# Side channel attack

https://youtu.be/qqF6Z333ykQ

Side Channel Attack

Timing Attack

Power Analysis (SPA, DPA, CPA)

Template Attack

# Side Channel Attack

- **기존의 암호 분석 방법**
  : 차분 분석, 선형 분석 등
- **표준 암호 알고리즘들은 거의 기존 암호 분석 방법에 대응할 수 있도록 잘 설계**
  → 키를 찾거나 평문을 알아내는 것이 어려움
- **부채널 정보** (전력, 소리, 시간, 전자파 등)을 활용하여 통계적으로 분석할 경우 분석 가능
  → 암호 알고리즘의 암호학적으로 안전하게 설계되더라도,
  실제 구현된 후 실행되는 과정에서 발생하는 **부채널 정보를 활용할 경우 안전하지 않음**
- 부채널 공격에 대한 대응 기법 또한 연구되고 있으나 이에 대한 부채널 공격 또한 계속 연구

# Side Channel Attack 종류

- **Timing Attack**

  데이터에 따라 연산 소요 **시간**이 다름 → 비밀 정보 복원

- **Power Analysis**

  SPA, DPA, CPA 등

  연산에 사용되는 **소비 전력이** 다름 → 여러 개 수집하여 통계적 분석 → 비밀 정보 복원

- **Template Attack**

  타겟 보드에서 전력 파형을 미리 여러 개 수집하여 **template 구성** → 통계적 분석 → 비밀 정보 복원

# Side Channel Attack – Timing Attack

- 데이터에 따라 연산 시간이 다름을 활용 (캐시 타이밍 어택 등)
- 암호화 동작 시 걸리는 시간을 분석
  → 암호 연산의 실행 시간은 키와 연관된 정보에 의존
- 개인키에 대한 연산에 걸리는 시간을 분석할 경우 RSA 키를 찾아내고, DH의 지수를 찾아낼 수 있다고 함
- 다른 부채널 분석에 비해 효과가 크지 않음

# Side Channel Attack – Simple Power Analysis

- **단일 파형으로 분석**

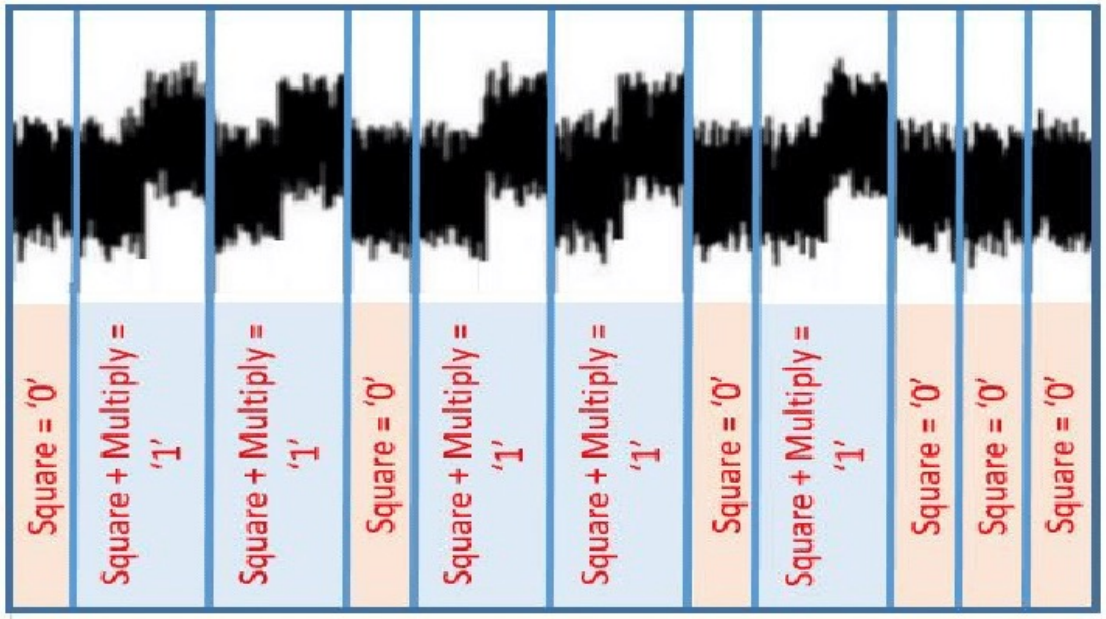
  → 공격자는 공격 지점의 **연산 과정 및 구현 방법을 정확히** 알아야 함 (어떤 명령어가 어떤 경우 수행되는지)

  암호화 1번 수행 시, 키 비트를 직접 찾음

  ex) RSA square and multiply 연산

  → data가 1이면 Square and mul, 0이면 square만 수행

  → 단일파형으로도 유추가 가능



# Side Channel Attack – Differential Power Analysis

- AES 등에서 단일 파형 정보로 라운드같은 정보를 찾는 건 가능 but 실제 키 값을 알아내기 어려움

  **→ 파형을 여러 개 모아서 통계적 분석 → 구현 방법을 정확히 모를 경우도 가능**

  - 암호 알고리즘 동작 시 저장되는 비트 값이 **1일 때와 0일 때의 소비전력이 다르다**는 사실을 이용하여 분석

- DPA 파생

  Correlation Power Analysis (CPA), High-Ordering DPA 등 존재

# Side Channel Attack – Differential Power Analysis

- **Hamming Weight Model**

  - Hamming Weight (HW) : 1의 개수

  → 1일 때의 소비 전력 > 0일 때의 소비 전력

  → 0110 이라는 데이터가 있을 때, 0000에서 0110으로 변화하는 것이므로 HW는 2

  - HW 값과 전력 소모는 비례하므로 HW를 소비 전력과 관련 지음

  - 8비트 프로세서인 경우, 8비트 단위이므로 HW의 최대 값은 8이며, 9개의 경우 존재 (0~8)

- $\mathbf{P_{Total} = P_{operation} + P_{data} + P_{noise} + P_{constant}}$ (총 소비 전력)

  **→ 수행하는 연산, 사용되는 데이터, 노이즈, 기본 소비전력에 의해 총 전력이 결정**

  → noise : 연산과정에서 추가될 수 밖에 없음

  해당 노이즈들을 제거하여 더 정확한 값을 얻기 위해 auto-encdoer를 통한 노이즈 제거 같은 연구 진행

  데이터에 따라 약간씩 전력 값이 차이 나는데, 노이즈가 끼면 다른 데이터가 같은 HW 값으로 보일 수 있음

  → noise 제거함으로써 공격 성공률 상승

# Side Channel Attack – Differential Power Analysis

- **HW model**

  - 전력파형을 얻을 경우, 다음과 같은 파형 생성됨

    → **세부적인 값 차이 : 데이터에 의한 차이**

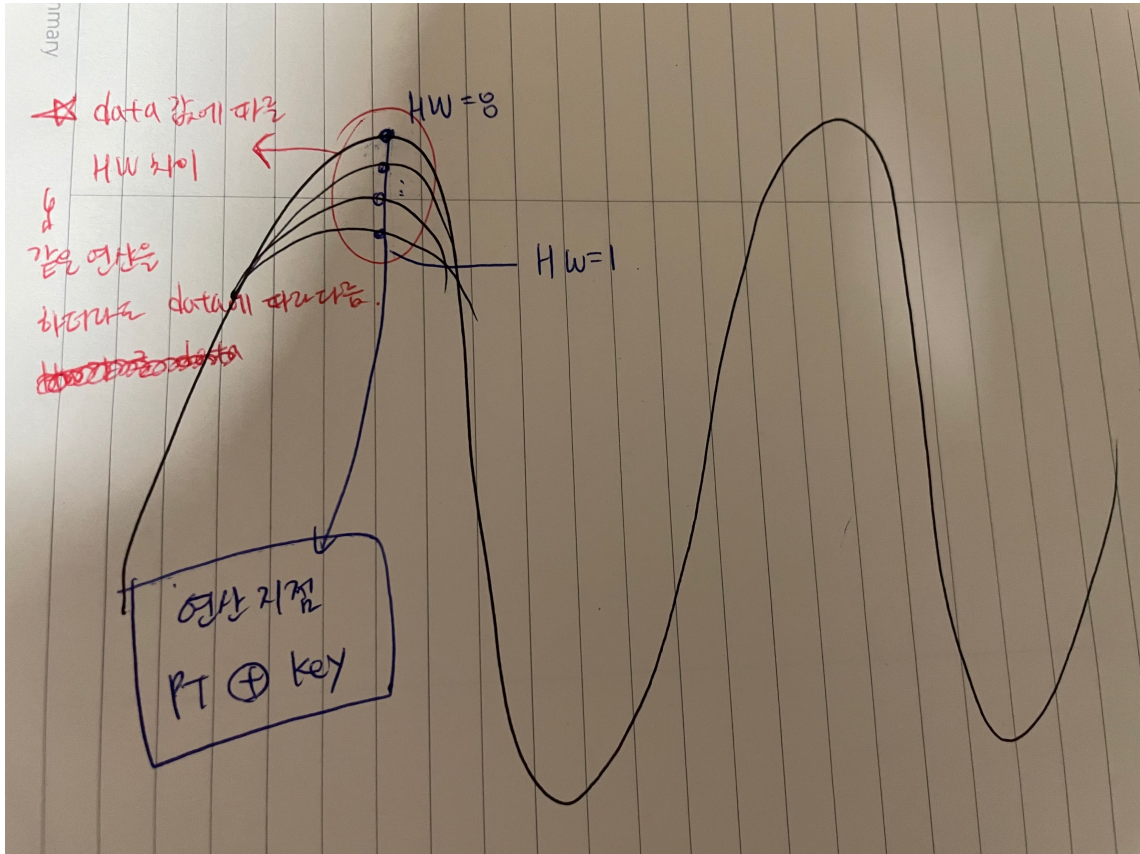
    **그래프 위상 (형태) 차이 : 연산 (명령어) 종류에 따라 결정**(전력 소비 높은 연산자면 HW값이 높고 반대면 낮음)

- 전체 그래프 패턴을 보면, 같은 패턴이 2번 반복

  → 명령어 패턴을 알 수 있는 것임,

  AES의 라운드 반복, 각 라운드의 내의 함수 **패턴 파악 등이 가능**

  이러한 점을 이용하여 **알고리즘 역공학 가능**



# Side Channel Attack – Differential Power Analysis

- **목적 : Key 찾기**
- **가정**

  1) **알고 있는 것** : 무작위 Plaintext, 각 pt에 대한 전력 소비량, 그에 따른 output

  2) Known plaintext ⊕ key = Cipher에서의 **Key는 알 수 없으며, 고정된 Key**

  → PT, CT만 가지고 하는 게 black box model인데 DPA는 파형 정보를 추가적으로 앎 → gray box model

- **DPA 과정**

  - **소비전력 여러 개 수집 → 모델링 → 통계적 분석**

  - 0001 이랑 0011 이 있을 경우 HW가 1 차이라서 구분이 어렵고 noise가 추가될 경우 같은 값으로 보일 수도 있음

    → 그러나 많은 데이터를 모은 후 **통계적 분석을 하게 되면, noise가 제거된 것과 비슷**

  - 또한, 단순 XOR은 HW가 유사한 게 많으므로 동일 input에 대해 여러 값이 가능한 **비선형 연산을 모델링 타겟**으로 함

  00 xor 00 → HW 0 // 01 xor 00 → HW 1 // 10 xor 00 → HW 1 // 11 xor 00 → HW 2

  → 이런 경우, HW가 0,1,2 뿐이라서 구분이 잘 안 되고, 데이터가 많이 필요

# Side Channel Attack – Differential Power Analysis

- **세부 과정**
  1. **무작위 pt 준비** (pt는 10개로 예시)
  2. **Key**는 알 수 없으므로 **전수조사** 필요 (8비트의 경우 0~255)
  3. **modeling** : 10개의 input에 대해 각 Key를 전수조사하여 HW 값 구함 (**무작위 input에 대한 HW 수집**)
  4. 8비트이므로 최대 HW는 8 → **8을 기준으로 반으로 나눔** (0~3이면 0 (Low), 4~8이면 1 (High))

     → **HW 기준으로 input을 그룹으로 나눔**

     → 즉, 무작위 pt에 대해 연산 → input당 256개의 HW 산출 → HW에 따라 input trace를 두 그룹으로 분할
  5. 3~4번을 모든 pt에 대해 반복 수행 (여기서는 10번)
  6. 두 그룹에 대해 각각 평균낸 후 차를 구함

$Sbox[pt \oplus k]$의 HW

| | Key 0 | … | Key 255 |
|---|---|---|---|
| Input 0 | HW 8 | … | HW 0 |
| … | … | … | … |
| Input 9 | HW 1 | … | HW 2 |

modeling

pt ⊕ key = cipher, 0000에서 cipher로 이동한다고 생각
Key = 0000일 경우,
0001 ⊕ 0000 = 0001 → HW = 1
0011 ⊕ 0000 = 0011 → HW = 2
이런 식으로 무작위 pt에 대해 모든 HW 수집

# Side Channel Attack – Differential Pc

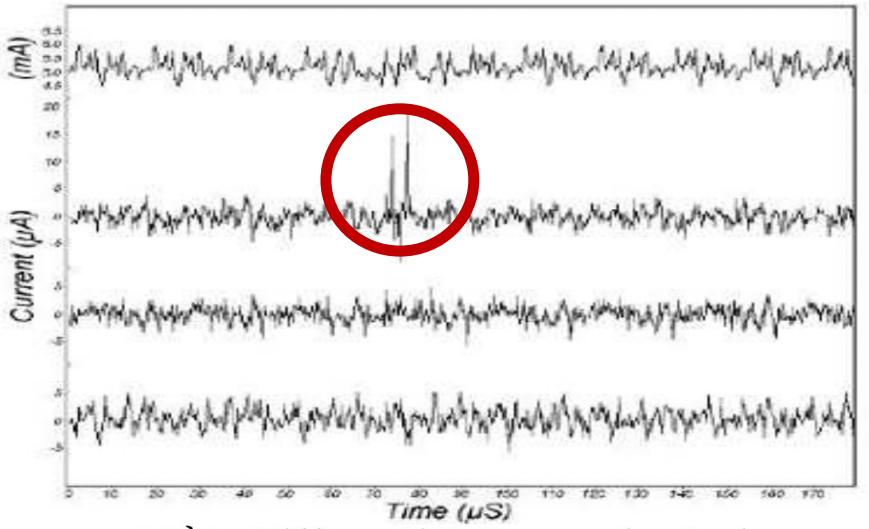


그림4. Differential Power Analysis

- 세부과정 2

  **키를 전수조사하면 옳은 키가 있을 것이고, 나머지 틀린 키 존재**

  **ex) 실제 파형은 특정 연산 지점에서 HW가 8이 나와야 됨**

  그럼 앞의 4번 과정에서

  **옳은 키의 경우 HW가 8**이 나옴 → 그럼 해당 input trace는 그룹 high(1)로 감

  **틀린 키의 경우 HW가 예를 들어 1**이 나옴 → 해당 input trace는 그룹 Low (0)으로 감 → **그룹 분할이 잘못됨**

  → 이럴 경우, 각 그룹에 속한 파형들의 평균을 냈을 때,

  HW 높은 그룹은 해당 파형 값의 평균이니까 높아야 되고 낮은 그룹은 낮아야 되는데

  **잘못 분할될 경우 두 그룹 간의 차이가 별로 없게 됨**

  → 즉, 평균의 차이가 별로 없고 그 차를 구하면 작아짐

  → 그러나 **옳은 키에 대해서는 두 그룹의 차이가 큼** (하나는 평균이 높고 하나는 낮음)

  즉, 올바른 키에서는 특정 연산 부분에 대해 두 그룹의 차가 크게 나옴

  (해당 연산에 대해 모델링 한 것이므로 그 시점의 파형만 솟고, 나머지는 그냥 data에 대한 HW로 구한 거라 차이 x)

# Side Channel Attack – Correlation Power Analysis

- DPA에서 구한 HW 값이 소비 전력을 모델링한 값이라 연관이 있으므로,
  **실제 파형에서 소비전력 값이 HW와 상관관계가 있을 것**

- DPA와 동일하게 모델링

| | Key 0 | … | Key 255 |
|---|---|---|---|
| Input 0 | HW 8 | … | HW 0 |
| … | … | … | … |
| Input 9 | HW 1 | … | HW 2 |

modeling

| | point 0 | … | point 99 |
|---|---|---|---|
| Input 0 | 0.8 | … | 0.4 |
| … | … | … | … |
| Input 9 | 0.1 | … | 0.1 |

파형에서 연산 지점을 100개라고 할 때,
각 point에서의 전력소비량

- **각 포인트의 전력 소비량에 대해 상관계수 분석**
  → 즉 여러 인풋에 대한 전력소비량과 여러 인풋에 대한 HW 값의 상관관계를 분석
  → 피어슨 상관계수 사용 → 높을수록 +1, -1을 나타내고, 낮을수록 0에 가까움
  → 1인 경우 선형 관계이며 상관계수가 높다 → 키 찾음

# Side Channel Attack – Template Attack

- **DPA보다 더 강력한 가정이 필요**
- 공격자가 타겟 디바이스를 완전히 복제해서 가지고 있다고 침 (**key값을 앎**, DPA는 키를 모름)
  → 즉, 평문 및 비밀키 설정이 가능
- 이를 통해 **중간 값을 계산한 후, 사전처럼 다 가지고 있음**
- **수집 파형의 template (평균 및 공분산)을 작성해 둔 후, 나중에 파형을 넣어서 매칭**시키는 공격

# Side Channel Attack

- AES 128의 경우 S-box 연산이 8비트 단위이고, 그게 16개씩 (16바이트 단위) 연산
  총 2^128의 전수조사가 필요 but 모델링 자체를 8비트 단위로 했고 공격이 그렇게 수행되기 때문에,
  **총 2^8 * 16번이면 공격 가능**

state

| 8-bit | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

- 그러나 공격 지점을 특정 짓지 못할 경우, 연산량이 많아지므로 **포인트를 잘 찾아야함**
  → 1000개의 포인트에 대해 전력 분석 수행 시 2^8 * 16 * 1000 만큼의 전수조사 필요
  → SPA 등으로 전체적인 패턴 (라운드, 라운드 내부 연산)을 파악
  → 1라운드의 S-box 부분 등과 같이 공격 지점 좁혀야 함

감사합니다.