Detecting Similar Code Segments through Side Channel Leakage in Microcontrollers

(마이크로 컨트롤러의 사이드 채널 누출을 통한 유사한 코드의 세그먼트 감지)

논문 리뷰

https://youtu.be/iFvV8H4EiKo

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

논문 목적

- 마이크로 컨트롤러의 소프트웨어를 표절 하였을 때
  지적재산권을 확인 필요

- 의심스러운 프로그램 코드를 내부 코드 메모리에서 물리적으로 추출
  해야하기 때문에 직접 바이너티 분석을 수행 할 수 없음

➢부채널 전력 분석으로 두 구현된 시스템을 비교 하여
  지적 재산권을 확인

관련 연구1

ATMega8 마이크로 컨트롤러가 명령을 가져올 때 opcode의 해밍 웨이트를 누출한다는 우연한 사실을 사용

- 모든 마이크로 컨트롤러가 연산 코드의 해밍 웨이트를 누출하지는 않음
- 연산 코드의 해밍 가중치는 고유하지고 다른 명령들이 동일한 해밍 웨이트를 가질 수 있음
- 어셈블리 명령어를 교환하는 코드 변환 공격에 대해서는 강력하지 않음

관련 연구2

PIC16F687 마이크로 컨트롤러의 전자기장 트레이스에서
명령어 클래스를구분할 수 있는 분류자를 훈련
해당 명령어를 분해하려고 시도

- 이 방법은 작은 범위에서만 테스트

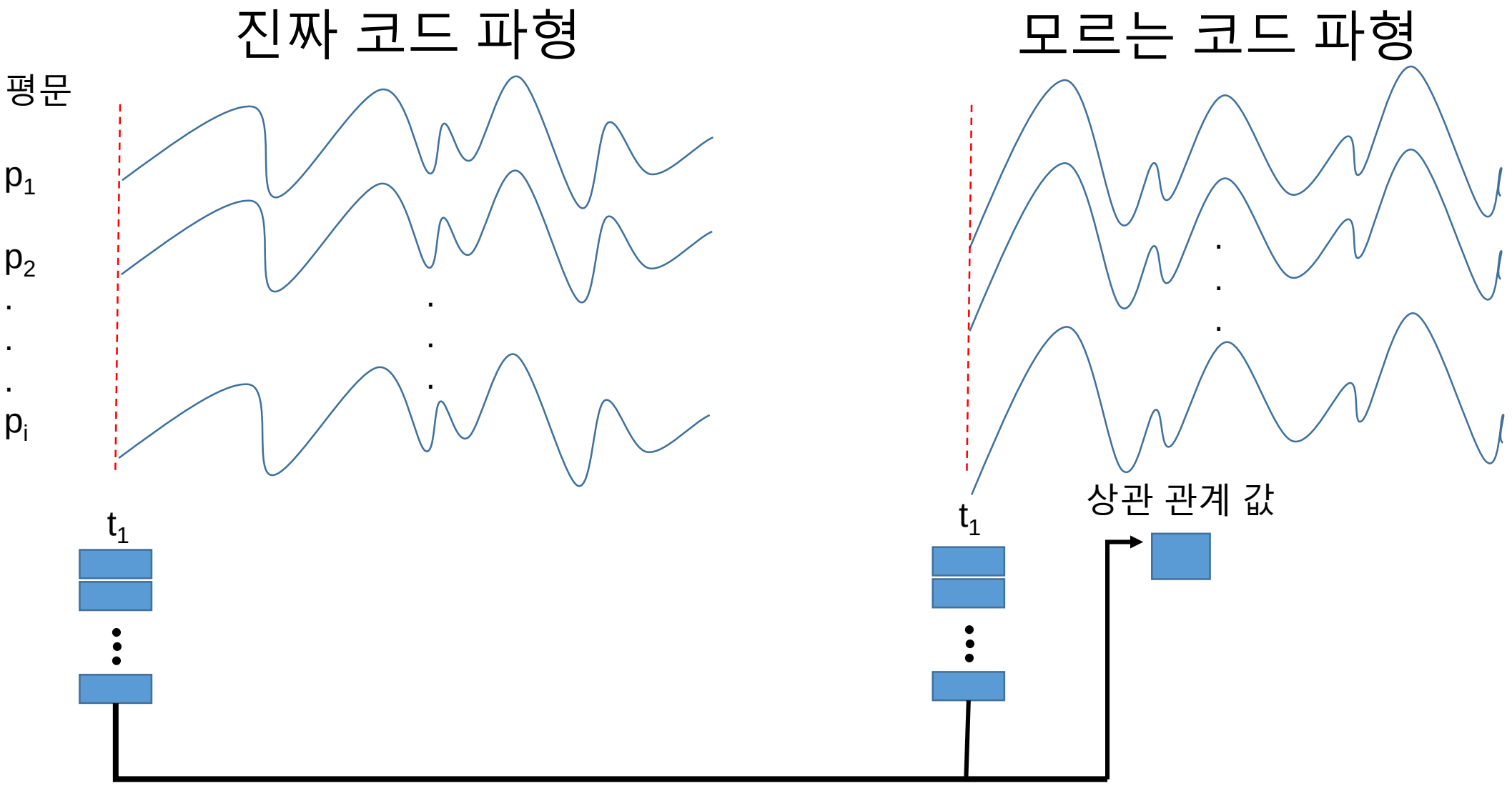
관련 연구3

시간축을 따라 피어슨의 상관 관계를 이용하여
2 개의 파워 트레이스의 유사도를 계산

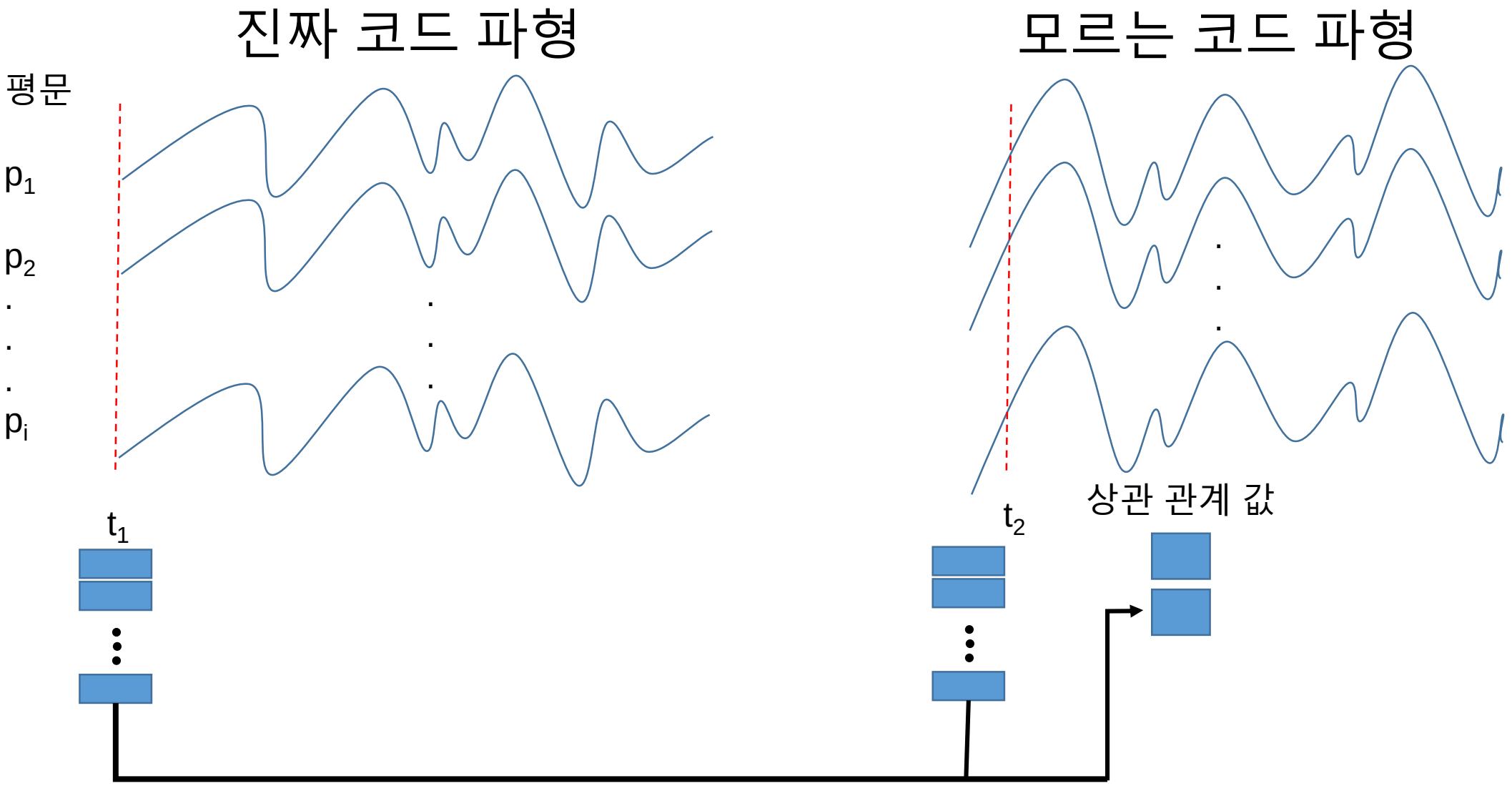
- 제안 된 방법은 공격자가 더미 명령어를 코드에 추가하면 견고하지 않음

➢소프트웨어 표절에대해 전체적인 표절 여부 뿐 아니라 세부적인 표절 확인 가능

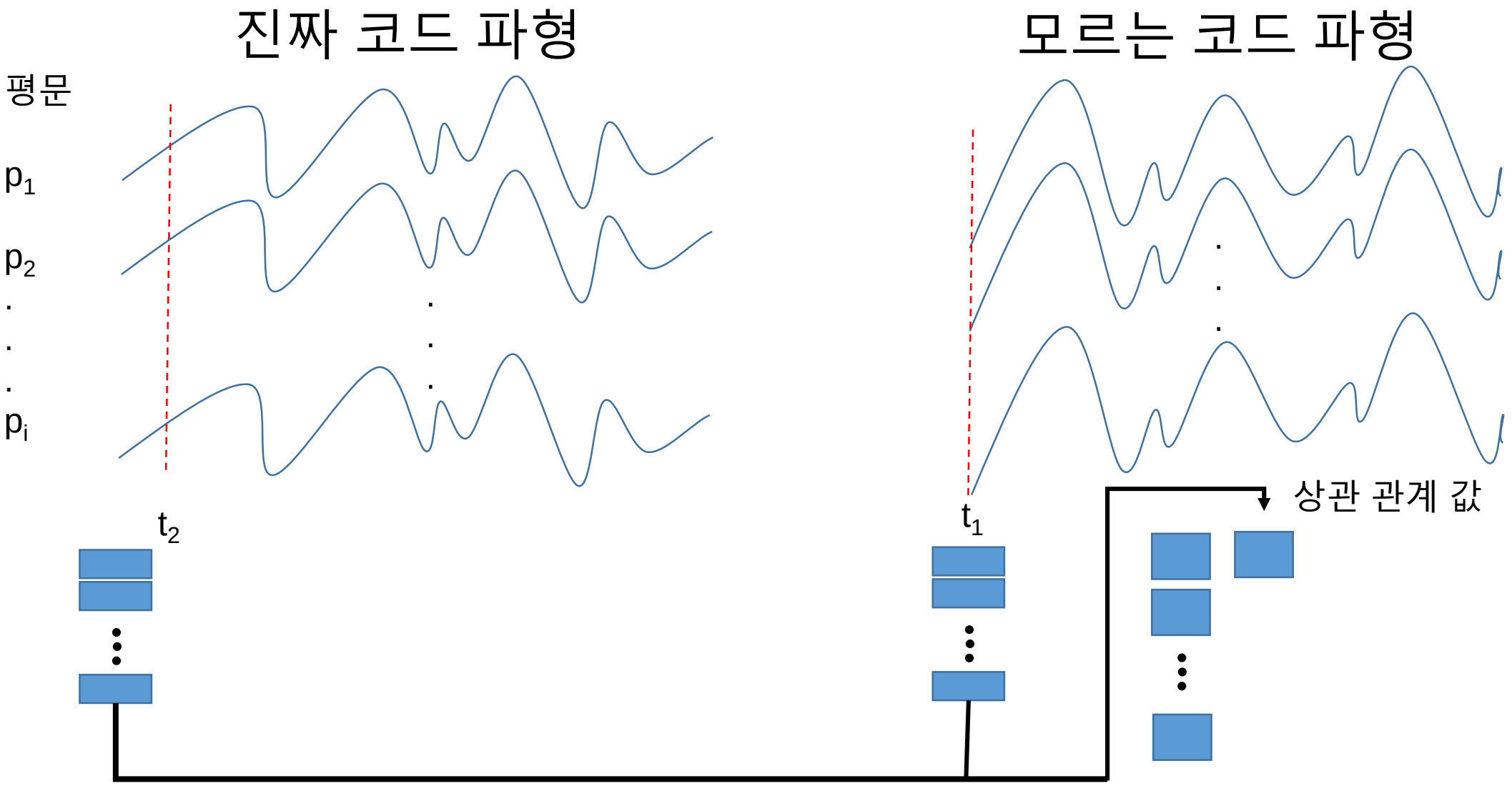
방 법



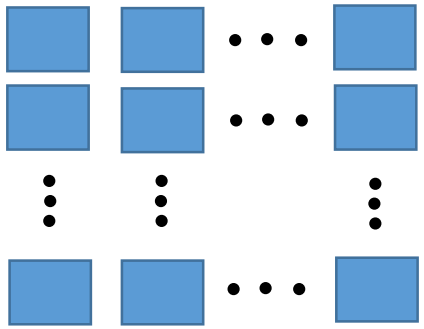
방 법



방 법



방 법



이미지화

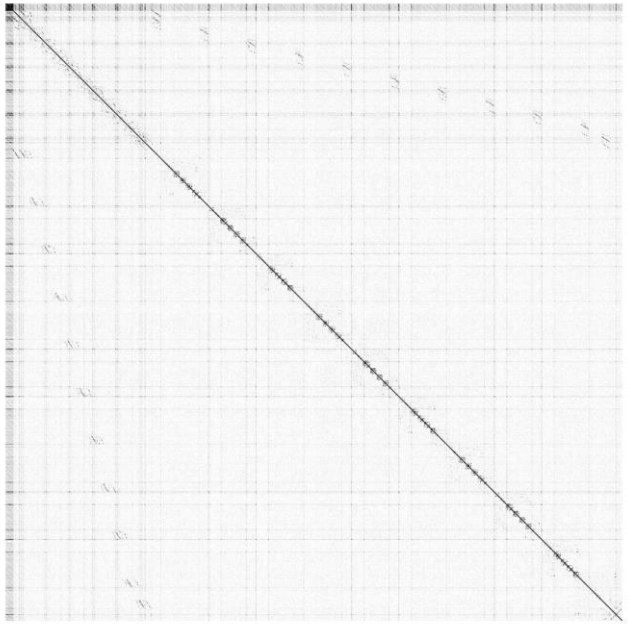
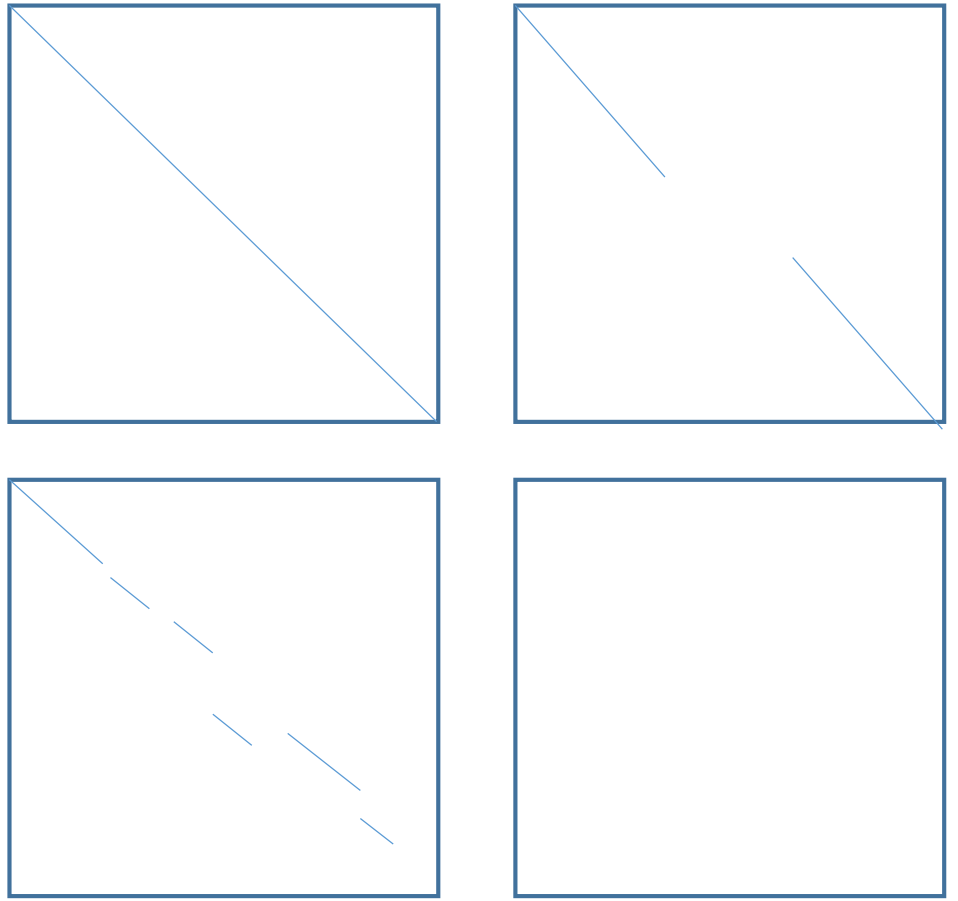
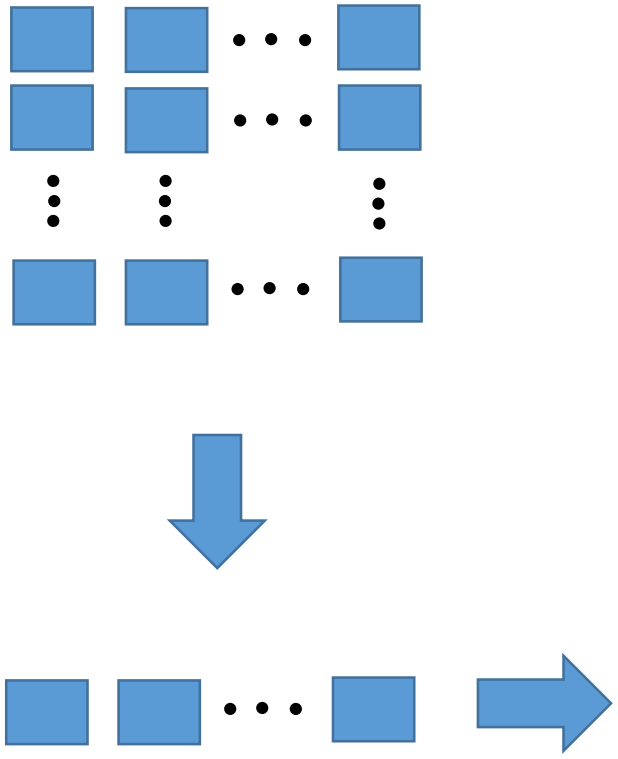


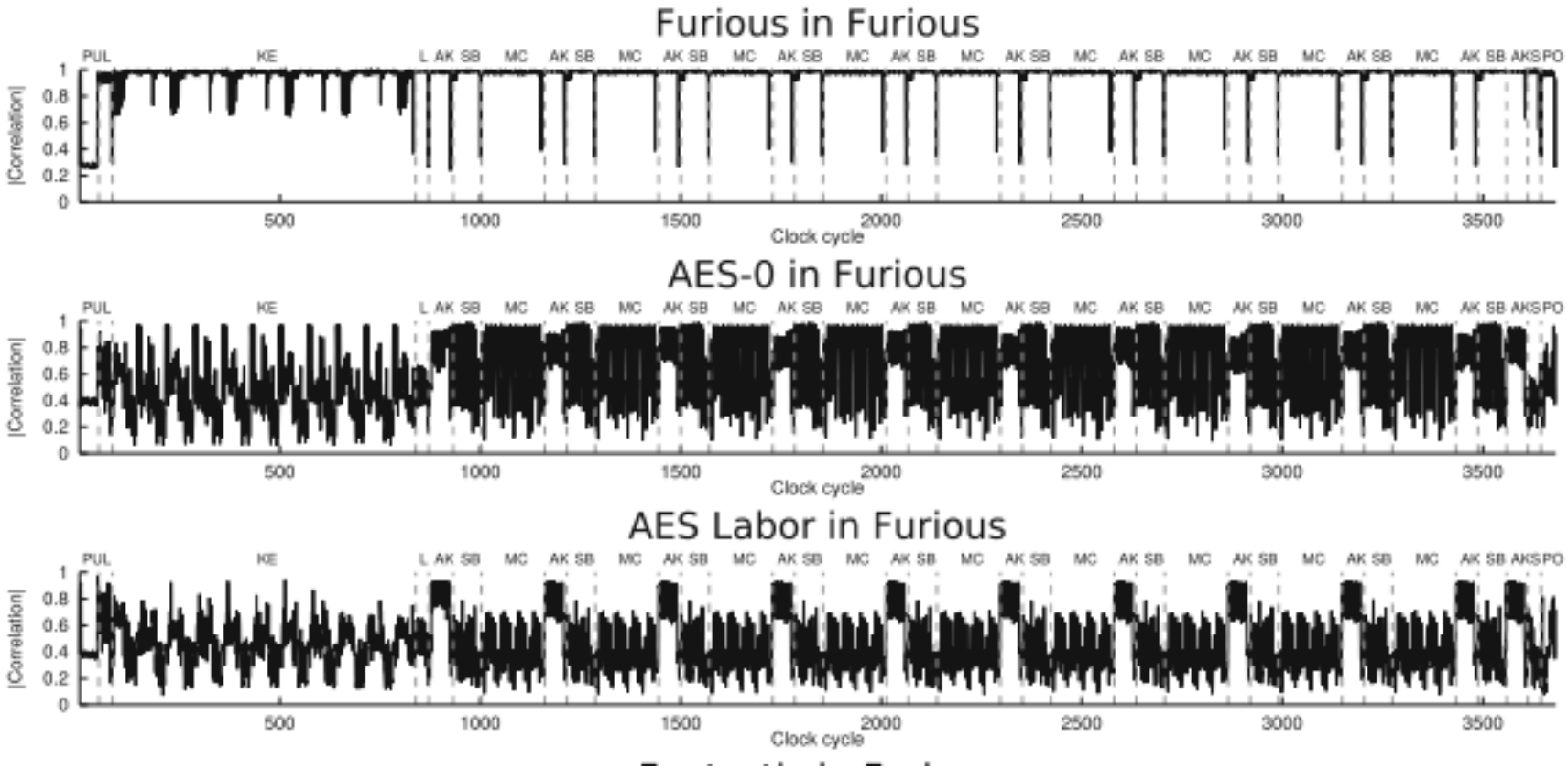
Fig. 10. Similarity matrix of *Furious* with itself.



방 법

- Maximum 값으로 사영





논문 내용 실행

```
import numpy as np
from sklearn import preprocessing
import  matplotlib.pyplot as plt
traces1 = np.load(r'C:₩Users₩khj93₩chipwhisperer₩projects₩tmp₩default_data₩traces₩traces1.npy')
traces2 = np.load(r'C:₩Users₩khj93₩chipwhisperer₩projects₩tmp₩default_data₩traces₩traces2.npy')

program1=traces1[:10000,:200]
program2=traces2[:10000,:200]
#traces2=traces

col1=[[0]*np.shape(program1)[1] for i in range(np.shape(program2)[1])]

for i in range(0,np.shape(program1)[1]):
    #print(i)
    for j in range(0, np.shape(program2)[1]):
        col1[j][i]=np.corrcoef(program1[:,i], program2[:,j])[0][1]

x=[255]*np.shape(program1)[1]

x_scaled = preprocessing.MinMaxScaler().fit_transform(col1)*x
f1 = plt.figure(1)
s = np.max(col1, axis=1)
plt.axis([0,np.shape(s)[0],0,1])
plt.plot(s)
f2 = plt.figure(2)
plt.imshow(x_scaled)
plt.gray()
plt.show()
```

논문 내용 실행

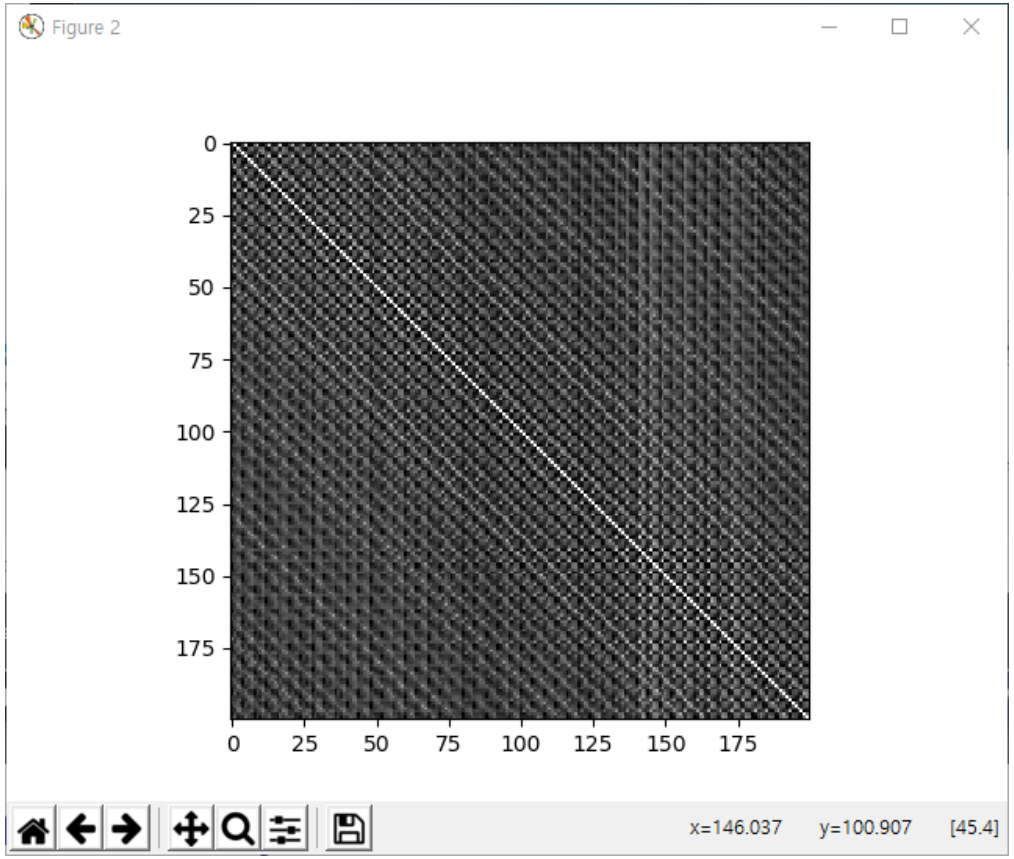


Figure 2
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175
x=146.037
y=100.907
[45.4]

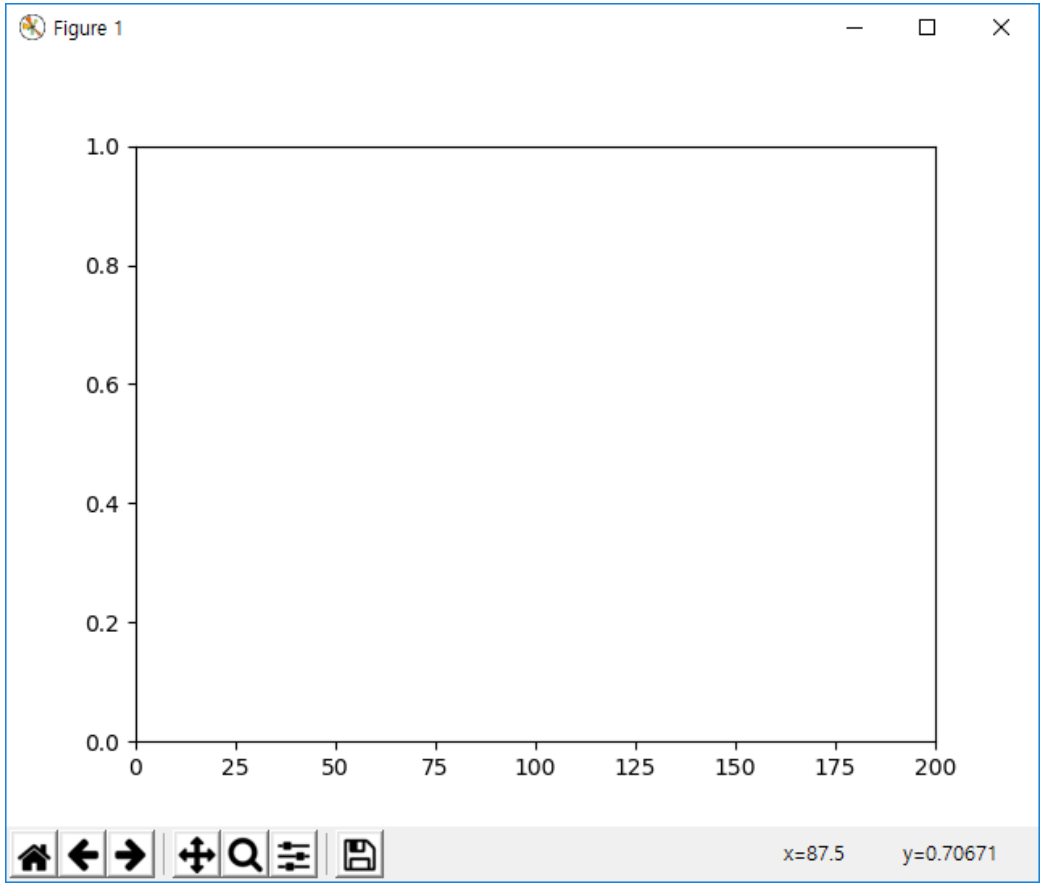


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200
x=87.5
y=0.70671

논문 내용 실행

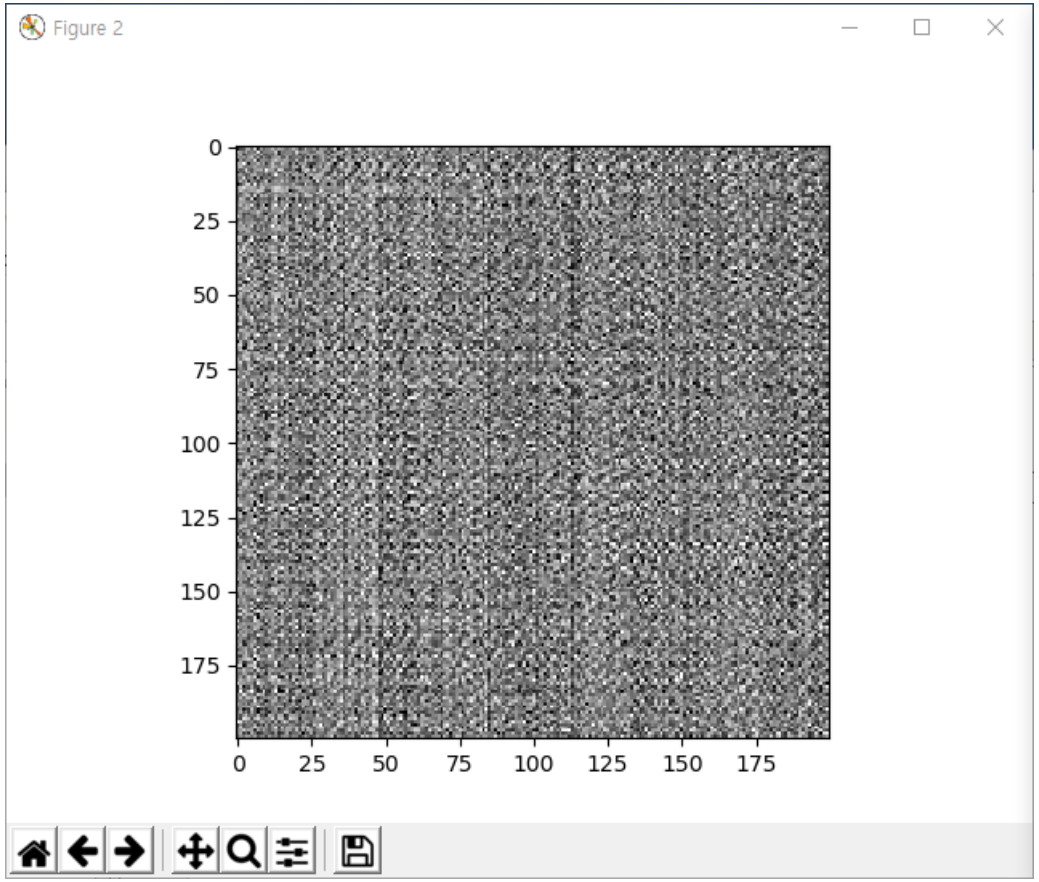


Figure 2
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

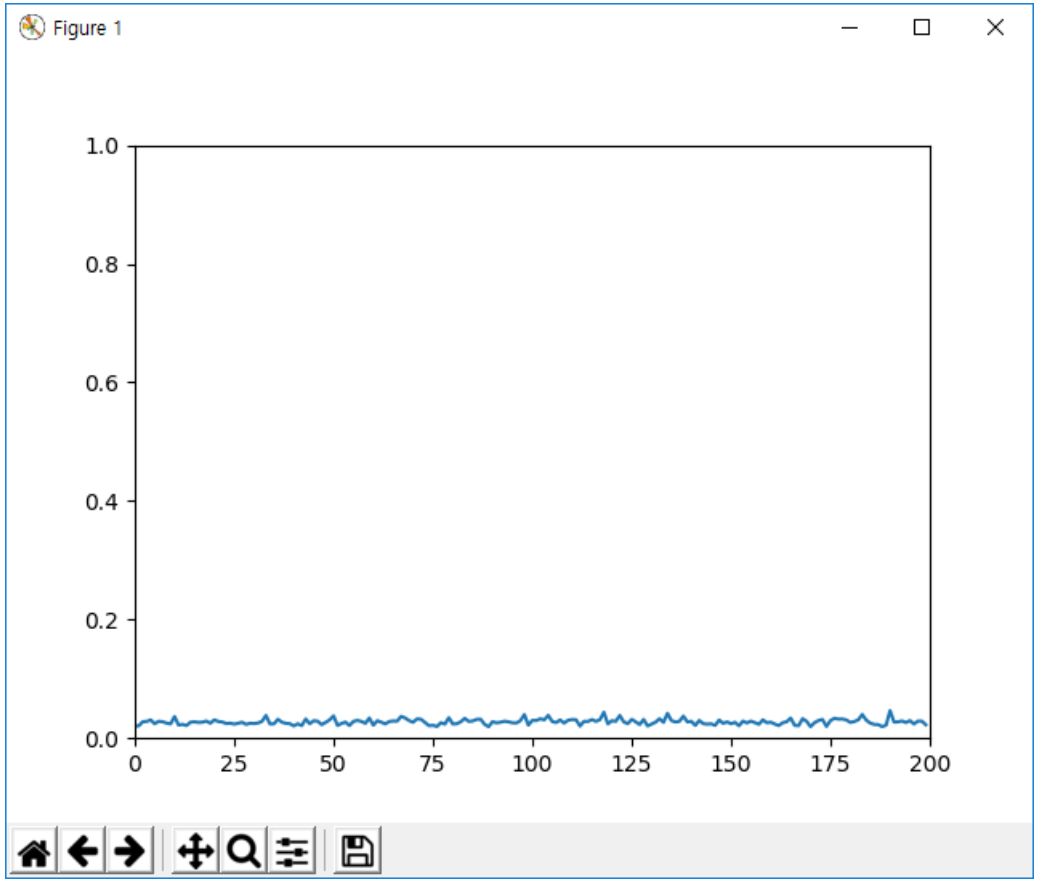


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200

문제점

- 같은 평문의 입력값을 사용해야함
- Chipwhisperer에는 그러한 기능이 없음

-> Chipwhisperer 여러개의 평문을 넣기위한 모듈 추가 필요

추후 응용 논문

- LEA, CHAM, HIGHT와 같은 국산 암호에 적용
- ARX 연산의 특성과 S-BOX연산의 특성 등을 분석하여 이를 이용하면 명확히 특허 침해 확인이 가능
- 새로운 부분에 대해서 아이디어 논의 필요

Q & A