrcoeffs)
```

3. CPA (Correlation Power Analysis)

```
def CPA(plaintexts, traces, key_estimation):
    corrcoeffs = [[0.0 for __ in range(POSSIBILITES)] for _ in range(NB_SUBKEYS)]
    for idx in range(NB_SUBKEYS):
        print("start subkey {}".format(idx))
        for i in range(POSSIBILITES):
            corrcoeffs[idx][i] = correlationMaxKeyByte(possible_keys[i], plaintexts, idx, traces)

        print("finish subkey {}".format(idx))

    for i in range(NB_SUBKEYS):
        key_estimation[i] = possible_keys[corrcoeffs[i].index(max(corrcoeffs[i]))]
```

3. CPA (Correlation Power Analysis)

- **비선형 연산**(S-box 연산) 에서 Pearson 상관계수를 통한 **선형성**을 판단.
- 선형성이 가장 높게 나타나는 key가 비밀키일 확률이 높다.