CS553: Cryptography

Assignment 4: Solutions

Rohit Das (11910230)

August 28, 2019

1. Fiestal and SPN

1..1 Fiestal Ciphers

- **Blowfish:**
  - Block Size: 64 bits
  - Key Size: 32 - 448 bits
  - Susceptible to 2^{nd}-order differential attack.
- **Data Encryption Standard (DES):**
  - Block Size: 64 bits
  - Key Size: 56 bits (+8 parity bits)
  - Considered insecure because of feasibility of brute-force attacks.
- **Rivest Cipher (RC5):**
  - Block Size: 32/64 (suggested)/128 bits
  - Key Size: 0 - 2040 bits (128 bits suggested)
  - Susceptible to differential attacks using 2^{44} plaintexts.
- **Information Concealment Engine (ICE):**
  - Block Size: 64 bits
  - Key Size: 64 bits
  - Differential attacks can break 15 of 16 rounds with complexity 2^{56}.
- **KASUMI:**
  - Block Size: 64 bits
  - Key Size: 128 bits

1..2 Substitution-Permutation Network (SPN)

- **Advanced Encryption Standard (AES):**
  - Block Size: 128 bits
  - Key Size: 128/192/256 bits
  - For AES-128, key can be recovered with complexity $2^{126.1}$ (biclique attack).
- **3-Way:**
  - Block Size: 96 bits
  - Key Size: 96 bits
  - Vulnerable to related key cryptanalysis.
- **Kuznyechik:**
  - Block Size: 128 bits

  - Key Size: 256 bits
  - Vulnerable to meet-in-the-middle attack on 5 rounds.

- **SAFER K-64 (Safer And Faster Encryption Routine):**
  - Block Size: 64 bits
  - Key Size: 64 bits

- **Square:**
  - Block Size: 96 bits
  - Key Size: 96 bits
  - Precursor to AES.

2. Random SBox (4-bit)

Python code (Python 3): Random_s-box_gen.py

```
import numpy as np


def rn_box(n): # random permutation of input symbols
    arr = [hex(int(i)) for i in np.arange(n)]# hex symbols
    # mapping each symbol to its random substitute
    return {i:j for i,j in zip(arr,
                               np.random.permutation(arr))}


def main(sbox): # takes plaintext and performs confusion
    p = input("Enter your plaintext:")
    if not all(x in [format(i,'x') for i in sbox]
               for x in p):# checking if in range [0-f]
        exit("Enter valid characters[0-f]!")
    str1 = {format(i,'x'):format(j,'x') for i,j
    in sbox.items()}
    return "".join([str1[i] for i in p])
```

```
sbox = {0x0: 0x5, 0x1: 0x4, 0x2: 0xd, 0x3: 0x1, 0x4: 0x2,
        0x5: 0xf, 0x6: 0x6, 0x7: 0x0, 0x8: 0x8, 0x9: 0xc,
        0xa: 0xb, 0xb: 0x9, 0xc: 0x7, 0xd: 0xe, 0xe: 0xa,
        0xf: 0x3} # generated using rn_box(int)


# print(rn_box(16)) # to generate a random s-box
print(main(sbox))
```

The S-Box to be used subsequently is as follows:

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	5	4	d	1	2	f	6	0	8	c	b	9	7	e	a	3

3. Differential Distribution Table (DDT)

Python code (Python 3): DDT_s-box.py

```
sbox = {0x0: 0x5, 0x1: 0x4, 0x2: 0xd, 0x3: 0x1, 0x4: 0x2,
        0x5: 0xf, 0x6: 0x6, 0x7: 0x0, 0x8: 0x8, 0x9: 0xc,
        0xa: 0xb, 0xb: 0x9, 0xc: 0x7, 0xd: 0xe, 0xe: 0xa,
        0xf: 0x3} # S-Box produced earlier


def ddt(sbox):
    lst1 = [format(i, 'x') for i,j in sbox.items()]
    print("in\\out|", ("{:>3}"*len(lst1)).format(*lst1))
    print('-'*56)
    for diff in sbox:
        u1 = [hex(i^diff) for i in sbox]
        S_u0 = [hex(j) for i,j in sbox.items()] # S[u0]
        S_u1 = [hex(sbox[(int(i,0))]) for i in u1] # S[u1]
        S_u0_x_S_u1 = [hex(int(i,0)^int(j,0)) for i,j
```

```
        in zip(S_u0,S_u1)] # S[u0] xor S[u1]
        # counting occurences and replacing 0 with '-'
        count = {hex(i):'-' if S_u0_x_S_u1.count(hex(i))
        == 0 else S_u0_x_S_u1.count(hex(i)) for i in sbox}
        # format output as a table
        lst = [str(i) for j,i in count.items()]
        frmt = "{:>3}"*len(lst)
        print(str(format(diff,'x')+'   |').rjust(7),frmt.
            format(*lst))
ddt(sbox)
```

The generated DDT is as follows:

in\out	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	2	2	-	2	-	2	-	-	4	-	-	2	2	-	-
2	-	-	-	2	2	4	-	-	2	-	-	-	-	4	-	2
3	-	2	2	-	6	-	-	2	-	4	-	-	-	-	-	-
4	-	4	2	-	-	-	-	2	-	-	2	4	-	-	-	2
5	-	-	-	2	-	-	4	2	2	-	2	2	-	2	-	-
6	-	-	2	2	2	-	-	2	-	-	-	-	2	-	2	4
7	-	-	4	2	-	4	2	-	-	-	-	2	-	-	2	-
8	-	2	-	2	-	2	2	-	4	-	-	-	2	2	-	-
9	-	-	-	-	2	2	-	-	2	2	4	-	4	-	-	-
a	-	2	-	-	-	2	-	-	2	-	-	-	2	4	4	-
b	-	4	-	-	-	2	-	2	2	2	-	-	2	-	-	2
c	-	-	4	2	-	-	-	2	-	2	4	-	-	2	-	-
d	-	-	-	2	-	-	-	2	-	-	-	6	-	-	4	2
e	-	-	-	-	-	-	2	2	-	2	2	-	2	-	2	4
f	-	-	-	2	2	-	4	-	2	-	2	2	-	-	2	-

Actual output(file): output.txt

The maximum differential probability of this S-Box is $\frac{6}{16}$ for the (input,output) difference transactions (3,4) and (d,b).

4. SBox as a Boolean Function

The Boolean table for the above S-Box is as follows:

x	x_0	x_1	x_2	x_3	y_0	y_1	y_2	y_3	$y = S[x]$
0	0	0	0	0	0	1	0	1	5
1	0	0	0	1	0	1	0	0	4
2	0	0	1	0	1	1	0	1	d
3	0	0	1	1	0	0	0	1	1
4	0	1	0	0	0	0	1	0	2
5	0	1	0	1	1	1	1	1	f
6	0	1	1	0	0	1	1	0	6
7	0	1	1	1	0	0	0	0	0
8	1	0	0	0	1	0	0	0	8
9	1	0	0	1	1	1	0	0	c
a	1	0	1	0	1	0	1	1	b
b	1	0	1	1	1	0	0	1	9
c	1	1	0	0	0	1	1	1	7
d	1	1	0	1	1	1	1	0	e
e	1	1	1	0	1	0	1	0	a
f	1	1	1	1	0	0	1	1	3

The 4 K-maps for the variables are as follows:

X_0X_1 \ X_2X_3	00	01	11	10
00	0	0	0	1
01	0	1	0	0
11	0	1	0	1
10	1	1	1	1

$y_0 = \sum(2, 5, 8, 9, 10, 11, 13, 14)$

X_0X_1 \ X_2X_3	00	01	11	10
00	1	1	0	1
01	0	1	0	1
11	1	1	0	0
10	0	1	0	0

$y_1 = \sum(0, 1, 2, 5, 6, 9, 12, 13)$

X_0X_1 \ X_2X_3	00	01	11	10
00	0	0	0	0
01	1	1	0	1
11	1	1	1	1
10	0	0	0	1

$y_2 = \sum(4, 5, 6, 10, 12, 13, 14, 15)$

X_0X_1 \ X_2X_3	00	01	11	10
00	1	0	1	1
01	0	1	0	0
11	1	0	1	0
10	0	0	1	1

$y_3 = \sum(0, 2, 3, 5, 10, 11, 12, 15)$

The formulas for the variables are as follows:

$$y_0 = \sum(2, 5, 8, 9, 10, 11, 13, 14) = x_0x_1' + x_1x_2'x_3 + x_0x_2x_3' + x_1'x_2x_3'$$

$$y_1 = \sum(0, 1, 2, 5, 6, 9, 12, 13) = x_2'x_3 + x_0'x_1'x_2' + x_0'x_2x_3' + x_0x_1x_2'$$

$$y_2 = \sum(4, 5, 6, 10, 12, 13, 14, 15) = x_0x_1 + x_0'x_1x_2' + x_1x_2x_3' + x_0x_2x_3'$$

$$y_3 = \sum(0, 2, 3, 5, 10, 11, 12, 15) = x_1'x_2 + x_0'x_1'x_3' + x_0'x_2x_3 + x_0'x_1x_2'x_3 + x_0x_1x_2'x_3'$$