Information Security

DES

Encryption and Decryption

32171550 박다은

2020-11-7

Contents

1. Overview

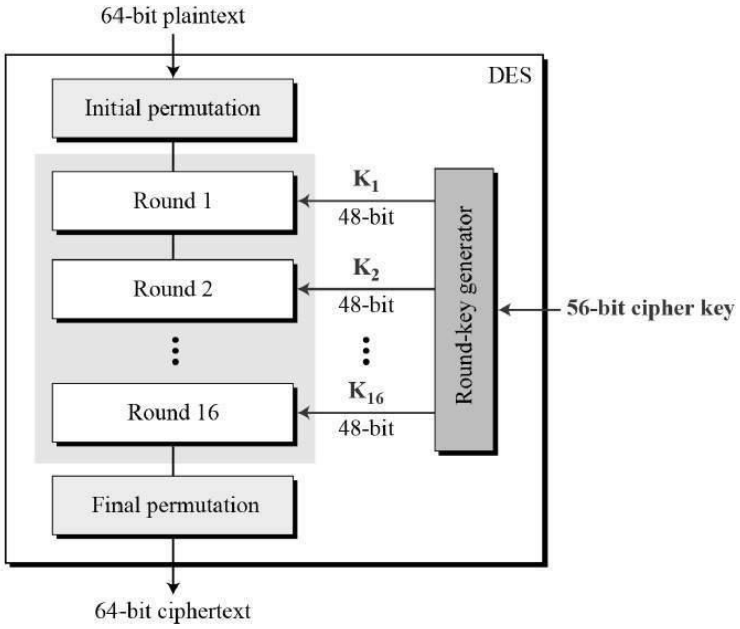
(1) Introduction

DES 는 IBM 에서 고안되어 NIST 가 미준 표준 알고리즘으로 채택된 대칭 암호화 알고리즘이다. 1998 년도에 해독된 암호화 기술이기 때문에, 현재의 일반 컴퓨팅 파워로도 쉽게 뚫린다. DES 는 16 단계 파이스텔 네트워크(Feistel Network)를 거쳐 암호화를 수행한다. 파이스텔 네트워크 구조는 치환(Substitution), 순열(Permutation)을 번갈아 수행하는 구조이다. 64bit 크기의 블록을 입력으로 가지고, 64bit 의 키를 사용하는데 실제키 암호화에 사용되는 키의 사이즈는 54bit 이다.

(2) Goals.

DES 의 암호화와 복호화를 직접 구현해보며 이해한다.

(3) Concepts in DES



DES 의 암호화 과정: Feistel Network 는 Round 로 구성되어 있고, DES 암호화 과정에서 크게 3 가지로 나눌 수 있다

- Initial Permutation & Final Permutation
- 16 Round Function
- Round-Key Generator

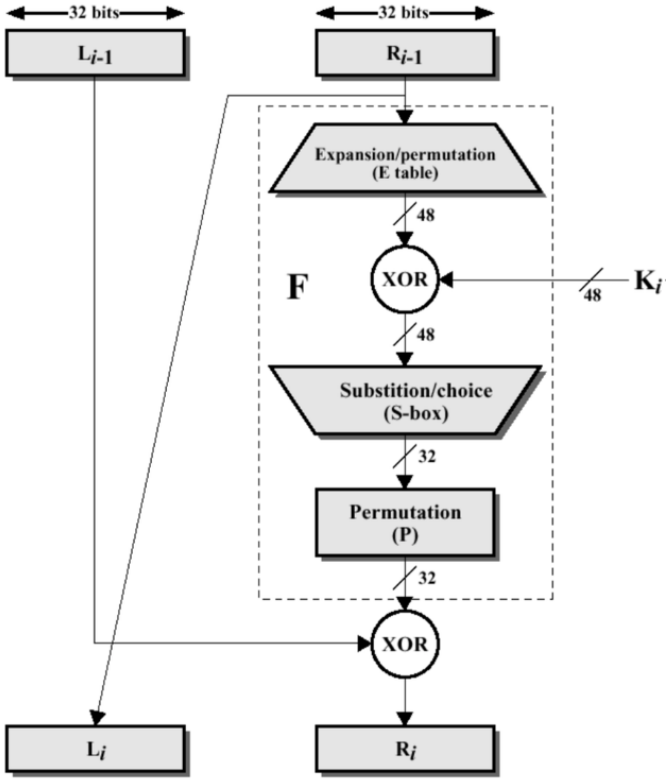
<Initial Permutation. & Final Permutation>

Initial Permutation	*Final Permutation*
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

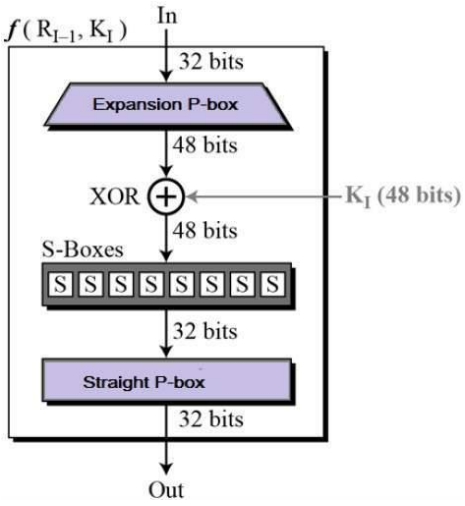
다음과 같은 Permutation Table에 따라서 bit의 교환이 일어나고, 이 부분에서는 암호화를 하지 않는다. 각각 IP, FP 라고 불린다.

<Round Function>

DES 암호화 알고리즘의 실제 암호화는 이 과정에서 일어난다.



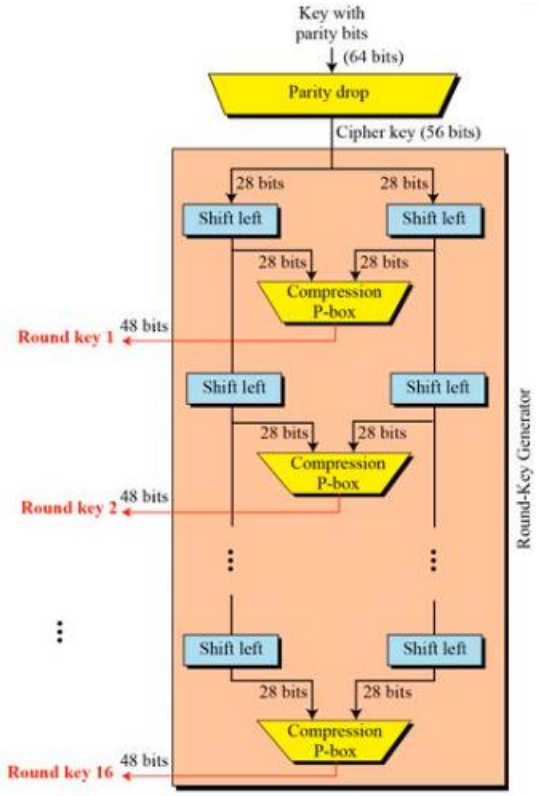
R(오른쪽 bit) 32bit는 Round Function과 XOR이 되어 암호화가 진행된다. L(왼쪽 bit) 32bit는 이와 다르게 암호화는 되지 않고 다음 단계의 R 32bit로 swap된다.



이 그림은 Round Function 의 디테일한 그림이다. 위에서 32bit 가 Expansion P-box 에서 48bit 확장전치가 일어난다. 그 다음 K 라는 서브키와 XOR 연산을 한 후 S-box 에 들어간다.

<Key Generator>

이 과정에서는 DES 에서 사용하는 56bit 암호키를 16 개의 Sub 키로 생성한다.



먼저 PC1 테이블에 의해 56bit 암호키를 얻게 된다. 그 후 두 개의 28bit 블록으로 나뉘게 되고, 두개의 블록은 1,2,9,16 의 Subkey 를 만들 때에는 1bit 만큼 왼쪽으로 shift 되고 나머지의 경우에는 2bit 만큼 왼쪽으로 shift 되게 된다. 각 28bit 블록은 다시 합쳐진 후 PC2 연산을 통해 56bit 가 Round Function 에서 사용될 48bit Subkey 가 된다.

(4) Program Structure

Open_file

+----------------------------

| Enter_Key

| encryption()

| decryption()

+----------------------------

Close_file

각 단계에서 각자의 역할은 결과를 출력하는 것이다.

Encryption 과 decryption 단계에서 각각 새로운 텍스트 파일에 output 을 쓰게 된다.

(5) Build Environment

Compilation: Python 3(Jupyter Notebook)

2.Code & Consideration

(1) 주요 코드

① Written Function

```
18 def split_text_to_bin(text,num):
19     i = 0
20     res_block = []
21     while i < len(text):
22         data = []
23         for j in text[i:i+num]:
24             data.append(ord(j))
25         if len(plain_text) % 8 != 0:
26             if i > len(text) - num:
27                 while len(data) < 8:
28                     data.append(0)
29         res_block.append(byteArrayToString(data))
30         i += num
31     return res_block
```

- 텍스트를 num 길이마다 하나의 블록으로 쪼개고 그 텍스트를 아스키코드를 이용해 bit 로 변환하는 함수이다.
- 만약, 텍스트의 길이가 8 의 배수가 아니라면, 즉 64bit 로 나누어지지 않는다면 마지막 블록의 값 뒤로 '0'을 채워 마지막 블록도 64bit 가 되도록 하였다.

```
34 def key_scheduling(K):
35     #텍스트를 bit로
36     key_ascii = []
37     for i in K:
38         key_ascii.append(ord(i))
39     key_bit = byteArrayToString(key_ascii)
40 
41     #PC1
42     res = ""
43     for i in range(0,56):
44         res += key_bit[PC1[i]]
45 
46     C = res[:28]
47     D = res[28:]
48 
49     ##Making Sub Keys
50     subkey = []
51     for i in range(0,16):
52         C = C[Shift[i]:] + C[:Shift[i]]
53         D = D[Shift[i]:] + D[:Shift[i]]
54         res_key = C+D
55         res = ""
56         for j in range(0,48):
57             res += res_key[PC2[j]-1]
58         subkey.append(res)
```

- 앞서 설명한 DES 암호화 과정 3 가지에서, Key Generator 에 해당하는 부분이다.
- 먼저 텍스트문자로 받은 키를 bit 로 바꿔주는 과정을 거친다.
- 그 후 PC1 테이블을 이용하여 64bit 의 키를 56bit 로 rearrange 한 뒤, 두개의 파트(C:left half, D:right half)로 나누어 주었다.
- 나눠진 C, D 를 1,2,9,16 round 일 때는 1bit 만큼 left shift, 나머지 경우에는 2bit 만큼 left shift 하도록 하였다. 이는 미리 Shift 리스트를 만들어 선언시켜두었다. shift 하는 방법은 shift 된 bit 만큼 후의 비트 string 과 전의비트 string 을 '+' 하여 합쳐주는 방법을 사용하였다.
- Shift 된 C, D 를 다시 합쳐주고 합쳐진 56bit 를 PC2 테이블을 이용하여 다시 48bit 로 rearrange 하였다.
- DES 는 총 16 라운드이므로 이와 같은 방법을 16 번 반복해서 16 개의 subkey 를 생성하도록 하였다.

```
62 def Initial_permutation(text):
63     after_IP = ""
64     for i in range(0,64):
65         after_IP += text[IP[i]-1]
66     L = after_IP[:32]
67     R = after_IP[32:]
68     return L,R
```

- IP 테이블을 이용하여 bit 를 rearrange 하였고, IP 과정 직후 L 과 R 로 나뉘므로 이 함수에 그 과정까지 넣어주었다.

```
71 def F(R,SK):
72     #Expansion R
73     expanded_R = ""
74     for i in range(0,len(E)):
75         expanded_R += R[E[i]-1]
76     temp = str(bin(int(SK,2) ^ int(expanded_R,2))[2:]).zfill(48)
77 
78     #S-box
79     s_box_input = []
80     i = 0
81     while i < len(temp):
82         s_box_input.append(temp[i:i+6])
83         i += 6
84 
85     s_box_output = []
86     for j in range(0,len(s_box_input)):
87         row = int((s_box_input[j][0]+s_box_input[j][5]),2)
88         col = int((s_box_input[j][1]+s_box_input[j][2]+s_box_input[j][3]+s_box_input[j][4]),2)
89 
90         s_box_output.append(str(bin(SBOX[j][row][col])[2:]).zfill(4))
91 
```

```
92      output_str = ""
93      for k in s_box_output:
94          output_str += k
95
96      return output_str
```

- F-box 과정을 함수로 만든 것이다.
- 먼저 32bit 의 R 을 E 테이블을 사용하여 48bit 로 확장시킨다. 그 후 subkey 와 확장시킨 R 을 XOR 해주었다.
- 그렇게 만들어진 48bit 를 6bit 씩 나눠 s-box 의 input 으로 리스트에 저장시켰다. 그 후 각 input 에 맞게 s-box 에서 output 을 골라 output 리스트로 저장하도록 하였다.
- 만약 input 이 '100101'이라고 하면, row 는 '100101'[0] = '1' 과 '100101'[5] = '1'을 string '+'연산을 통하여 '11'이 되고 이 2 진수를 정수로 바꾸면 3 이 된다. 따라서 row=3 이다. 마찬가지로, col 은 '100101'[1] = '0', '100101'[2] = '0', '100101'[3] = '1', '100101'[4] = '0'을 합친 '0010' = 2 이고 그러므로 col = 2 이다.
- S-box 에서 row 와 col 에 맞는 값을 찾아 이를 다시 이진수문자열로 output 리스트에 저장하도록 하였다. 이 4bit * 8 리스트를 다시 32bit 의 문자열로 합쳐 이를 반환하도록 하였다.

```
98  def feistel_round(L,R,SK):
99      temp = R
100     after_F = F(R,SK)
101
102     #Permutaion(Another Bit Shuffling)
103     permute = ""
104     for j in range(0, len(P)):
105         permute += after_F[P[j]-1]
106
107     R = str(bin(int(L,2) ^ int(permute,2))[2:]).zfill(32)
108     L = temp
109
110     return L,R
```

- DES 의 16round 중 한 라운드에 해당하는 함수이다. 이 함수를 16 번 반복해야 한다.
- 먼저 앞서 설명한 F 함수를 이용하여 나온 output 블록을 P 테이블을 이용하여 bit shuffling 해준다. Shuffling 된 bit 와 L 을 XOR 한 것이 다음라운드의 R 이 되고, 이번 라운드의 R 이 다음 라운드의 L 이 된다.

```
112 def final_permutation(bits):
113     res = ""
114     for j in range(0,64):
115         res += bits[FP[j]-1]
116     return res
```

- 16 라운드가 끝난 뒤 최종 bit 를 FP 테이블을 통해 rearrange 하는 함수이다.

```
119 def block_to_text(block):
120     text = ""
121     for i in range(0,len(block)):
122         res = ""
123         for j in range(0,64,8):
124             bin_text = block[i][j:j+8]
125             res+= chr(int(bin_text,2))
126         text += res
127     return text
```

- Bit block 을 합쳐 텍스트로 바꾸는 함수이다.
- 한 블록의 64bit 를 8bit 씩 나눠 정수로 바꾼 뒤 아스키코드를 통해 문자로 바꾸도록 하였다.

② Encryption

```
1  def encryption(PlainText):
2      plain_block = split_text_to_bin(PlainText,8)
3      cipher_block = []
4
5      #Key Derivation
6      SubKey = key_scheduling(key)
7
8      #Encode the input text block
9      for i in range(0,len(plain_block)):
10         L, R = Initial_permutation(plain_block[i])
11
12         for i in range(0,16):
13             L,R = feistel_round(L,R,SubKey[i])
14
15         cipher_block.append(final_permutation(R+L))
16
17     CipherText = block_to_text(cipher_block)
18     return CipherText
```

- DES 는 블록암호이므로 평문을 split_text_to_bin 함수를 이용하여 64bit 씩 블록으로 나누어주도록 하였다.
- 그 후 key_scheduling 함수를 이용하여 16 개의 subkey 들을 만들어 주었다.
- Initial_permutation 함수를 통해 IP 를 적용시키고 여기서 반환된 L, R 을 feistel_round 함수를 통해 한 라운드씩 16 번 반복하도록 하였다.
- 마지막으로 final_permutation 를 통해 최종 bit 를 한번 더 rearrange 하였다.
- 이 과정을 64bit 씩 나눈 블록의 수만큼 for 문을 통해 반복하였고, 최종 블록을 block_to_text 함수를 통해 최종 암호문이 나오도록 하였다.

③ Decryption

```
def decryption(CipherText):
    cipher_block = split_text_to_bin(CipherText,8)
    plain_block = []

    #Key Derivation
    SubKey = key_scheduling(key)

    #Decode the input text block
    for i in range(0,len(cipher_block)):
        L, R = Initial_permutation(cipher_block[i])

        for i in range(16,0,-1):
            L,R = feistel_round(L,R,SubKey[i-1])

        plain_block.append(final_permutation(R+L))

    DecText = block_to_text(plain_block)
    return DecText
```

- 암호화할 때와 같은 이유로 암호문을 split_text_to_bin 을 사용하여 블록으로 쪼개주었다.
- 또한 역시 key_scheduling 함수를 통하여 16 개의 subkey 들을 생성하였다.
- Initial_permutation 함수를 통해 IP 과정을 적용시키고, feistel_round 함수를 16 번 반복시켰다. 암호화때와 다른 점은 subkey 순서를 역순으로 행해야 하므로, for 문을 암호화때와 반대로 반복시켰다.
- 마찬가지로 final_permutation 함수를 통해 FP 과정을 적용시켰고 여태까지의 과정을 암호문 블록의 개수만큼 반복시켰다.
- 그렇게 생성된 해독문 bit 블록을 block_to_text 함수를 통해 텍스트로 다시 바꾸어주면 최종 해독문이 나오게 된다.

④ main

```
#input 파일 열기
in_f = open('des_input.txt','r',encoding='UTF8')
line = in_f.readline()
plain_text = line
while line:
    line = in_f.readline()
    plain_text += line
in_f.close()
```

- 과제 워드파일을 텍스트파일로 변환시킨 후 그것을 input 파일로 사용하였다.
- readline() 함수를 통해 읽은 값을 DES 의 평문으로 사용하였다.

```
#Key 지정
while True:
    key = input("64bit Key를 입력해주세요: ")
    if len(key) == 8:
        break
    print("다시 입력하세요.")
```

- Key 는 프로그램을 이용하는 유저가 정하도록 하였다.
- 8 글자의 키를 입력하지 않았으면 다시 입력하도록 예외처리를 하였다.

```
print('₩n₩n평문은 다음과 같습니다.₩n',plain_text[0:100],
      '₩n₩n~(중략)~₩n₩n', plain_text[len(plain_text)-100:len(plain_text)])

#encryption
cipher_text = encryption(plain_text)
c_f = open('cipher_text.txt', 'w', encoding='UTF8')
c_f.write(cipher_text)
print('₩n₩n암호문은 다음과 같습니다.₩n',cipher_text[0:100],
      '₩n₩n~(중략)~₩n₩n', cipher_text[len(plain_text)-100:len(cipher_text)])

#decryption
decrypted_text = decryption(cipher_text)
d_f = open('decrypted_text.txt', 'w', encoding='UTF8')
d_f.write(decrypted_text)
print('₩n₩n해독문은 다음과 같습니다.₩n',decrypted_text[0:100],
      '₩n₩n~(중략)~₩n₩n', decrypted_text[len(plain_text)-100:len(decrypted_text)])
```

- 출력은 텍스트의 길이가 너무 길어 각각 맨앞, 맨뒤 100 글자씩만 보이도록 하였다.
- encryption 함수를 통해 평문을 암호화해주고 암호문을 cipher_text.txt 파일에 저장하도록 하였다.
- 마찬가지로 decryption 함수를 통해 암호문을 해독해주고 해독문을 decrypted_text.txt 파일에 저장하도록 하였다.

(2) Screenshots

① 출력화면

```
64bit Key를 입력해주세요: InfoSecu


평문은 다음과 같습니다.
 Introduction to Information Security #2: DES (Data Encryption Standard)
due by Oct. 31th, 2020

Intr

~(중략)~
```

```
 on text.

Optional Req3. Make a decoding method that gets you back the original text.

Seehwan Yoo
```

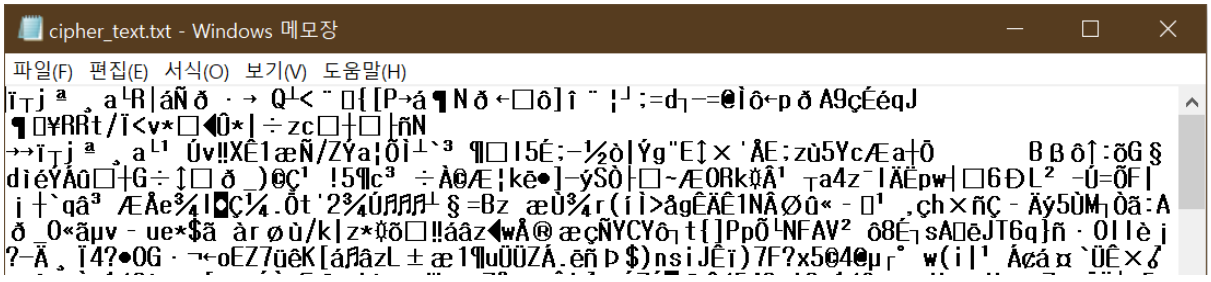
암호문은 다음과 같습니다.

```
 ï┬↔j ª ¸ a└R|á□Ñð · → Q┴<□ ¨ □{[P→á ¶ N ð ←□ô] □□î ¨ ¦┘;□=□□□d┐-=@ìô←p ð A9çÉéqJ
¶ □¥RRt/Ï<□v*□◀Û□*|÷zc□┼□├ñN
→→ï┬↔j

~(중략)~

ù<Ê ß }÷ Wvëkð» ð &i□T%Ï□Nj□ý'2h8┘ ª § •ùìT•□ß•@ÑÅ□Éí|#W□Å←k × Yóë□Ck□eJ□ãi~L□□uù┐▼↕
¥Ñ→ ý✲□¾yŸ✲ß 0; § ▼s┘`ý□Æ ¤@□7□ÚÓF
```

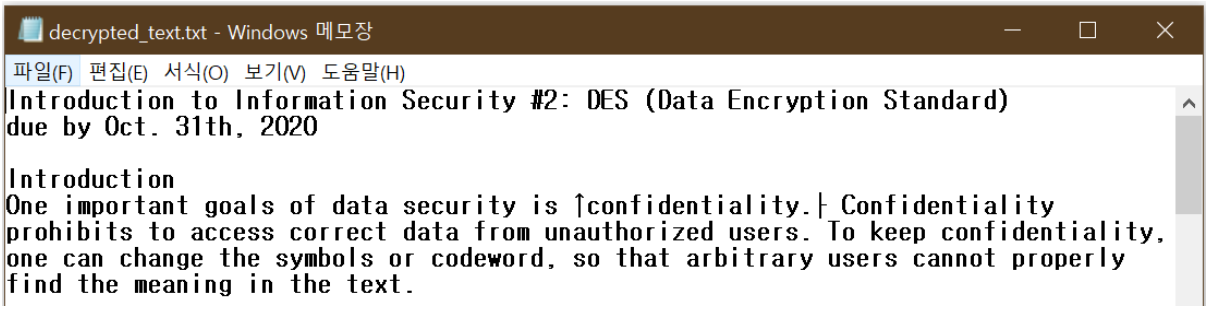
해독문은 다음과 같습니다.

```
 Introduction to Information Security #2: DES (Data Encryption Standard)
due by Oct. 31th, 2020

Intr

~(중략)~

 on text.

Optional Req3. Make a decoding method that gets you back the original text.

Seehwan Yoo
```

② cipher_text.txt



③ decrypted_text.txt



(3) Problems & Solutions

① 파일 입출력

- 처음에 텍스트파일을 읽을 때, 예전과 같이 그냥 f = open('des_input.txt', 'r')으로 입력하여 읽도록 하였는데, 어떠한 이유인지 계속 오류가 났다.
- 구글을 찾아보니 뒤에 encoding = 'UTF8'을 추가하라고 하여, f = open('des_input.txt', 'r', encoding = 'UTF8')처럼 입력하였더니 오류가 사라졌다.
- 나중에 텍스트파일을 작성할 때도 같은 오류가 나서 같은 방법으로 해결했다.

② XOR

- XOR 을 할 때 오류가 났다가 나지 않았다가 했었다. String 을 2 진수로 바꾸는 과정에서 좀 헤맸던 지라 그게 오류의 이유인 줄 알고 그 부분을 중점적으로 봤었다.
- 오류를 찾을 때 애매한부분을 출력해보며 찾는 편인데 이번에도 그렇게 해보니 XOR 을 한 뒤 앞부분이 0 이면 그 부분이 다 사라졌던 것이 이유였다.
- zfill()함수를 사용해 각 비트에 맞게 앞부분을 0 으로 채워주도록 하여서 해결했다.

③ 블록 나누기

- 블록을 나눌 때 평문의 글자 수가 8 의 배수가 아니라면 마지막 블록이 64bit 가 되지 못하므로 오류가 날 것이라는 것은 인지하고 있었다.
- 따라서 위처럼 zfill()함수를 사용하여 64bit 로 맞추어주었고 따라서 오류가 나지는 않았다.
- 하지만 암호화를 한 것을 다시 복호화를 할 때 마지막 부분만 이상한 값이 나오게 되었고 zfill 을 사용하여 앞을 0 으로 채우던 것을, 뒷부분을 0 으로 채우게 하는 방법으로 바꾸어 해결하였다.

(4) Personal Feeling

충분한 이해 후에 코드작성을 시작했어야 했는데, 시험기간과 겹쳐 시간이 충분하지 않았고 쫓기는 느낌이 들어 개념이 긴가민가할 때 코드를 작성하기 시작했다. DES 에는 많은 과정들이 있기 때문에 한 과정의 코드를 다 작성할 때마다 다음 과정의 개념들을 다시 살펴봐야 했었다. 따라서 이해를 완전히 하고 코드를 작성하는 것보다 결과적으로 더 오랜 시간이 걸리게 된 것 같다.

먼저 암호화를 구현하고 그 이후에 복호화를 구현했는데, 암호화를 완벽히 구현시켰다고 생각했는데 복호화를 해보니 제대로 구현된 것이 아니었다. 알고 보니 F-box 의 과정 중에서 한 부분이 빠져 있었던 것이다. 완전하게 이해를 하지 않고 코드를 작성한 결과 같았다. 다음부터는 정말 개념을 확실히 익힌 후 코드를 작성해야겠다고 다시 한번 깨닫았다.

DES 와 블록암호에 대해 공부를 꽤 했다고 생각했는데, 코드를 작성하다 보니 모르고 새로웠던 부분도 있었다. 직접 코드를 작성해 봄으로써 잊지 못할 오래가는 지식이 된 것 같다.

저번 과제였던 Base64 에 비해 이번 과제는 난이도가 좀 있던 과제였다. 저번 과제처럼 코드를 작성하는 게 순탄하진 않았지만 그만큼 뿌듯함은 더 큰 것 같다. 이렇게 큰 과제를 하나하나씩 해내다 보면 점점 실력이 늘어가는 게 느껴진다. 정보보호에 관해 흥미도 많이 생겼고, 다음 과제도 열심히 참여하여 완벽히 해내고 싶다.