2023 암호분석경진대회

2번 문제

목차

0. 문제풀이 코드 목록

* 2번 문제 관련 파일 목록

1. KUICS_2번/solver.py: 정답 코드
2. KUICS_2번/nr6_speck_distinguisher_structure.png: pytorch 기능으로 export한 모델 구조 이미지
3. KUICS_2번/report.json: 예시 공격 실행 결과를 저장한 파일

* speck.py의 기존 코드를 수정하였다. make_train_data 함수에 테스트를 위해서 고정된 키를 사용할 수 있도록 변경하였으며, 변경사항은 다음과 같다.

```
117         - def make_train_data(n, nr, diff=(0x0040,0)):
            117 + def make_train_data(n, nr, diff=(0x0040,0), fixed_key = None):
118         118     Y = np.frombuffer(urandom(n), dtype=np.uint8); Y = Y & 1;
119         -       keys = np.frombuffer(urandom(8*n),dtype=np.uint16).reshape(4,-1);
            119 +   # fixed_key 사용 가능하도록 패치
            120 +   keys = np.frombuffer(urandom(8*n),dtype=np.uint16).reshape(4,-1) if fixed_key is None else np.frombuffer(fixed_key, dtype=np.uint16).repeat(n).reshape(4, -1);
```

1. 개요

SPECK은 2013년 6월에 NSA가 공개한 ARX 구조를 가지는 경량 블록암호로, 소프트웨어 구현에서의 성능에 최적화되어 있는 암호이다. 문제에서 사용하는 SPECK-32/64의 경우, 32bit의 block size, 64bit의 key size를 가지며 총 22라운드로 구성되어 있다. 라운드 함수는 2개의 word(각 16비트)로 나눠진 입력을 받아 2번의 rotate, 블록 간 addition, 계산된 라운드 키와 xor을 수행한다.

주어진 문제는 제공된 neural distinguisher를 활용하여 7라운드 SPECK-32/64의 7라운드 서브키를 복구하는 것을 목표로 한다. 제공된 딥러닝 모델은 6라운드 SPECK-32/64로 암호화된 암호문을 구별할 수 있는 모델이며 이를 활용하여 차분 공격을 수행할 수 있다.

- 차분 공격(differential cryptanalysis)

차분 공격은 선택 평문 공격(chosen-plaintext attack)을 가정하며 입력값의 변화에 따른 출력값의 변화를 이용하여 암호를 공격하는 방법이다. 블록 암호의 경우에, substitution-permutation을 거치는 network에서 값의 변화가 어떻게 일어나는 지 분석하여 랜덤하지 않은 상태를 찾는다.

공격자는 고정된 차분(difference)를 가지는 임의의 평문 쌍((P_i, P_j) with $P_i \oplus P_j = \alpha$)들을 선택한다. 이 중 하나를 (P_0, P_1)이라고 하자. 암호화 함수 E에 대하여 $E(P_0) = C_0$, $E(P_1) = C_1$라고 할 때, $C_0 \oplus C_1$이 특정 값 β일 확률, $P(C_0 \oplus C_1 = \beta | P_0 \oplus P_1 = \alpha)$을 계산한다. 이 확률은 이상적인 암호시스템에서 $1/2^n$이 되어야 하지만, 그렇지 않을 경우에($N-1$ 라운드 암호화를 진행한 값 D_0, D_1이 위의 성질을 만족할 때), 아래와 같은 과정으로 라운드 키를 복구할 수 있다.

1. $P_0 \oplus P_1 = \alpha$를 만족하는 P_0, P_1을 N 라운드 암호화한 값 (C_0, C_1)을 준비한다.
2. 라운드 키가 될 수 있는 모든 k에 대해 1라운드 복호화를 진행하여 D_0^k, D_1^k를 획득하고, 차분이 β인지 확인한다 ($D_0^k \oplus D_1^k = \beta$). 이를 만족시키는 k가 N 라운드 서브키일 확률이 높다.
3. 위 과정을 동일한 마스터키로 암호화하는 다른 평문 쌍들에 대해 진행하여 가장 확률이 높은 k를 선택한다.

- 딥러닝 기반 구분자를 이용한 차분 분석

차분 공격을 진행하기 위해서는 암호화 함수 E의 가능한 모든 차분쌍 (P_i, P_j)에 대하여 입력/출력 차분 $P_i \oplus P_j = \alpha, C_i \oplus C_j = \beta$와 차분확률 $P(C_i \oplus C_j = \beta | P_i \oplus P_j = \alpha)$를 계산해야 하며, 차분확률이 가장 높은 차분쌍을 알고 있어야 한다는 제약이 있다. 이러한 차분 확률을 이용하여 E의 출력 결과와 랜덤한 출력 결과를 구분하는 것을 구분자라고 하는데, 본 문제에서는 딥러닝을 적용한 구분자를 사용하여 위의 과정을 생략하였다.

2. 풀이

1.

주어진 Neural distinguisher는 ResNet 구조의 CNN을 모델로 한다.

출력 Z는 입력으로 암호문 쌍 (C, C')가 주어졌을 때 해당 암호문 쌍이 $0x0040/0000$의 차분을 가진 평문 쌍으로부터 나온 암호문 쌍으로 분류될 확률이다.

2.

$$D_i^k = Dec1r(C_i)$$
$$k = \underset{0 \le i \le 2^{64}-1}{\operatorname{argmax}} \; Net((D_0^k, D_1^k))$$

입력 데이터 (C_0, C_1)을 7라운드 서브키로 가능한 모든 키 $k \in \{i | i = 0, 1, 2, \cdots, 2^{64}-1\}$으로 1라운드 복호화하여 그 값을 (D_0^k, D_1^k)라고 하자. 2^{64}개의 (D_0^k, D_1^k)가 neural distinguisher의 입력이 되며 총 2^{64}개의 신경망의 예측값 Z_k가 출력된다. 이 출력들 중 최댓값을 갖도록 하는 k가 해당 라운드에서 사용된 서브키일 확률이 가장 높은 키이다.

3.

2.의 수식과 nerual distinguisher를 활용하여 다음과 같이 7라운드 서브키를 복구하기 위한 차분 공격을 구성할 수 있다.

1) $P_0 \oplus P_1 = 0x0040/0000$를 만족하는 P_0, P_1을 선택하여 암호화한다. $C_0 = E(P_0), C_1 = E(P_1)$인 암호문 쌍 (C_0, C_1)을 획득한다.

2) 7라운드 서브키로 가능한 모든 k에 대해 1라운드 복호화를 진행하여 D_0^k, D_1^k를 얻는다.

3) $Net(D_0^k, D_1^k)$의 출력 Z에 따라 k의 score를 저장하고, score가 가장 높은 k를 취한다.

추가로, 참고 논문에서 제시된 key averaging 알고리즘을 적용하여 scoring 시 정확도를 개선하였으며, multiprocessing을 활용하여 더욱 효율적인 공격을 진행하도록 최적화했다.

차분공격의 정확도는 100%가 아니며, 주어진 nerual distinguisher의 정확도 역시 약 75%이기 때문에 공격이 실패할 가능성도 존재한다.

3. 실행 결과

다음은 실행 결과 파일(report.json)의 내용 일부를 캡처한 것이다. 공격은 M1 Pro 기준 약 3분 이내로 완료되었다.

master_key에 대한 각 라운드별 서브키가 subkeys 배열에 저장되어 있으며, 이 예시에서는 22297이 7라운드 서브키임을 확인할 수 있다. attack_size는 생성할 암호문 쌍의 개수로, 32개의 쌍만 있어도 성공적으로 7라운드 서브키를 복구할 수 있다. score 배열은 라운드 키로 가능한 값과 score를 저장하고, score에 따라 결과를 정렬한 배열이다. 예시에서 가장 높은 score를 가지는 22297이 실제 7라운드 서브키와 일치하여 공격이 성공했음을 알 수 있다.


```
{
  "master_key": "94e5369d7d16e434",
  "subkeys": [
    13540,
    64640,
    39864,
    43875,
    26188,
    17450,
    22297
  ],
  "attack_size": 32,
  "score": [
    [
      22297,
      71.05068693200029
    ],
    [
      38681,
      70.17296121706673
    ],
    [
      55065,
      69.76322452937144
    ],
    [
      5913,
      69.68802439479082
    ],
    [
      22425,
      38.7992230554056
    ],
```


4. 참고 자료

https://www.youtube.com/watch?v=ONhPfIABxFs

https://www.kci.go.kr/kciportal/ci/sereArticleSearch/ciSereArtiView.kci?sereArticleSearchBean.artiId=ART002832772

https://journal-home.s3.ap-northeast-2.amazonaws.com/site/2020kics/presentation/0475.pdf