Analysis Assignment 2

1. Read the design of Feistel cipher Twofish, which was one of the AES finalists, and implement it. (25)
2. Implement AES (Rijndeal) along with its KSA. (25)
3. Read the specification for the small scale variant of the SPN Present from [here]().
   a. Implement it along with its KSA. Your algorithm should work for any given $n > 5$, s.t., $n = 0 \bmod 4$ and $n <= 80$. Use the test vectors given at the end of the article to verify the correctness of your code. (10)
   b. Implement a Linear cryptanalysis of the cipher, i.e., (20)
      i. Randomly generate an input (plaintext) mask and a mask to the output of the last but one round.
      ii. Generate N random plaintext-ciphertext pairs.
      iii. For each such pair, partially decrypt the ciphertext by one-round to evaluate the xor of the plaintext and the output masks.
      iv. Let count denote the number of times the xor value is equal to 1 out of N such pairs.
      v. Repeat the above S = 100N many times to generate empirical distributions $count_1/N$, $count_2/N$, … , $count_S/N$. (*count_i/S)
      vi. Plot the empirical distribution along the y-axis and the values {0, 1, ... , N} along the x-axis and compare it (pictorially) with the normal distribution with mean N/2 and variance N/4.
   c. Implement a Differential cryptanalysis of the cipher, i.e., (20)
      i. Randomly generate an input (plaintext) difference and a difference to the output of the last but one round.
      ii. Generate N random plaintext-ciphertext pairs, such that each plaintext pair satisfies the input difference.
      iii. For each such pair, partially decrypt the ciphertext pairs by one-round.
      iv. Check if their xor is equal to the output difference or not...
      v. Let count denote the number of times the xor value is equal to the output difference out of N such pairs.
      vi. Repeat the above S = 100N many times to generate empirical distributions $count_1/N$, $count_2/N$, … , $count_S/N$. (*count_i/S)
      vii. Plot the empirical distribution along the y-axis and the values {0, 1, ... , N} along the x-axis and compare it (pictorially) with the normal distribution with mean $N/2^n$ and variance $N/2^n(1 - 1/2^n)$.

   It is advisable to use Python for this problem. (10 + 20 + 20 = 50)

1. Twofish Cipher – Link to code my_two.c

Code has been developed on Dev-C++ 5.11

The code is not true representation of twofish as in theory. Because round keys have been generated randomly and mds function is not complete.
All functions except round keys and mds(i.e., sBox, PHT, right/left rotation of subparts of intermediate ciphertexts) have been coded properly.

2. AES – Link to code AES.c

Code has been developed on Dev-C++ 5.11

Input: It takes 128-bit (16-byte) input in an array p as follows
int p[16]={'T','w','o',' ','O','n','e',' ','N','i','n','e',' ','T','w','o'};
It can also be represented as:

```
Printing plain(decimal)::
84 79 78 32
119 110 105 84
111 101 110 119
32 32 101 111
```

Output: It gives ciphertext in the form of bytes(decimal numbers)

```
Encrypted text::

41 87 64 26
195 20 34 2
80 32 153 215
95 246 179 58
```

Key: *int k[16]={'T','h','a','t','s',' ','m','y',' ','K','u','n','g',' ','F','u'};*

Examples can be verified using below files on my github page -
AES-example-matching.pdf
For immediate conversion of binary to hex- link
There are function for each operation as follows-
1.hex_to_bin()
2.bin_to_hex()
3. set_roundKey()
4 inv_set_roundKey()
5. add_roundKey()
6. sub_bytes()
7. in_sub_bytes()

8. shift_rows()
9. inverse_shift_rows()
10. mix_column()

Note: For inverse mix-column

3. Small PRESENT: Link to code -- smallPresent.py

Code has been developed on Spyder (Python 3.7).
It can be executed on python powershell using below command-
>>python smallPresent.py

a.Encryption-Decryption – line 117-159 in python file.
Master_key : all 0's
Plaintext : all 0's
Number of rounds: 10
Value of n: as given by user

Test vectors have been taken from the paper- from here.

Anaconda Powershell Prompt

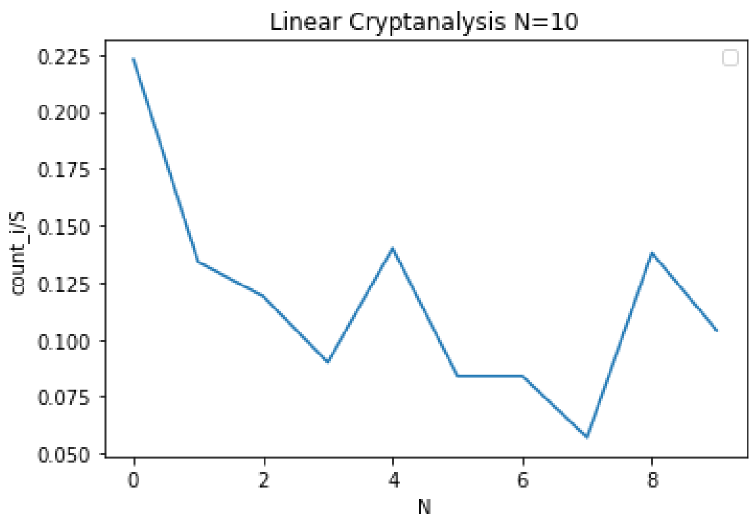
```
KeyboardInterrupt
(base) PS D:\Research 1.0\1 Applied Cryptography\Assignment-2> python smallPresent.py
Enter n: 2
['0', '0']
['f', '0']
['5', '9']
['5', '5']
['6', '2']
['b', '6']
['f', '7']
['0', '2']
['f', '8']
['8', '9']
['0', 'b']
Encrypted text
['0', 'b']
Decryption starts here----------
Deceypted text
['0', '0']
```

b.Linear Cryptanalysis – line **164-235** in python file under """3(b) Linear Cryptanalysis"""

```
Linear Cryptanalysis for number of rounds = 10
round:  0
round:  1
round:  2
round:  3
round:  4
round:  5
round:  6
round:  7
round:  8
round:  9
round:  10
round:  11
round:  12
```

```
round:  991
round:  992
round:  993
round:  994
round:  995
round:  996
round:  997
round:  998
round:  999
[244, 150, 135, 67, 152, 89, 87, 55, 123, 89
```

```
Mean of calculated distribution 0.039062500000000001
Actual Mean 0.0390625
```



Mean of calculated distribution 0.039062500000000014
Actual Mean 0.0390625

c.Differential Cryptanalysis- line **238-310** under """3(c) Differential Cryptanalysis"""
Mean of calculated distribution 0.02225
Actual Mean 0.078125

```
Differential Cryptanalysis for number of rounds 20
round:  0
round:  1
round:  2
round:  3
round:  4
round:  5
round:  6
round:  7
round:  8
round:  9
round:  10
round:  11
round:  12
round:  13
round:  14
```

```
round:  1991
round:  1992
round:  1993
round:  1994
round:  1995
round:  1996
round:  1997
round:  1998
round:  1999
[0, 56, 100, 55, 102, 13, 41, 23, 27, 68, 60, 36, 45, 21, 26, 10, 90, 55, 39, 23,
```

```
Mean of calculated distribution 0.022250000000000002
Actual Mean 0.078125
(base) PS D:\Research 1.0\1 Applied Cryptography\Assignment-2>
```