Cryptography Homework 3

- عباس یزدان مهر
- 99243077

1.

با استفاده از کد زیر یک IP, IP-1 تصادفی تولید می کنیم:

```
import random

def rand_ip_gen(iplen=64):
    available_indexes = [i for i in range(iplen)]
    ip = [0] * iplen
    for i in range(iplen):
        randi = random.randint(0, len(available_indexes)-1)
        ip[i] = available_indexes.pop(randi)
    return ip

def get_ip_inverse(ip):
    iplen = len(ip)
    ip_inverse = [0] * iplen
    for i in range(iplen):
        ip_inverse[ip[i]] = i
    return ip_inverse

def print_ip(ip, w=8, h=8):
    for i in range(w):
        print(' & '.join(map(str, ip[i*8:i*8+8])))

def test():
    rand_ip = rand_ip_gen()
    ip_inverse = get_ip_inverse(rand_ip)

    print('-- IP')
    print_ip(rand_ip)

    print('-- IP Inverse')
    print_ip(ip_inverse)
```

- IP:

$$\begin{bmatrix} 46 & 16 & 55 & 41 & 25 & 34 & 17 & 59 \\ 54 & 27 & 31 & 38 & 60 & 28 & 3 & 19 \\ 0 & 52 & 4 & 50 & 63 & 40 & 51 & 53 \\ 8 & 35 & 15 & 32 & 20 & 47 & 6 & 48 \\ 10 & 43 & 5 & 49 & 24 & 22 & 36 & 62 \\ 2 & 42 & 39 & 56 & 21 & 30 & 29 & 13 \\ 9 & 26 & 37 & 11 & 14 & 57 & 33 & 61 \\ 1 & 44 & 18 & 12 & 45 & 7 & 58 & 23 \end{bmatrix}$$

- IP Inverse:

$$\begin{bmatrix} 16 & 56 & 40 & 14 & 18 & 34 & 30 & 61 \\ 24 & 48 & 32 & 51 & 59 & 47 & 52 & 26 \\ 1 & 6 & 58 & 15 & 28 & 44 & 37 & 63 \\ 36 & 4 & 49 & 9 & 13 & 46 & 45 & 10 \\ 27 & 54 & 5 & 25 & 38 & 50 & 11 & 42 \\ 21 & 3 & 41 & 33 & 57 & 60 & 0 & 29 \\ 31 & 35 & 19 & 22 & 17 & 23 & 8 & 2 \\ 43 & 53 & 62 & 7 & 12 & 55 & 39 & 20 \end{bmatrix}$$

2.

با استفاده از کد زیر و ساختن لیست نگاشتی که در این گسترش و جایگشت انجام می شود نگاشت ها را انجام می دهیم:

```
mapper = [32, 1, 2, 3, 4, 5,
          4, 5, 6, 7, 8, 9,
          8, 9, 10, 11, 12, 13,
          12, 13, 14, 15, 16, 17,
          16, 17, 18, 19, 20, 21,
          20, 21, 22, 23, 24, 25,
          24, 25, 26, 27, 28, 29,
          28, 29, 30, 31, 32, 1]

def expansion(inp:list, expmapper) -> list:
    result = [0] * len(expmapper)
    for i in range(len(expmapper)):
        result[i] = inp[expmapper[i] - 1]
    return result

def test():
    global mapper
    num = 0x6A31E9FB
    bits_list = list(bin(num)[2:].zfill(32))
    print(''.join(map(str, bits_list)))
    result = expansion(bits_list, mapper)
    print(''.join(map(str, result)))
```

```
input:  01101010001100011110100111111011 # 0x6A31E9FB
result: 10110101010000011010001111110101001111111110110
```

3.

S-box ها تنها جزء غیرخطی هستند و وجود آنها برای امنیت بالای یک رمزنگاری لازم است به این دلیل که اگر این عنصر وجود نداشته باشد تمام توابع موجود خطی می شوند و توابع خطی با داشتن چندین معادله از ورودی ها و خروجی ها قابل حل برای یافتن کلید است که اصلا خوب نیست، از طرفی به دلیل غیر خطی بودن این بخش ها و برگشت پذیر نبودن آنها با داشتن خروجی ها عملا پیدا کردن ورودی ها غیرممکن است و در بهترین حالت می توان آنها را حدس زد یا تقریب خطی از آنها پیدا کرد و بهترین راه حل احتمالا پیمایش کل فضای

حالت است که آن ها ممکن نیست. پس وجود این بخش در توابع رمزنگاری الزامی است.

4.

- a

  on GF(5)

$4x^7+2x^4+3x^3+x+1$ $4x^7+2x^6+4x^5$	x^3+3x^2+x $q=4x^4+3x^3+2x^2+3x+2$
$3x^6+1x^5+2x^4+3x^3+x+1$ $3x^6+9x^5+3x^4$	
$2x^5+4x^4+3x^3+x+1$ $2x^5+6x^4+2x^3$	
$3x^4+1x^3+x+1$ $3x^4+9x^3+3x^2$	
$2x^3+2x^2+x+1$ $2x^3+6x^2+2x$	
$r=x^2+4x+1$	

- b

$$\boxed{\begin{array}{c} x^8=x^4+x^3+x+1 \\ x^9=x^5+x^4+x^2+x \end{array}}$$

$$\begin{aligned} h&=(x^5+x^4+x^3)(x^4+x^2+x) \\ &=x^9+x^7+x^6+x^8+x^6+x^5+x^7+x^5+x^4 \\ &=x^9+x^8+x^4 \\ &=(x^5+x^4+x^2+x)+(x^4+x^3+x+1)+x^4 \\ &=x^5+x^4+x^3+x^2+1 \end{aligned}$$

- c

$$(C2)_{hex}=(1100|0010)_2=x^7+x^6+x$$
$$(x^7+x^6+x)^{-1} \mod x^8+x^4+x^3+x+1$$
$$EEC(x^7+x^6+x,x^8+x^4+x^3+x+1):$$

i	q_i	r_i	u_i	v_i
0	$-$	$x^8+x^4+x^3+x+1$	0	1
1	$-$	x^7+x^6+x	1	0
2	x	$x^7+x^4+x^3+x^2+x+1$	x	1
3	1	$x^6+x^4+x^3+x^2+1$	$x+1$	1
4	x	x^5+x^2+1	x^2	$x+1$
5	x	x^4+x^2+x+1	x^3+x+1	x^2+x+1
6	x	x^3+x+1	x^4+x	x^3+x^2+1
7	x	1	$x^5+x^3+x^2+x+1$	$x^4+x^3+x^2+1$

$$(x^7+x^6+x)^{-1} \mod x^8+x^4+x^3+x+1=u_7$$
$$=x^5+x^3+x^2+x+1=(0010|1111)_2=(2F)_{hex}$$

5.

$$S(0x51)=?$$

$$0x51 = 0b0101|0001 = x^6 + x^4 + 1$$
$$EEC(x^6 + x^4 + 1, x^8 + x^4 + x^3 + x + 1):$$

i	q_i	r_i	u_i	v_i
0	$-$	$x^8 + x^4 + x^3 + x + 1$	0	1
1	$-$	$x^6 + x^4 + 1$	1	0
2	$x^2 + 1$	$x^3 + x^2 + x$	$x^2 + 1$	
3	$x^3 + x^2 + x$	$x^2 + 1$	$x^5 + x^4 + x^2 + x + 1$	
4	x	x^2	$x^6 + x^5 + x^3 + x + 1$	
5	1	1	$x^6 + x^4 + x^3 + x^2$	

$$x^6 + x^4 + x^3 + x^2 = (0101|1100)_2$$
$$s = 0101|1100 \oplus 1011|1000 \oplus 0111|0001 \oplus$$
$$1110|0010 \oplus 1100|0101 \oplus 0110|0011$$
$$= 1101|0001 = (D1)_{hex}$$

6.

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0A \\ 05 \\ 01 \\ 03 \end{bmatrix}$$

$$d_0 = (2 \cdot A) + (3 \cdot 5) + (1 \cdot 1) + (1 \cdot 3)$$
$$= (x \cdot (x^3 + x)) + ((x + 1) \cdot (x^2 + 1)) + (1) + (1 \cdot (x + 1))$$
$$= (x^4 + x^2) + (x^3 + x^2 + x + 1) + 1 + x + 1$$
$$d_0 = x^4 + x^3 + 1 = 25$$

$$d_1 = (1 \cdot A) + (2 \cdot 5) + (3 \cdot 1) + (1 \cdot 3) = (1 \cdot A) + (2 \cdot 5)$$
$$= (1 \cdot (x^3 + x)) + (x \cdot (x^2 + 1)) = x^3 + x + x^3 + x$$
$$d_1 = 0$$

$$d_2 = A + 5 + 2 + (3 \cdot 3)$$
$$= x^3 + x + x^2 + 1 + x + x^2 + 1$$
$$d_2 = x^3 = 8$$

$$d_3 = (3 \cdot A) + 5 + 1 + (2 \cdot 3)$$
$$= ((x + 1) \cdot (x^3 + x)) + x^2 + 1 + 1 + (x \cdot (x + 1))$$
$$= (x^4 + x^3 + x^2 + x) + x^2 + x^2 + x$$
$$d_3 = x^4 + x^3 + x^2 = 28$$

$$\begin{bmatrix} d_0 = 25 \\ d_1 = 0 \\ d_2 = 8 \\ d_3 = 28 \end{bmatrix}$$

7.

$$L_2 = R_1 = 0x6A31E9FB$$

قسمت گسترش را در سوال ۲ حل کردیم شش تا شش تا جدا می کنیم:

```
101101
010100
000110
100011
111101
010011
111111
110110
```

خروجی ها را از sbox ها به ترتیب می نویسیم:

```
0001
0110
0001
1100
0110
0110
1101
0111
```

حال جایگشت را با کد زیر انجام می دهیم:

```
def permute(s:list): # len(s) = len(p) = len(p)
    p = [
        16, 7  ,20 ,21, 29, 12, 28 ,17,
        1  ,15 ,23 ,26, 5  ,18 ,31 ,10,
        2  ,8  ,24 ,14 ,32 ,27 ,3,  9,
        19, 13, 30, 6,  22, 11, 4 , 25,
    ]
    result = [0] * len(s)
    for i in range(len(s)):
        result[p[i] - 1] =  s[i]
    return result

def test():
    s = list('00010110000111000110011011010111')
    result = permute(s)
    print(''.join(map(str, result)))
```

```
00111101001110000110100111110000
```

حال کافی است این عدد را با قسمت L1 جمع کنیم:

```
00111101001110000110100111110001
```