부채널 파형 데이터를 사용한 머신러닝 암호 분류

권혁동 김현지 서화정
한성대학교

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

서론

부채널 분석

제안 모델

결론

서론

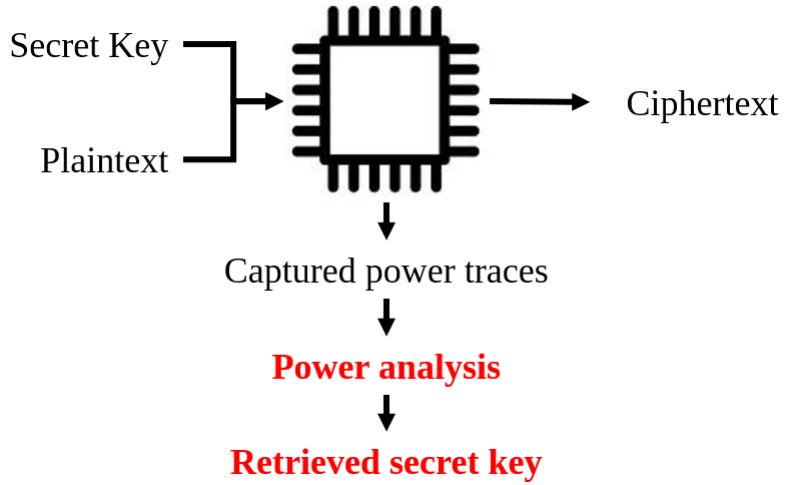
- 부채널 분석에는 분석에 많은 시간이 소요
- 암호 알고리즘 별로 공격 방법이 상이함
  - 하드웨어 기기 상에서 어떤 **암호 알고리즘이 가동 중인지 알 수 없음**

- 부채널 분석 데이터를 사용한 분류 모델 제안 및 구현
  - **제공된 부채널 데이터만을 가지고 어떤 암호 알고리즘인지 분류**

- 하이퍼 파라미터 조절을 통해 작성한 모델의 효율성 분석

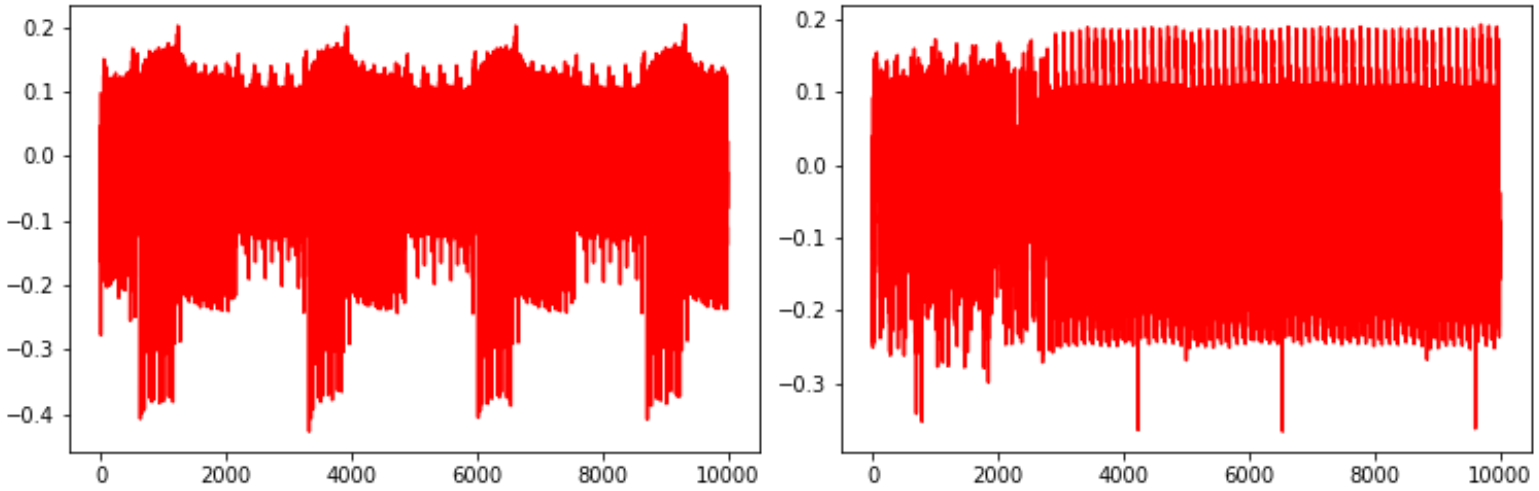
부채널 분석

- 1996년 P. Kocher의 제안에서 발전된 공격
- 암호 알고리즘 자체를 공격하지 않음
- 알고리즘 연산 시 발생하는 **부채널 정보 활용**
  - 빛, 소리, 전자기파, 열...
- 전력량을 분석하는 전력 분석 공격 (Power analysis attack)
  - 단순 전력 분석(Simple power analysis)
  - 차분 전력 분석(Differential power analysis)
  - 상관 관계 분석(Correlation power analysis)



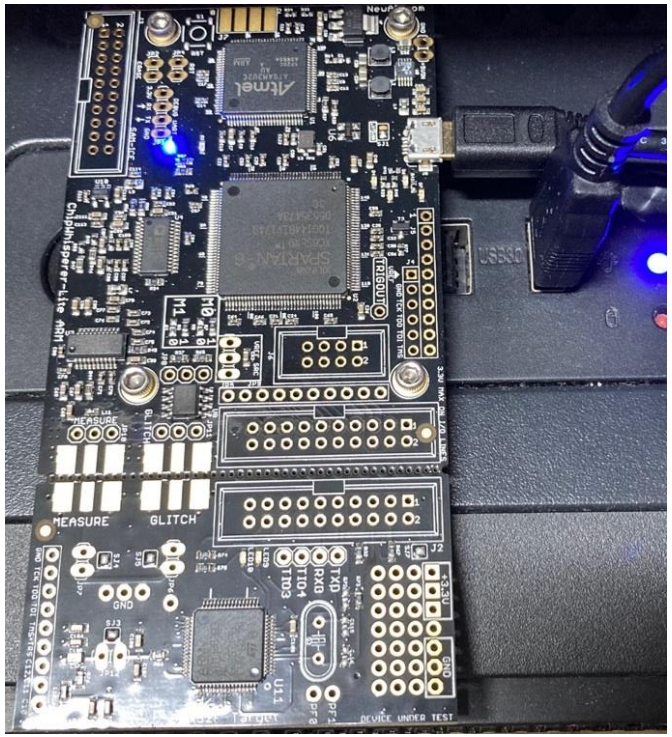
부채널 분석

- 암호 알고리즘 별로 내부 구조가 다르기 때문에 파형의 형태가 다름
  - 따라서 알고리즘마다 **공격 방식이 다름**
- 실험 환경에서는 하드웨어 내부의 알고리즘을 알고 있음
  - 원하는 공격 방식을 선택하기 쉬움
- **실제로는 어떤 알고리즘이 가동 중인지 알 수 없음**
  - 수집된 파형 값을 통해 구분하는 방법



제안 모델

- 부채널 분석에 사용되는 장비는 대체로 고가
- ChipWhisperer 실험용 장비를 사용
  - 연산을 담당하는 Device
  - 파형을 수집하는 Capture
- PC와 연결하여 전용 소프트웨어를 통해 동작



제안 모델

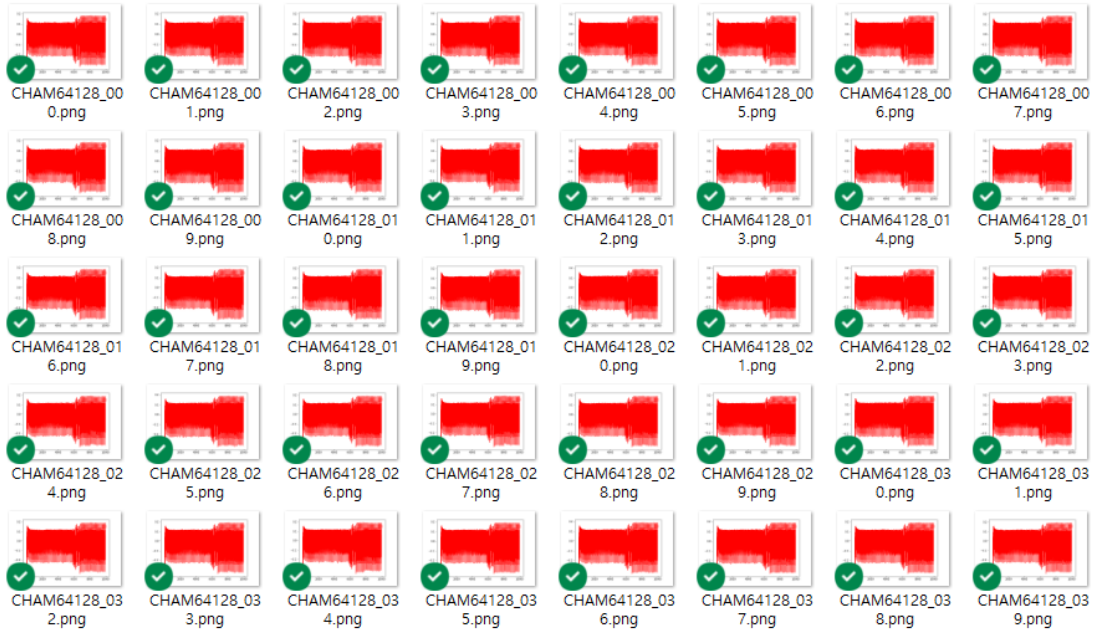
- ChipWhisperer 상에서 파형 캡쳐 코드를 작성
- 대상 알고리즘
  - CHAM64-128
  - CHAM128-128
  - CHAM128-256
  - LEA128
  - LEA192
  - LEA256
  - PIPO64-128
  - PIPO64-256
  - AES128
- **각 알고리즘 별로 5000개씩의 파형을 캡쳐**
  - **총 45,000개**

```
%%bash -s "$PLATFORM" "$CRYPTO_TARGET" "$SS_VER"
cd ../../hardware/victims/firmware/HDserial-CHAM64-128

make PLATFORM=$1 CRYPTO_TARGET=$2 SS_VER=$3
```

```
from tqdm.notebook import tnrange
import numpy as np
import time
import matplotlib.pylab as plt


ktp = cw.ktp.Basic()
trace_array = []
textin_array = []


key, text = ktp.next()

target.set_key(key)

i = 0
TRACE_NUMBER = 5000
scope.adc.samples = 10000

plt.figure()

N = TRACE_NUMBER

for i in tnrange(N, desc='Capturing taces'):
    scope.arm()

    target.simpleserial_write('p', text)

    ret = scope.capture()
    if ret:
        print("Target timed out!")
        continue

    response = target.simpleserial_read('r', 16)

    trace_array.append(scope.get_last_trace())
    textin_array.append(text)

    key, text = ktp.next()

    if i < 3000:
        filepath_png = 'D:₩₩Dropbox₩₩Projects₩₩A
        filepath_npy = 'D:₩₩Dropbox₩₩Projects₩₩A
```

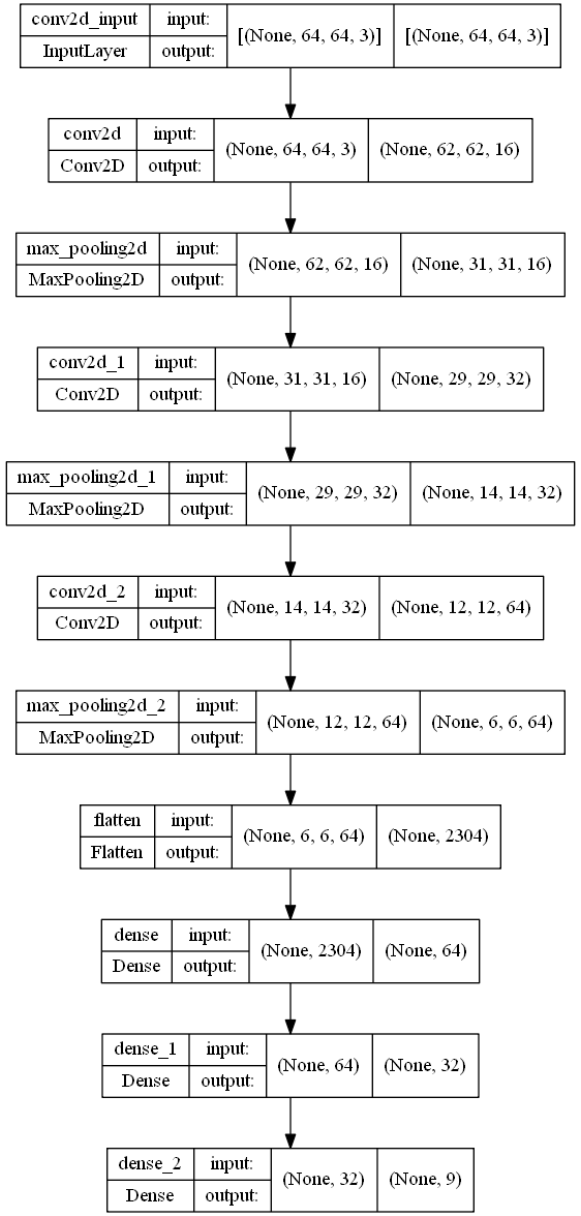
제안 모델

- 파형을 수집하여 **그래프(PNG)**와 값(NPY) 형태로 출력
- 수집한 파형 데이터를 일정 비율로 분리
  - 6(Train) : 2(Validation) : 2(Test)
- 전체 데이터의 수
  - Train: 27,000
  - Validation: 9,000
  - Test: 9,000



제안 모델

- 파형 이미지를 사용하여 학습을 위해 CNN 구축
- Sequential 형태로 레이어를 생성
- 하이퍼 파라미터 조절을 통해 다수의 모델을 생성
  - 뉴런 수 조절, Dropout 레이어 추가
  - 총 4종의 모델 생성



제안 모델

종류	파라미터 수	학습 정확도	학습 손실	검증 정확도	검증 손실	시험 정확도
1	**173,481**	**93.10%**	**0.125**	**94.38%**	**0.108**	**92.19%**
2	173,481 + Dropout	77.90%	0.452	85.00%	0.167	88.75%
3	1,303,305	90.60%	0.152	93.12%	0.136	93.44%
4	1,303,305 + Dropout	90.35%	0.142	92.50%	0.122	90.62%

- 1의 정확도는 전체적으로 높음
- 현 상태로는 1은 좋은 모델이라 평가할 수 있음

제안 모델

종류	파라미터 수	학습 정확도	학습 손실	검증 정확도	검증 손실	시험 정확도
1	173,481	93.10%	0.125	94.38%	0.108	92.19%
2	**173,481 + Dropout**	**77.90%**	**0.452**	**85.00%**	**0.167**	**88.75%**
3	1,303,305	90.60%	0.152	93.12%	0.136	93.44%
4	1,303,305 + Dropout	90.35%	0.142	92.50%	0.122	90.62%

- 하지만 1에 Dropout 레이어를 추가한 2는 정확도가 매우 내려감
  - Dropout은 과적합을 방지하는 역할
- 1은 과적합이 되었다고 판단할 수 있음

제안 모델

종류	파라미터 수	학습 정확도	학습 손실	검증 정확도	검증 손실	시험 정확도
1	173,481	93.10%	0.125	94.38%	0.108	92.19%
2	173,481 + Dropout	77.90%	0.452	85.00%	0.167	88.75%
3	**1,303,305**	**90.60%**	**0.152**	**93.12%**	**0.136**	**93.44%**
4	1,303,305 + Dropout	90.35%	0.142	92.50%	0.122	90.62%

- 3은 1에서 파라미터를 늘린 모델로 좋은 정확도를 보임
  - 시험 정확도가 가장 높음

제안 모델

종류	파라미터 수	학습 정확도	학습 손실	검증 정확도	검증 손실	시험 정확도
1	173,481	93.10%	0.125	94.38%	0.108	92.19%
2	173,481 + Dropout	77.90%	0.452	85.00%	0.167	88.75%
3	1,303,305	90.60%	0.152	93.12%	0.136	93.44%
4	**1,303,305 + Dropout**	**90.35%**	**0.142**	**92.50%**	**0.122**	**90.62%**

- 4는 3의 모델에 Dropout 레이어를 포함한 모델
- 높은 정확도가 거의 유지됨
  - 3은 적합한 모델이라 평가할 수 있음

결론

- **부채널 파형 데이터**를 사용하여 **알고리즘을 판단**하는 모델
- 적은 파라미터를 지니는 모델은 가볍지만 과적합 발생
- **많은 파라미터를 지니는 모델은 높은 정확도를 보유**
  - 하지만 학습에 걸리는 시간이 길며 모델이 무거워짐
- 더 높은 학습률(95%)을 달성하기 위한 파라미터 조절 필요
- **파형 값(NPY) 데이터를 학습한 모델 제시**
  - 파형 그래프(PNG)를 사용한 모델과 비교
  - 어느 데이터를 사용하여 분류하는 것이 효과적인지 비교 가능

Q & A