2024 암호분석경진대회 - 비밀분쇄기팀 5번 문제 풀이

Table 1: 실험 결과.

Round	Model	Accuracy	평문-암호문 쌍 개수에 따른 키 순위				
			32	64	128	256	512
4	Baseline	0.6950	3.5	0.4	0.4	0.4	0.4
	Hou et al.	0.5778	8.7	1.8	0.6	0.6	0.6
	Ours	0.6617	6.0	0.4	0.4	0.4	0.4
5	Baseline	0.5609	19.1	12.9	6.4	3.9	2.9
	Hou et al.	0.5100	11.3	18.4	20.7	19.2	15.9
	Ours	0.5069	16.0	19.1	15.8	14.4	13.9

Notation. 문제 풀이에 다음과 같은 notation을 사용한다.

- P : 64-bit 평문
- C : 64-bit 암호문
- P_H : 평문의 상위 32-bit
- P_L : 평문의 하위 32-bit
- C_H : 암호문의 상위 32-bit
- C_L : 암호문의 하위 32-bit
- X_i : i번째 라운드에서 F_i의 입력으로 사용되는 32-bit 중간 계산 결과
- K_i : i번째 라운드에 사용되는 48-bit subkey
- $F_i(X_i, K_i)$: i번째 라운드의 F-function
- $A[i]$: A의 i번째 bit
- $A[i, j, ..., k]$: $A[i] \oplus A[j] \oplus ... \oplus A[k]$
- $A < i, j, ..., k >$: $A[i]||A[j]||...||A[k]$. $||$는 bit concatenation이다.

질문 1. Feistel 구조 블록 암호 알고리즘에서 선형 공격을 통한 1-bit 키 복구를 위한 알고리즘을 기술하시오.

답변 1.

어떤 Feistel 구조 블록 암호 알고리즘에 대해서 다음과 같은 선형 등식을 생각해볼 수 있다.

$$P[i_1, i_2, ..., i_a] \oplus C[j_1, j_2, ..., j_b] = K[k_1, k_2, ..., k_c] \tag{1}$$

여기서 $i_1, ..., i_a, j_1, ..., j_b, k_1, ..., k_c$는 각각 평문, 암호문, 키에서의 bit index이다. 즉, 평문에서 a개 bit, 암호문에서 b개 bit를 뽑아 xor 하였을때 키에서 c개 bit을 뽑아 xor 한 것과 같음을 나타내는 등식이다. 선형 근사 공격의 핵심 아이디어는 위 식이 랜덤한 P와 C에 대해서는 1/2 확률로 성립하지만, P만 랜덤하고 C가 P에 대한 암호문이라면 특정 bit index 집합에 대해 위 식이 성립할 확률이 $p \neq 0.5$라는 것이다. 예를 들어, 3-round DES에서 아래 식이 성립할 확률은 69.5%임을 유도할 수 있다 [3].

$$L_3 : P_H[7, 18, 24, 29] \oplus P_L[15] \oplus C_H[7, 18, 24, 29] \oplus C_L[15] = K_1[22] \oplus K_3[22] \tag{2}$$

식의 형태가 조금 다르지만 P_H, P_L, C_H, C_L, K_1, K_3는 결국 P, C, K의 비트 중 일부이므로 같은 꼴로 변형이 가능하다. **만약 많은 수의 평문-암호문 쌍을 가지고 있다면 다음과 같은 알고리즘으로 1-bit information을 얻어낼 수 있다.**

1. 평문-암호문 쌍의 개수를 N이라고 하자.

2. 평문-암호문 쌍을 식 1의 좌변에 대입하여 결과가 0인 쌍의 개수를 T라고 하자.

   (a) 만약 $T > N/2$ and $p > 1/2$ 또는 $T <= N/2$ and $p < 1/2$이면 $K[k_1, k_2, ..., k_c] = 0$으로 추측한다.

   (b) 만약 $T > N/2$ and $p < 1/2$ 또는 $T <= N/2$ and $p > 1/2$이면 $K[k_1, k_2, ..., k_c] = 1$으로 추측한다.

위 알고리즘의 추측이 맞을 확률은 $|p - 1/2|$가 클수록 높아지기 때문에 그러한 식(i.e., $i_1, ..., i_a, j_1, ..., j_b, k_1, ..., k_c$)를 찾는 것이 중요하며, Matsui et al [3]는 3 $\sim$ 20라운드 DES에 대해 $|p - 1/2|$가 최대인 식을 밝혔다. 또한 평문-암호문 쌍을 많이 가지고 있어도 추측이 맞을 확률이 높아진다.

질문 2. n-라운드로 축소된 DES에서 평문, 암호문, 키의 bit location mask를 선택하고, 선택한 bit location mask를 이용하여 구성한 선형 근사식 L을 기술하시오.

답변 2.

실질적으로 n-라운드 DES를 공격하기 위해서는 위의 1-bit 알고리즘보다는 multi-bit 정보를 얻을 수 있도록 아래와 같이 변형된 알고리즘을 사용한다. $(n-1)$-라운드 선형근사식에 첫번째 라운드의 영향을 적용하여 아래와 같은 식을 생각할 수 있다.

$$P[i_1, i_2, ..., i_a] \oplus C[j_1, j_2, ..., j_b] \oplus F_1(P_L, K_1)[l_1, l_2, ..., l_d] = K[k_1, k_2, ..., k_c] \tag{3}$$

위 식을 가지고 아래와 같은 알고리즘을 적용할 수 있다.

1. 모든 가능한 K_1 값을 순회하면서 평문-암호문 쌍을 식 3의 좌변에 대입하여 결과가 0인 쌍의 개수를 센다. 그 중 가장 큰 값을 T_{max}, 가장 작은 값을 T_{min}이라고 하자.

2. 평문-암호문 쌍을 식 1의 좌변에 대입하여 결과가 0인 쌍의 개수를 T라고 하자.

   (a) 만약 $|T_{max} - N/2| > |T_{min} - N/2|$를 만족하면 T_{max}에 대응되는 K_1 값을 취하고, 우변은 $p > 1/2$라면 0, $p < 1/2$라면 1로 추측한다.

   (b) 만약 $|T_{max} - N/2| < |T_{min} - N/2|$를 만족하면 T_{min}에 대응되는 K_1 값을 취하고, 우변은 $p > 1/2$라면 1, $p < 1/2$라면 0로 추측한다.

여기서 K_1은 48-bit이지만 $F_1(P_L, K_1)[l_1, l_2, ..., l_d]$의 값에 실제로 영향을 미치는 비트는 더 적으므로 전수조사가 가능하다. 영향을 미치는 비트의 수를 e라고 하면, 우변에서 얻는 1-bit information을 합하여 총 $e + 1$ 비트의 정보를 얻을 수 있다. 또한 유사하게 $(n-1)$-라운드 선형근사식에 마지막 라운드의 영향을 적용한 식을 얻어내고 같은 알고리즘을 적용할 수 있다.

다음은 Matsui et al.로부터 유도한 4-round 및 5-round DES를 공격하기 위한 선형근사식 L_4와 L_5이다. 아래의 bit location 및 선형 근사식을 사용하여 이후 풀이를 진행하고자 한다.

$$\begin{aligned}
L_4 :& P_L[7, 18, 24, 29] \oplus P_H[15] \oplus C_L[15] \oplus C_H[7, 18, 24, 29] \oplus F_1(P_L, K_1)[15] \\
=& K_2[22] \oplus K_4[22] \\
L_5 :& P_L[7, 18, 24, 29] \oplus P_H[15] \oplus C_L[7, 18, 24, 29, 27, 28, 30, 31] \oplus C_H[15] \oplus F_1(P_L, K_1)[15] \\
=& K_2[22] \oplus K_4[22] \oplus K_5[42] \oplus K_5[43] \oplus K_5[45] \oplus K_5[46]
\end{aligned}$$

질문 3. 딥러닝을 활용하여 문제 (2)에서 기술한 선형근사식을 이용한 키 복구 공격 수행 알고리즘을 기술하시오.

답변 3.

답변 2의 알고리즘을 생각했을 때, 주어진 평문-암호문 쌍에 대해 올바른 K와 틀린 K를 구분할 수 있는 classification model을 학습하여 키 복구 공격을 수행할 수 있다. 구체적으로 선형근사식 L에서 좌변의 각 비트를 입력으로 하는 딥러닝 모델을 구성하면, xor 연산보다 더 나은 representation을 찾을 수 있으리라고 기대할 수 있다.

Hou et al. [2]은 확률이 p인 베르누이 분포와 확률이 0.5인 베르누이 분포를 구분하는 네트워크를 학습하는 방식으로 접근하였다. 그러나 실험 결과 선형근사식의 좌변을 입력으로 하고 우변을 라벨로하는 네트워크를 학습하는 것이 더 효과적임을 알 수 있었다.

다음은 Hou et al.의 학습 데이터 생성 방식을 정리한 것이다.

1. 선형근사식 L의 우변을 0으로 하는 K를 Uniform distribution으로 생성한다.

2. Uniform distribution으로 랜덤한 P를 생성하고 N-round DES를 적용하여 암호문 C를 생성한다.

3. Uniform distribution으로부터 0 또는 1인 라벨 Y를 생성한다.

4. Y 값에 따라서 입력 X를 생성한다.

   - **Positive sample 생성** : Y가 1인 경우, 식 L의 좌변을 계산하되 xor을 적용하지 않고 각 비트를 한개의 원소로 하여 입력 벡터 X를 생성한다. 예를 들어, L_4에서 X는 길이 11의 벡터

   $$\{P_L[7], P_L[18], P_L[24], P_L[29], P_H[15], C_H[7], C_H[18], C_H[24], C_H[29], C_L[15], F_1(P_L, K_1)[15]\} \quad (4)$$

   가 된다.

   - **Negative sample 생성** : Y가 0인 경우, 위와 유사하게 X를 생성하되 K에 영향을 받는 값에 대해서는 uniform distribution으로부터 랜덤한 값을 생성하여 채운다. 예를 들어, L_4에서 X의 마지막 원소를 $F_1(P_L, K_1)[15]$ 대신 랜덤한 값으로 생성한다.

다음은 우리가 사용한 학습 데이터 생성 방식을 정리한 것이다.

1. Uniform distribution으로 랜덤한 P 및 K를 생성하고 N-round DES를 적용하여 암호문 C를 생성한다.

2. 식 L의 좌변을 계산하되 xor을 적용하지 않고 각 비트를 한개의 원소로 하여 입력 벡터 X를 생성한다.

3. 식 L의 우변을 계산하여 0 또는 1인 라벨 Y를 생성한다.

위 학습 데이터로 딥러닝 모델 M의 학습을 마쳤다고 가정하면 아래와 같이 키 복구 공격을 수행할 수 있다.

1. L의 좌변에 영향을 미치는 K의 bit location 집합 $mask_K$를 찾는다. 그리고 $K' = K < mask_K >$로 정의하자.

   - 예를 들어, L_4의 좌변에서 K가 영향을 미치는 항은 $F_1(P_L, K_1)[15]$이다. DES의 알고리즘을 역으로 추적하면 좌변에 영향을 미치는 K의 bit location은 $mask_K = \{4, 13, 15, 30, 47, 54\}$임을 알 수 있다.

2. 가능한 K' 값의 후보집합 $K'_{cand} = \{i | 0 \leq i < 2^{|mask_K|}\}$를 생성한다.

   - 예를 들어, L_4에 대한 K'_{cand}는 $\{000000_2, 000001_2, ..., 111111_2\}$이다.

3. 각 $K' \in K'_{cand}$에 대해 평문-암호문 쌍을 딥러닝 모델에 넣어 $M(P, C, K')$가 1로 분류되는 개수 T를 센다.

4. 각 K'의 rank는 T를 내림차순 한 것과 오름차순 한 것 중 작은 rank를 취한다. 해당 랭크를 기준으로 리스트를 반환한다.

반환된 리스트에서 앞에 있을수록 키가 맞을 확률이 높으므로, 앞에 있는 키부터 $mask_K$가 아닌 나머지 비트에 대해 bruteforce 공격을 수행하면 된다. 반대로 올바른 키가 반환된 리스트에서 앞에 있을 수록 좋은 모델이라고 말할 수 있다.

질문 4. 키 복구 공격을 위해 딥러닝 모델의 하이퍼-파라미터와 모델 구조를 상세히 기술하시오.

답변 4.

모델은 ResNet [1]를 참고하여 구성하였다. ResNet과의 차이점은 입력이 spatial 데이터가 아니므로 Convolution 레이어 대신 Linear 레이어를 사용하였고, mini-batch의 크기에 민감한 BatchNorm 레이어 대신 LayerNorm 레이어를 사용하였다. 또한 레이어 수를 적절하게 축소하였다. 사용한 모델은 ResNet과 마찬가지로 basic block의 반복 구조로 이루어져있는데 각 basic block은 2개의 Linear 레이어, 2개의 LayerNorm 레이어, 1개의 ReLU, 그리고 residual connection으로 구성하였다. 모델 코드는 제출한 `model.py`에 있으며 표 2 및 표 3에 모델 구조를 정리하였다. 학습한 모델은 약 1.4MB이며 각각 `model_hou_round_4.pt`, `model_ours_round_4.pt`, `model_hou_round_5.pt`, `model_ours_round_5.pt`로 제출물에 포함되어있다.

Table 2: `BasicBlock(C,K)`의 구조. B는 mini-batch 크기이다.

Layer	Input size
Linear	[B, C]
LayerNorm	[B, K]
ReLU	[B, K]
Linear	[B, K]
LayerNorm	[B, K]
Addition	[B, K]
ReLU	[B, K]

학습에는 cross entropy loss를 사용하였으며 Adam optimizer와 mini-batch size 1k, learning rate 10^{-4}를 사용하였다.

Table 3: 전체 모델의 구조. W는 입력의 길이이다. (4-round의 경우 11, 5-round의 경우 15)

Layer	Input size
BasicBlock(W, 64)	[B, W]
BasicBlock(64, 64)	[B, 64]
BasicBlock(64, 64)	[B, 64]
BasicBlock(64, 128)	[B, 64]
BasicBlock(128, 128)	[B, 128]
BasicBlock(128, 256)	[B, 128]
BasicBlock(256, 256)	[B, 256]
Linear	[B, 128]
-	[B, 2]

질문 5. 해당 알고리즘을 이용해 학습할 때 사용된 평문-암호문 쌍의 개수와 키 복구 정확도를 작성하시오.

답변 5.

학습에는 100k개의 평문-암호문 쌍을 사용하였으며, 과적합을 방지하기 위해 매 epoch마다 별도의 10k개 평문-암호문 쌍으로 이루어진 test 데이터셋으로 정확도를 측정하여 200 epoch 중 가장 좋은 모델을 저장하였다. 실험은 4-round, 5-round DES로 진행하였으며 딥러닝을 사용하지 않고 선형근사식의 좌변을 직접 연산한 Baseline, Hou et al.을 재현한 모델, 그리고 우리 모델을 비교하였다.

실험 결과는 풀이 최상단의 표 1와 같다. 키 순위는 공격을 10번 수행하여 평균을 측정하였다. 4-round DES 공격에서 더 작은 모델로 Hou et al.의 정확도(0.5677)과 유사한 0.5778의 정확도를 재현할 수 있었으며, 데이터셋 생성 방법을 새로 제안한 방법으로 했을 시 더 높은 0.6617의 정확도를 달성할 수 있었다. 그러나 딥러닝을 사용하지 않은 선형근사식 기반 공격의 성능만큼은 달성할 수 없었다. 마찬가지로 5-round DES에서 baseline은 공격 가능한 정도의 정확도를 보여주었으나 딥러닝 모델은 선형근사식이 성립할 확률이 0.5에 가까워져서 분포를 구분하는 능력을 학습하지 못하였다.

부록. 제출물에 대한 설명.

`train.py`는 PyTorch를 이용한 모델 학습 코드이며 모델 코드는 `model.py`에 있다. DES 및 핵심 연산은 C 모듈로 작성하였으며 `helper.cpp`, `DES.h`, `DES.cpp` 파일로 구성되어있다. 아래는 학습 커맨드 예제이다.

```
# Hou et al. style 4-round DES attack model train
python train.py --round 4 --model hou

# Our style 4-round DES attack model train
python train.py --round 4 --model ours

# Hou et al. style 5-round DES attack model train
python train.py --round 5 --model hou

# Our style 5-round DES attack model train
python train.py --round 5 --model ours
```

공격 실험 예시는 다음과 같다. 전체 실험 재현을 위해서는 `exp.sh`를 실행하면 된다.

```
# Compute average rank across 10 iteration with Baseline using 32 pairs
python attack.py --round 4 --model xor --npair 32 --niter 10

# Compute average rank across 10 iteration with Hou et al. style model using 64 pairs
python attack.py --round 4 --model hou --npair 64 --niter 10

# Compute average rank across 10 iteration with our model using 128 pairs
python attack.py --round 4 --model ours --npair 128 --niter 10
```

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Botao Hou, Yongqiang Li, Haoyue Zhao, and Bin Wu. Linear attack on round-reduced des using deep learning. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II 25*, pages 131–145. Springer, 2020.

[3] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer, 1993.