|  |
| --- |
| **2021 국민대학교 정보보안암호수학과 동아리 연합 암호경진대회** |
| 4번 문제 답안 |
| 정보보안암호수학과 20192243 이용진 |
| 4-1.    위의 소비전력 파형이 AES 1-Round에서 수집된 파형이라고 가정해보겠다.  AddRoundKey 에서는 16-byte Key를 평문과 xor 연산해주기 때문에 전력 파형 또한 비슷한 파형이 16번 반복적으로 나타나야 한다.  SubBytes 또한 16-byte를 S-box Lookup Table로 바이트 치환을 해주기 때문에 파형 또한 16번 반복적인 형태가 나타나야 한다.  ShiftRows는 바이트 Sift연산 임으로 반복적인 형태가 나타나지 않는다.  MixColumns는 16-byte 값이 전부 변하기 때문에, 이 또한 16번의 반복적인 형태가 나타나야 한다. MixColumns 은 AddRoundKey와 SubBytes 보다 연산량이 많고 더 오래 걸리기 때문에 파형의 포인트가 더 길어야 한다.  4-2.    이미지 출처: \_\_lecturenote\_2020\_2\_secure\_protocols\_final.pdf  AES-128 Key Schedule는 위의 Pseudo로 진행된다.  3라운드 Key를 Pseudo code line 5~ 13을 거꾸로 진행하여 마스터 키를 복구한다.  Master Key 복구 방법 논리는 다음과 같다.    먼저 3라운드 키를 입력 받아 4개의 word로 만든다.  그런 다음 Key Schedule의 과정을 거꾸로 진행한다.  Master key 복구 Pseudo code에서 중요한 부분은 line 12이다.  Key Schedule Pseudo code의 line12를 보면, rk\_i = rk\_i-4 ⊕ t를 진행한다.  그럼으로 복구 과정에서는 rk\_i-4 = rk\_i ⊕ t를 진행하여 거꾸로 돌아간다.  이런 과정을 반복하다 보면, 3라운드 키로 2라운드 키를 찾을 수 있고  2라운드 키로 1라운드 키를 찾을 수 있으며,  마지막으로 마스터 키를 찾는 것이 가능하다.  3라운드 키: 0x32, 0x43, 0xf6, 0xa8, 0x88, 0x5a, 0x30, 0x8d, 0x31, 0x31, 0x98, 0xa2, 0xe0, 0x37, 0x07, 0x34  마스터 키: 0xa6, 0xc9, 0x5d, 0xac, 0x84, 0x83, 0x90, 0x60, 0xe0, 0xf3, 0xce, 0xb9, 0x6b, 0x1f, 0x59, 0xb3  4-3.  <공격 논리>   1. 먼저 훈련셋으로 주어진 10000개의 소비전력 파형을 출력해본다.      1. 학습시킬 구간을 결정한다.   위의 소비전력 파형에서 point1000 부터 point2400까지 SubBytes한 부분이라고 추측한다.  SubBytes를 학습 구간으로 정한 이유는 SubBytes가 비선형 함수이기 때문이다.   1. 10000개의 소비 파형을 전부 point1000~2400까지 줄여 SubBytes 소비 파형을 만든다. 2. 학습을 효과적으로 진행하기 위해 파형을 정규화 한다.   정규화는 최소-최대 정규화를 진행하고 값을 -1~1사이로 만든다.   1. 정규화한 10000개의 SubBytes 소비 파형을 출력해본다.        1. 주어진 훈련셋으로 3라운드 SubBytes 출력 값을 구한다. 2. 3라운드 SubBytes 출력 값과 10000개의 SubBytes 소비 파형을 섞는다.   효과적인 학습을 위해 섞어준다.   1. x\_train 은 10000개의 SubBytes 소비 파형으로, y\_train 은 3라운드 SubBytes 출력 값으로 설정한다. 2. 공격하고자 하는 부분을 설정한다.   어떤 부분의 키를 예측하고 싶은 지 고려해야 한다.   1. y\_train에서 공격 부분을 원 핫 인코딩을 진행한다.   원 핫 인코딩을 진행하면, 0~255의 수를 255개의 0과 1개의 1로 표현해준다.  원 핫 인코딩을 진행하면, y\_train은 행이 10000, 열이 256으로 변한다.     1. 신경망 모델을 구성하고 x\_train과 y\_train을 학습한다. 2. 검증셋은 train 셋의 일부를 사용한다. 3. 콜백 함수를 사용하여, 매 에포크마다 검증셋의 손실 값을 체크한 후, 이전 손실 값보다 낮은 경우 모델을 파일로 저장한다.   가장 성능이 좋은 모델을 파일로 저장한다.   1. 매 에포크에 해당하는 훈련셋과 검증셋에 대한 손실 값을 확인한다.     위의 사진은 공격 대상을 0으로 설정했을 때, 이다.   1. 30개의 공격 소비파형을 출력해본다.      1. 30개의 SubBytes 소비 파형 부분으로 자르고 똑같이 정규화를 진행한다.      1. 학습 모델에 30개의 SubBytes 공격 소비 파형을 넣어 30개의 SubBytes 출력 값을 예측한다.   3라운드 Key를 예측하기 위해선, 예측한 3라운드 SubBytes 출력 값을 InvSubBytes를 한다.  그 후, 주어진 공격셋(3round AddRoundKey input)과 xor 연산을 해주어 예측 키를 만든다.   1. 예측 키를 출력해본다.      1. 120 = 0x78 이 가장 많이 나타난다.   공격 대상이 0 이었으므로 3라운드 Key의 0번째 Key는 0x78로 예측한다.   1. 이런 과정을 16번 진행하여 3라운드 전체 Key를 찾아준다.   <신경망 구조>    처음 input은 1400이고 마지막 output은 256이다.  1400인 이유는 SubBytes 소비 파형이 1400포인트이기 때문이다.  또 output이 256인 이유는 0~255중에서 하나를 예측하고 싶기 때문이다.  은닉층의 개수와 노드는 임의로 설정했다(더 좋은 방법이 있을 것으로 생각된다)  <공격 대상이 0번째인 경우 키 히팅 수>    <키 예측 결과>  예측 3라운드 Key: 0x78, 0x6c, 0x16, 0x9d, 0x2b, 0xcf, 0xcf, 0xb1, 0x6, 0x36, 0x36, 0xf3, 0x67, 0xe6, 0xbb, 0x10  예측 마스터 Key: 0x25, 0x83, 0xbf, 0x78, 0xf4, 0x31, 0xeb, 0x5, 0x21, 0xc8, 0xfe, 0x30, 0x32, 0x73, 0x54, 0xcf |