PEPSI
Protect Every Personal Secret Information

KMU

MLSCA SEED

2021년 11월 25일

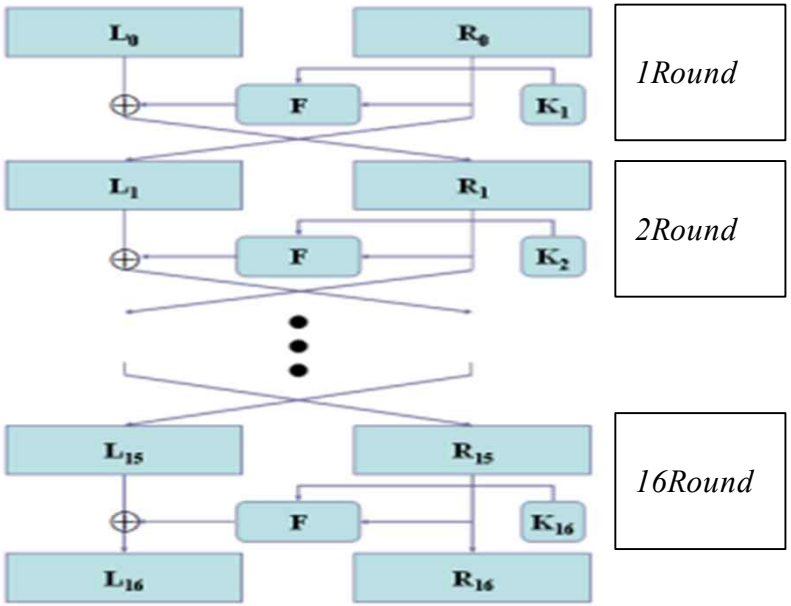
PEPSI

발표자: 최건희

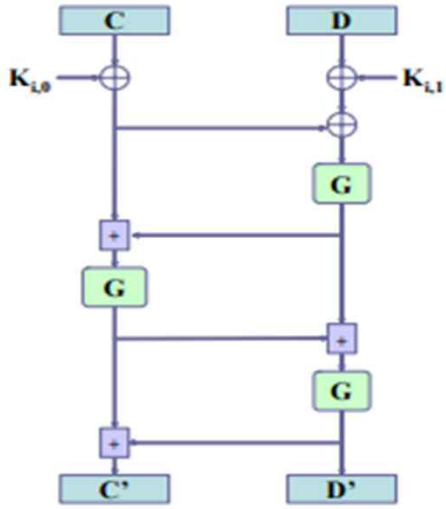
분석한 파형

- AVR-SEED
  - **Profiling**
  - **Attack**

SEED 암호



(그림 1) SEED 전체 구조도



(그림 2) F-함수 구조도

- ✓ **Input: 평문: 128 bit, KEY: 128 bit**
- ✓ **Output: 128bit**
- ✓ **총 round 수 : 16 round**

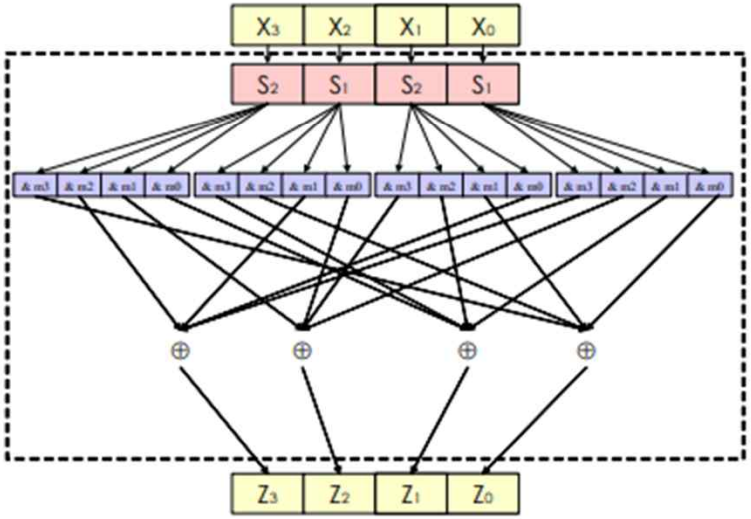
- ✓ **Input: C: 32 bit, D: 32 bit , round_keyL: 32bit, round_keyR: 32 bit**
- ✓ **Output: C': 32bit, D': 32bit**

SEED 암호

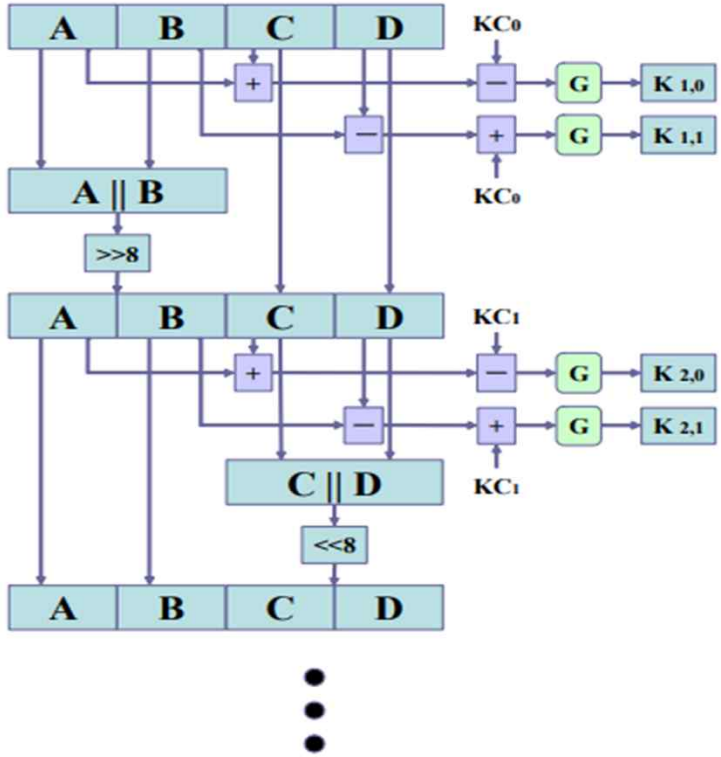
$$Y_3 = S_2(X_3),\quad Y_2 = S_1(X_2),\quad Y_1 = S_2(X_1),\quad Y_0 = S_1(X_0),$$

$$\begin{aligned}
Z_3 &= (Y_0 \& m_3) \oplus (Y_1 \& m_0) \oplus (Y_2 \& m_1) \oplus (Y_3 \& m_2)\\
Z_2 &= (Y_0 \& m_2) \oplus (Y_1 \& m_3) \oplus (Y_2 \& m_0) \oplus (Y_3 \& m_1)\\
Z_1 &= (Y_0 \& m_1) \oplus (Y_1 \& m_2) \oplus (Y_2 \& m_3) \oplus (Y_3 \& m_0)\\
Z_0 &= (Y_0 \& m_0) \oplus (Y_1 \& m_1) \oplus (Y_2 \& m_2) \oplus (Y_3 \& m_3)
\end{aligned}$$

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



(그림 3) G 함수

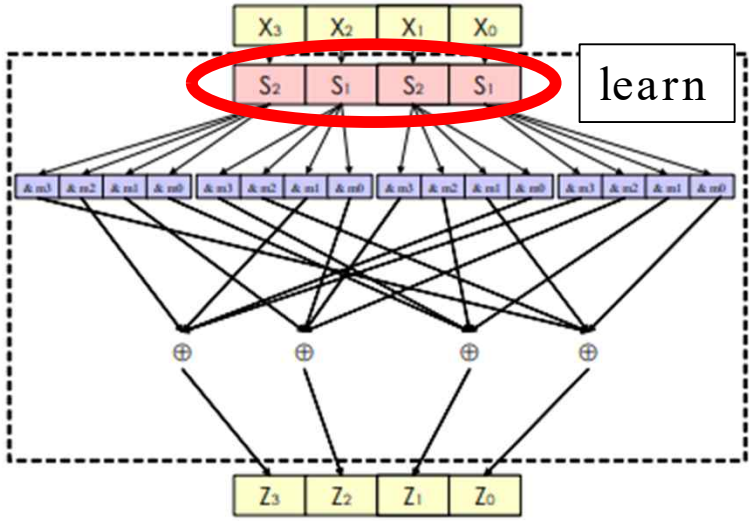


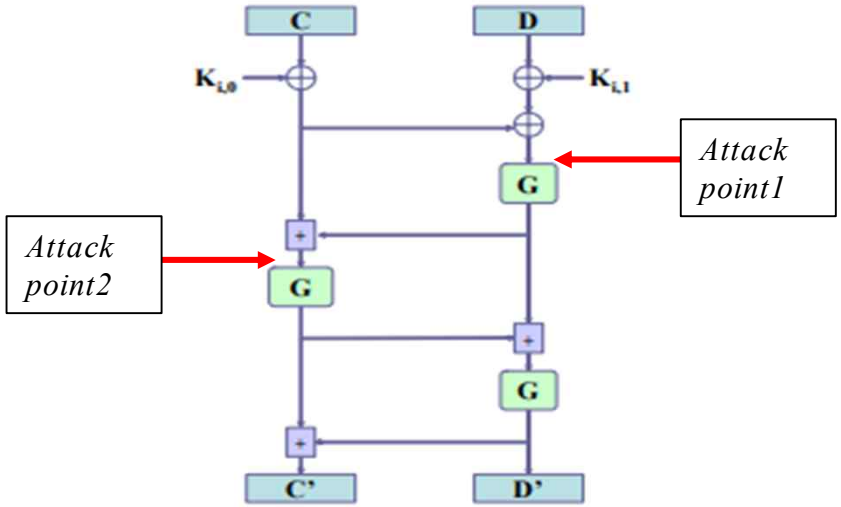
(그림 4) 라운드키 생성과정 구조도

- ✓ **Input: list X: 8 bit || 8 bit || 8 bit || 8bit**
- ✓ **Output: list Z: 8 bit || 8 bit || 8 bit || 8bit**

- ✓ **Input: list ABCD: 32bit || 32bit || 32bit || 32bit**
- ✓ **Output: round_keyL, round_keyR: 32bit, 32bit**

공격 설계





- **Profiling**

✓ 1번째 G함수의 S_box_output 를 중간 값으로 설정하여 모델을 설계

- **Attacker**

✓ Attacker는 학습된 모델과 S_box_inv를 통해
✓ 1번째 G함수의 Input(=C ⊕ *D* ⊕ round _keyL ⊕round_keyR)을 획득.

✓ Attacker는 G2 관련 학습된 모델과 S_box_inv를 통해
✓ 2번째 G함수의 Input(=C ⊕ round _keyR + G1_ output)을 획득.

✓ Attacker는 C ⊕ round _keyL 가 계산 가능하다 또한 G1_input을 통해
✓ D⊕ round _keyR 이 계산 가능하다.

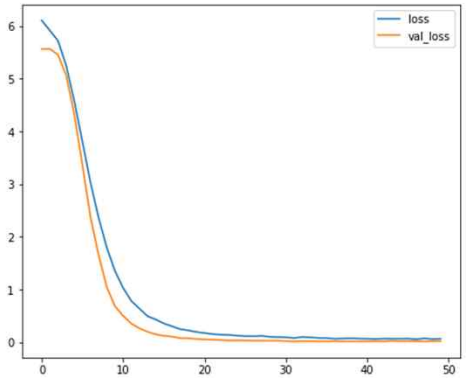
✓ 이로부터 Attacker는 round_key를 얻을 수 있다.

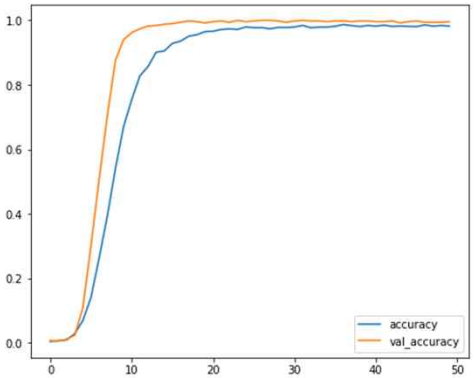
MLP 신경망 구조

```
def build_MLP(Points_len = 6001, Hidden_node = [400, 200], Output_node = 256, Activations = "relu", Dropout = True, BatchNo = False):

    # 입력층
    inp = keras.layers.Input(shape = (Points_len, ), name = "input")

    # 입력층 배치 정규화
    Hidden_layer = keras.layers.BatchNormalization()(inp)

    for node in Hidden_node:
        # 은닉층 추가
        Hidden_layer = keras.layers.Dense(node, activation = Activations)(Hidden_layer)
        if Dropout: # 드롭아웃 수행
            Hidden_layer = keras.layers.Dropout(0.5)(Hidden_layer)
        if BatchNo: # 배치 정규화 수행
            Hidden_layer = keras.layers.BatchNormalization()(Hidden_layer)

    # 활성화 함수를 softmax로 설정 (각 class일 확률을 반환)
    out = keras.layers.Dense(Output_node, activation = 'softmax', name = 'output')(Hidden_layer)

    # 입력층과 출력층을 받아서 객체화
    m = keras.Model(inp, out)

    # 손실함수를 categorical_crossentropy, 성능 지표를 정확도로 설정하여 모델을 구성
    m.compile(optimizer = keras.optimizers.adam_v2.Adam(), loss = 'categorical_crossentropy', metrics = ['accuracy'])

    # 모델 shape과 파라미터 수를 출력
    m.summary(80)

    return m

SCA2_MLP1 = build_MLP(6001, [1024, 512], 256, "relu", True, True)
SCA2_MLP2 = build_MLP(6001, [1024, 512], 256, "relu", True, True)
SCA2_MLP3 = build_MLP(6001, [1024, 512], 256, "relu", True, True)
SCA2_MLP4 = build_MLP(6001, [1024, 512], 256, "relu", True, True)
```

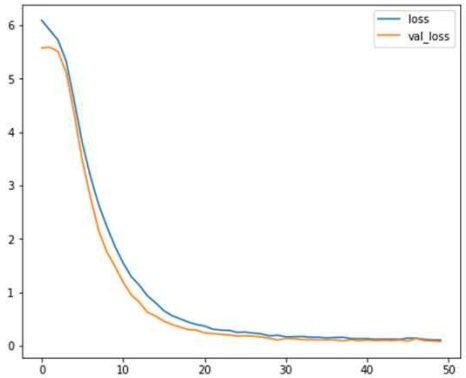
Profiling g-1함수 s-box 출력 값 학습

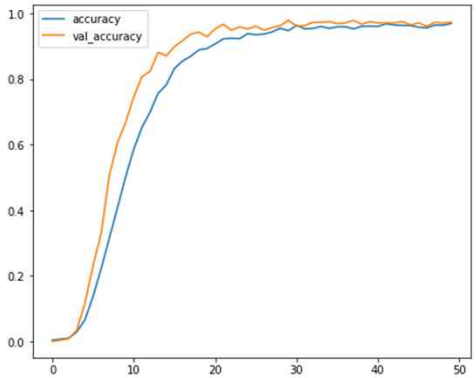
◆ 학습결과



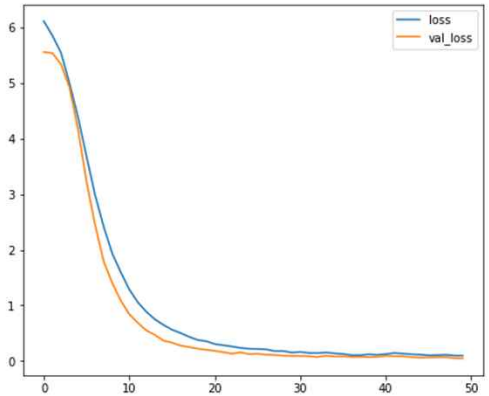


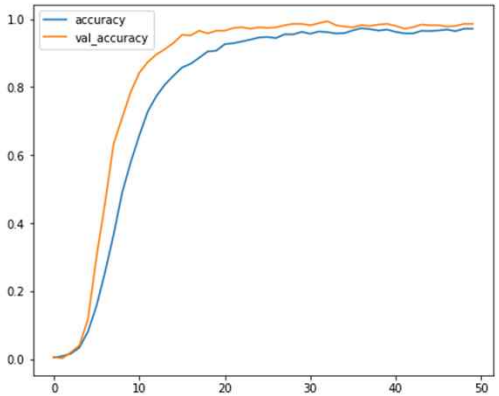
MLP1



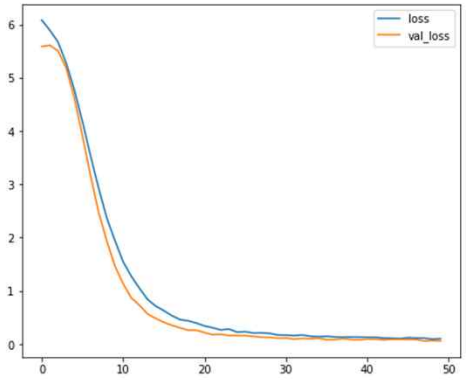


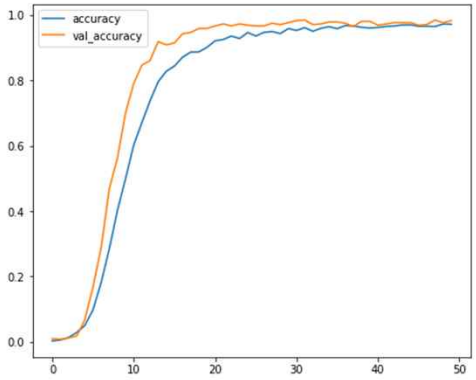
MLP2





MLP3

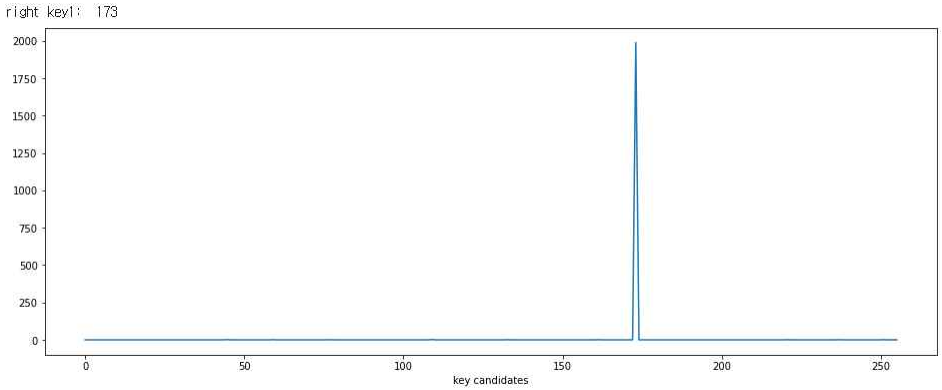


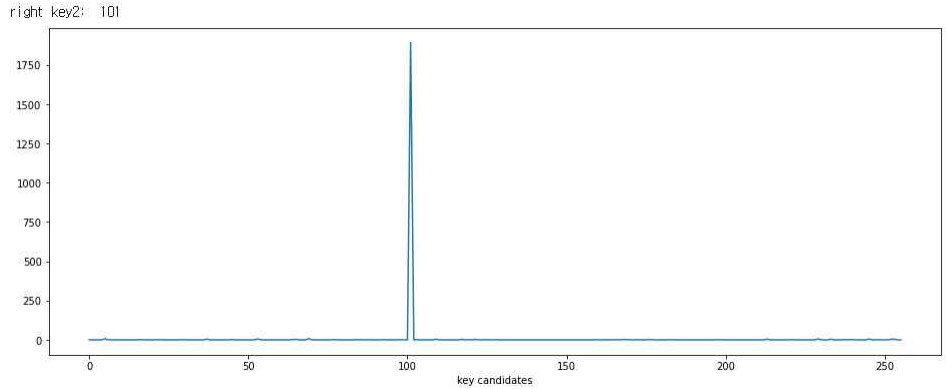


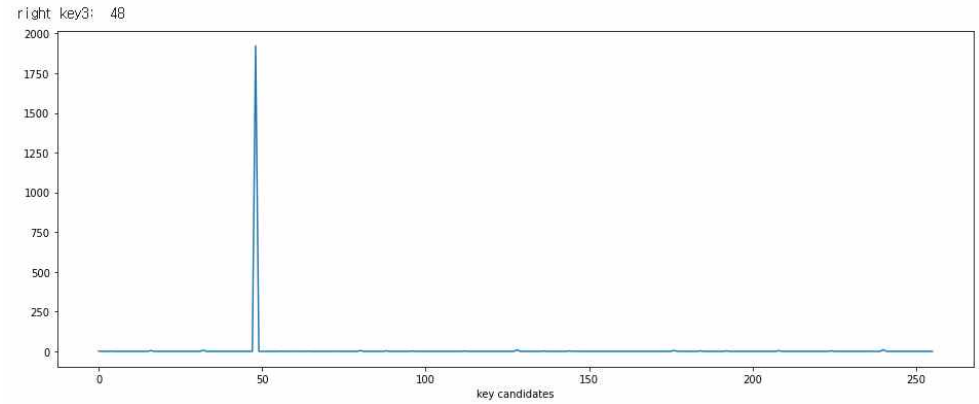
MLP4

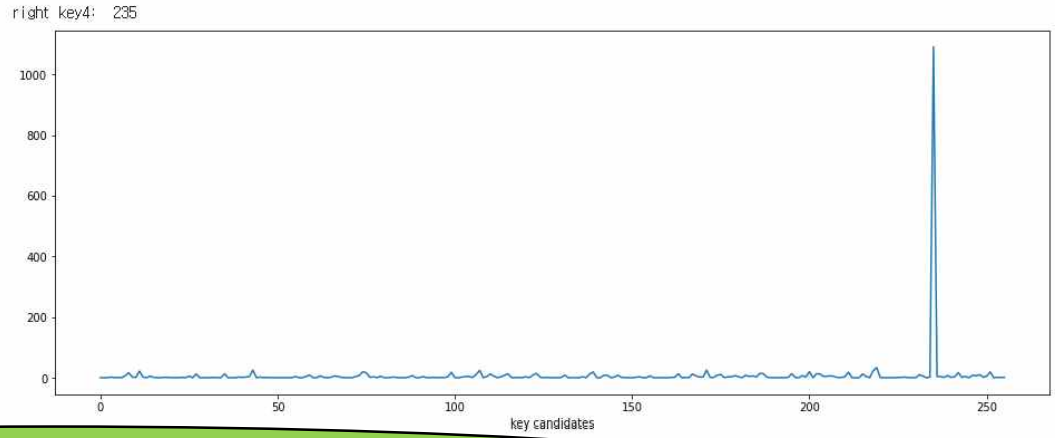
Attack XOR(KL, KR)획득

◆ 공격결과





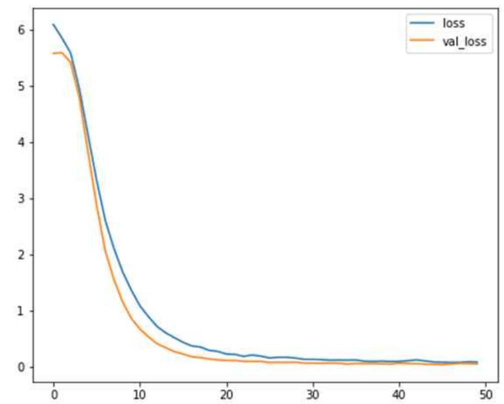


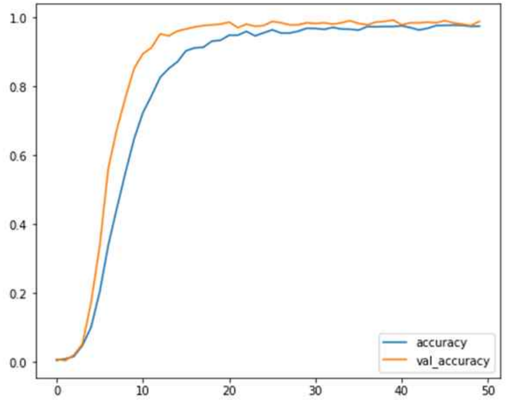


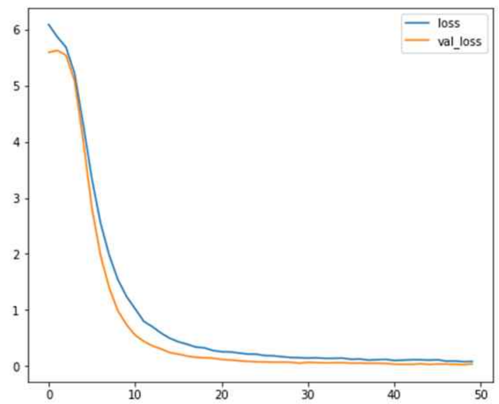
KL 과 KR의 XOR 연산 된 값을 획득

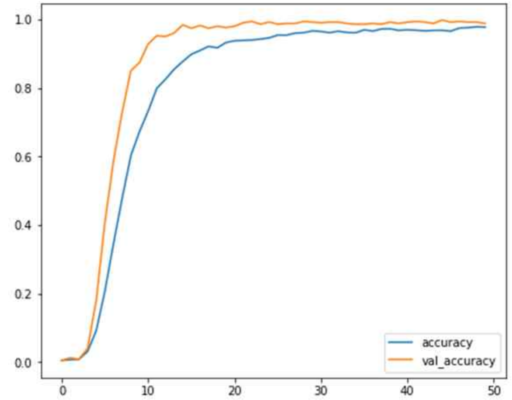
Profiling g2의 s-box에 대한 학습

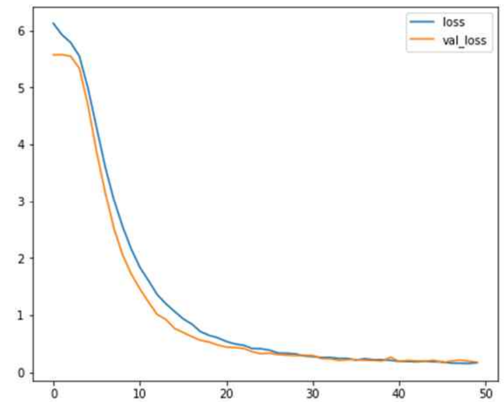
◆ 학습결과

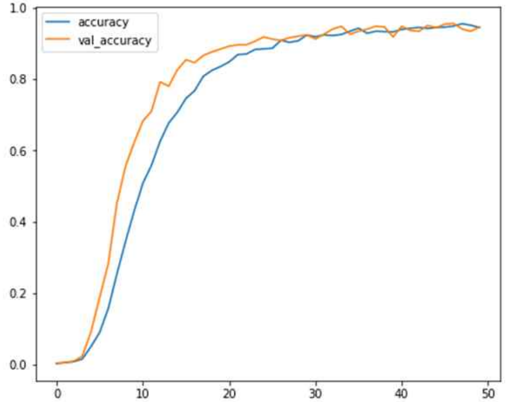


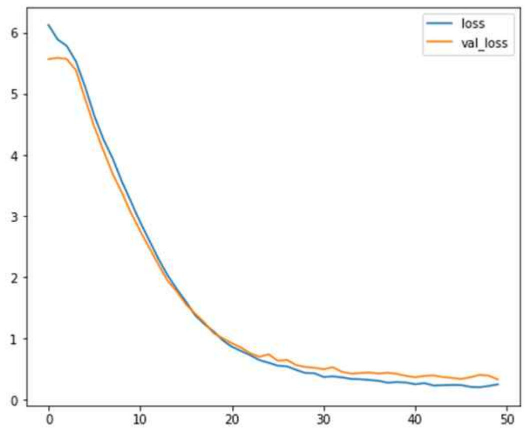


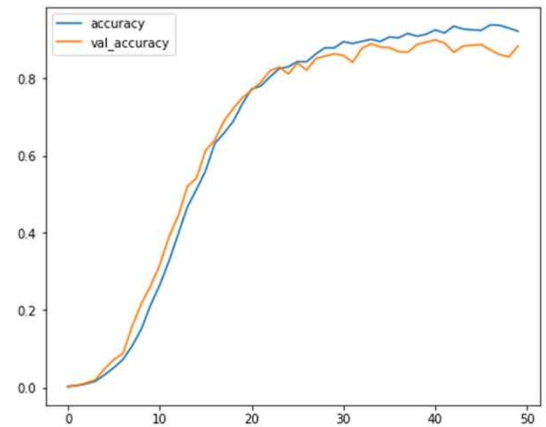








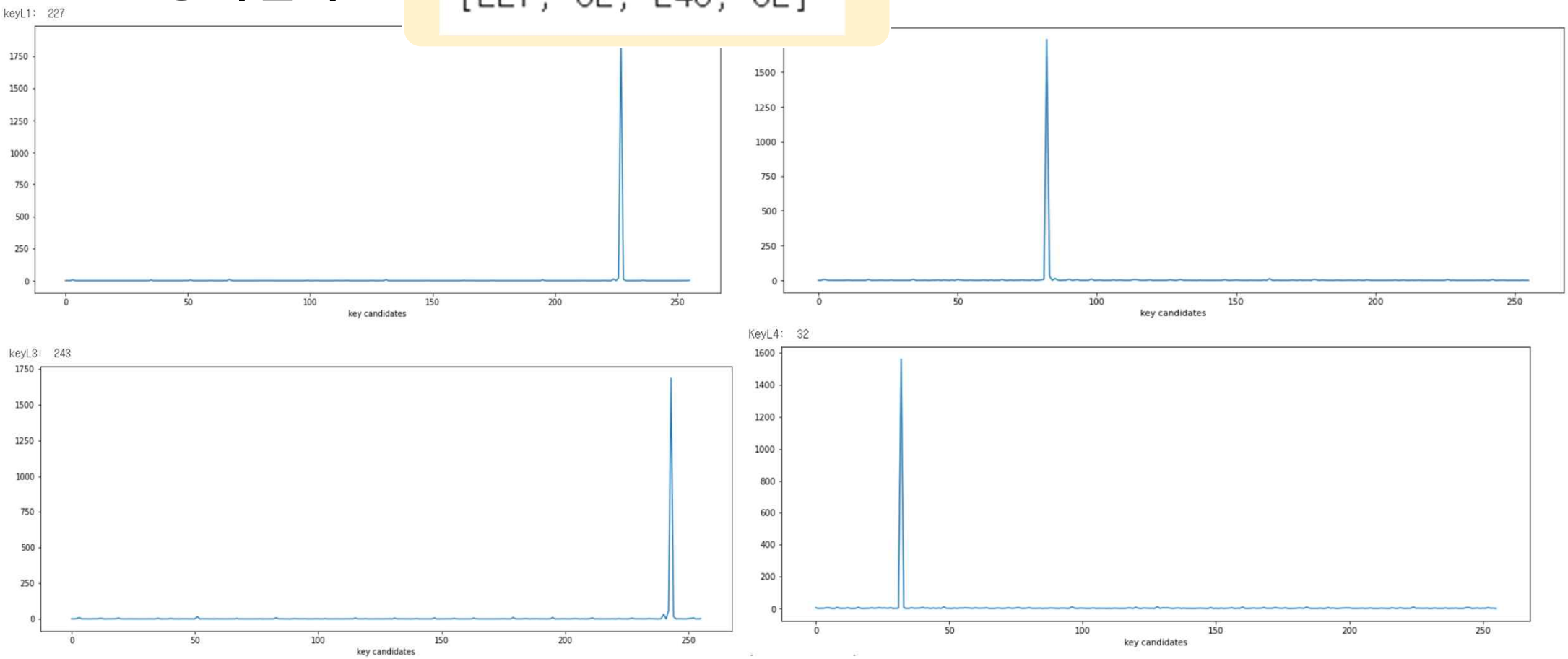




Attack KL획득

◆ 공격결과

[227, 82, 243, 32]



KL 과 C의 XOR 연산 된 값 => KL 값 획득

Attack 최종공격 결과 보고

- XOR(KL, KR)의 값을 첫번째 학습모델을 통해 정확하게 복구 하였다.
- 2번째 G함수의 S-box 출력 값을 통해 XOR(C,KL) 을 정확하게 복구하였다.
- 평문을 통해 KL을 복구하고, 처음 공격한 결과 XOR(KL,KR)을 통해 KR 또한
  복구 가능했다.
- 파형과 중간값 (S-box) 출력 만으로 key를 공격하는데 성공했다.
- 즉, 우리는 암호를 설계할 때 부채널 공격에 대비하여 설계하여야 한다.

Question?