암호분석 HW2

Toy Cipher - TC20

20175204 김영범
2020년 4월 18일

EX_1 : 소스 파일에 있는 Sbox 테이블을 이용한 함수 y = Sbox[x]의 역함수를 나타내는 테이블 ISbox를 만드는 python 프로그램을 만들어라.

```python
def Make_isbox(sbox):
    isbox = []
    for i in range(len(sbox)):
        isbox.append(0x00)
    idx = 0
    for i in sbox :
        isbox[i] = idx
        idx  += 1
    return isbox

#-----------------------------------------
def main():
    isbox = Make_isbox(Sbox)
    for i in range(len(isbox)):
        print ("0x%02x" %(isbox[i]))
```

```
~/vscode/Crypto_Analysis master*
❯ python3 isbox.py
0x52
0x09
0x6a
0xd5
0x30
0x36
0xa5
0x38
0xbf
0x40
0xa3
0x9e
0x81
0xf3
```

TC20에서 사용된 Sbox는 AES SBox입니다. Sbox는 Subbyte연산을 정리해둔 표이며, Subbyte연산은 GF(2^8)에서의 역원 계산후 좌표변환 입니다. 즉 우리는 구한 역원에 대하여 Sbox에 대응하는 Isbox함수를 만들 수 있습니다.

코드를 살펴보면 isbox는 sbox와 list 크기가 같기때문에 0으로 초기화된 256개의 index를 가지는 list를 만들어 주었습니다. 그 후에는 loop를 통하여 0~255의 값을 차례대로 sbox에 대입하여, 얻은 결과를 Isbox의 index로 삼고 초기에 대입해준 0~255의 값을 넣어주어 Isbox를 완성시켰습니다. 우측에 보이는 실행화면은 제가 만든 Isbox값을 출력한 것입니다. 출력한 값은 AES 복호화에 사용하는 Isbox와 일치함을 확인하였고 아래 그림과 같이 isbox.py 안에 행렬로 적어놓았습니다.

```
Isbox = [
#    0     1     2     3     4     5     6     7     8     9     10    11    12    13    14    15
    0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
    0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
    0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
    0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
    0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
    0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
    0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
    0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
    0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
    0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
    0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
    0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
    0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
    0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
    0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
    0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d]
```

EX_2: Toy Cipher TC20의 암호화 과정에 대응되는 복호화 과정을 라이브러리로 작성하라.

```python
def main():
    message = 'LOVE'
    key = [0, 1, 2, 3]
    input_state = [ ord(ch) for ch in message ]
    output_state = [ item for item in TC20_Enc_lib.TC20_Enc(input_state, key)]
    Cipher_message = [chr(num) for num in output_state]

    Recovered_state = [item for item in TC20_Dec(output_state, key)]
    Recovered_char_list = [chr(num) for num in Recovered_state]
    Recovered_message = ''.join(Recovered_char_list)

    print('message =', message)
    print('input plaintext =', input_state)
    print('output ciphertext =', output_state)
    print('Cipher_message =', Cipher_message)
    print('Recovered_state =', Recovered_state)
    print('Recovered_message =', Recovered_message)
```

1

```python
#-- Decrypt Round
def Dec_Round(in_state, rkey):
    out_state1 = [0, 0, 0, 0]
    out_state2 = [0, 0, 0, 0]
    out_state3 = [0, 0, 0, 0]
    out_state1 = LM(in_state)
    out_state2 = ISB(out_state1)
    out_state3 = AR(out_state2, rkey)

    return out_state3
```

2

```
~/vscode/crypto_Analysis master*
❯ python3 TC20_Dec_lib.py
message = LOVE
input plaintext = [76, 79, 86, 69]
output ciphertext = [115, 132, 40, 237]
Cipher_message = ['s', '\x84', '(', 'í']
Recovered_state = [76, 79, 86, 69]
Recovered_message = LOVE
```

3

위의 1,2번 그림은 TC20_Dec_lib 코드의 일부분이며, 3번그림은 실행파일입니다. TC20의 복호화 함수는 암호화에서 Sbox연산을 EX1에서 만들어 놓은 ISbox의 연산으로 바꾸고, 각각의 Round마다 Adroundkey -> Sbox -> Linear Map 순으로 암호화가 이루어진걸 뒤집어서 LM->ISB ->AR순으로 복호화 Round 함수를 만들어야 합니다. 수업시간에도 설명해 주셨듯, AR과 LM은 건드릴 필요가 없습니다. AR은 한번 더실행하는것이 원래 상태로 돌아가는것이며, LM연산에 사용되는 Matrix는 Matrix(LM) = Inv_Matrix(LM) 이기 때문에 원래 코드랑 동일합니다.

복호화함수를 잘만들었는지 확인하기 위하여 1번그림에 보이는 코드를 이용하여, 복호화된 메시지가 원본 메시지와 동일한지 확인해 보았습니다. 3번 그림을 보시면 복호화가 잘되었음을 확인할수 있습니다. Cipher_message는 암호화시 ascii 문자열 값을 벗어나는 숫자가 존재하여, 문자열로 표현하지 않고 list로 표현했습니다. 만약 추후에 cipher_message를 문자열로 만들고 싶으면, 특수문자 혹은 알파벳으로 매핑하는 함수를 추가해야 할 것입니다.

EX_3 : 라이브러리를 활용하여, 암호화 및 복호화를 수행하는 예제 프로그램을 작성하라.

```python
import TC20_Dec_lib
import TC20_Enc_lib
import string
import random

def Make_Random_String(len):
    string_pool = string.ascii_letters # 소문자
    string_digits = string.digits
    random_string = "" #message 임의로 생성
    for i in range(len) :
        random_string += random.choice(string_pool) # 랜덤한 문자열 하나 선택 print(result)
    return random_string

def Run_TC20(msg,key):
    input_state = [ord(ch) for ch in msg]
    output_state = [item for item in TC20_Enc_lib.TC20_Enc(input_state, key)]
    Cipher_message = [chr(num) for num in output_state]
    Recovered_state = [item for item in TC20_Dec_lib.TC20_Dec(output_state, key)]
    Recovered_char_list = [chr(num) for num in Recovered_state]
    Recovered_message = ''.join(Recovered_char_list)
    return Cipher_message , Recovered_message

def main():
    len = 4
    message = Make_Random_String(len)
    key = [0, 1, 2, 3]
    Cipher_message , Recovered_message = Run_TC20(message,key)

    print('message =', message)
    print('Cipher_message =', Cipher_message)
    print('Recovered_message =', Recovered_message)
```

```
~/vscode/Crypto_Analysis master*
❯ python3 Run_TC20.py
message = EWbU
Cipher_message = ['Đ', 'ñ', '\x9b', 'X']
Recovered_message = EWbU
```

4 1 2 3

위의 그림 1,2,3은 Run_TC20.py의 코드 일부분이며, 4번그림은 실행결과 입니다.
1번그림은 msg를 임의로 만들어주는 함수입니다. 파이썬의 random 라이브러리를 이용하여, 대소문자 구별없이 random string을 만들어준 모습입니다. TC20의 msg 는 4byte이므로 3번 그림에서 len = 4로 설정해준 모습니다. 그림 2는 TC20 암복호화를 수행하는 프로그램(Run_TC20)입니다. Return value는 암호화된 메시지와 복호화된 메시지 입니다. 그림 3을 통하여 Run_Tc20을 실행해 보면 그림4를 얻을 수 있고, 암복호화가 잘 된모습을 확인할 수 있습니다.