National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS-381: Network Security

Class: BESE-9A, BSCS-8A

Lab 04: Implementation of Data Encryption Standard (Part 1)

MUHAMMAD AMMAR
258341
BSCS 8C

Link to my Code Files

https://drive.google.com/drive/folders/1B1UR6b_EUF9q8vDwak2V6T5BcjRLJXF-?usp=sharing

Lab Tasks

Note: The input key and input plaintext is a hardcoded value. It is also possible to store key, plaintext, and ciphertext in separate '.txt' files, example in *D:/* drive and then read the files either for encryption or decryption.
KeyFile = D://KeyFile
MsgFile = D://MsgFile
CphFile = D://CphFile (should be empty at the time of encryption)

1. Write a method called **desPlaintextBlock** that takes the plaintext as input and splits it up into 64 bit blocks. If the last block is less than 64 bits, pad it with 0s.

[1 Marks]

Solution

```
def desPlaintextBlock():
    f=open("plainttext.txt",'r')
    str=f.read()
    x=''.join(format(ord(x), 'b') for x in str)
    arr_of_blocks=[]
    chunk_size =64
    arr_of_blocks = [x[i:i+chunk_size] for i in range(0, len(x), chunk_size)]
    last_block=arr_of_blocks[len(arr_of_blocks)-1]
    count=len(last_block)
    if (count <64):
        app=64-count
        for x in range(app):
            last_block='0'+last_block
    print(last_block)
    arr_of_blocks.pop()
    arr_of_blocks.append(last_block)
    # arr_of_blocks[len(arr_of_blocks)-1]=int(last_block)
    print(arr_of_blocks)
    return arr_of_blocks
desPlaintextBlock()
```

2. Write a method called **desInitialPermutation** that takes a 64-bit block, performs permutation according to the 8 x 8 IP table and returns the permuted block.

[1 Marks]

Solution

```
initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
                60, 52, 44, 36, 28, 20, 12, 4,
                62, 54, 46, 38, 30, 22, 14, 6,
                64, 56, 48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17, 9, 1,
                59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5,
                63, 55, 47, 39, 31, 23, 15, 7]
def desInitialPermutation (arr_of_blocks):
    arr_of_ip_block=''
    for i in range(len(arr_of_blocks)):
        x=initial_perm[i]-1
        y=arr_of_blocks[x]
        arr_of_ip_block+=y
    return arr_of_ip_block
arr_of_blocks=desPlaintextBlock()
print("Plain text 64 bits chunks Before Intial Permutations")
print (arr_of_blocks)
ip_blocks=[]
for x in arr_of_blocks:
    ip_blocks.append(desInitialPermutation (x))
print("Plain text 64 bits chunks after Intial Permutation:")
print(ip_blocks)
```

3. Write a method called **desFinalPermutation** that takes a 64-bit block, performs permutation according to the 8 x 8 FP table and returns the permuted block.

[1 Marks]

Solution

```
final_perm = [ 40, 8, 48, 16, 56, 24, 64, 32,
               39, 7, 47, 15, 55, 23, 63, 31,
               38, 6, 46, 14, 54, 22, 62, 30,
               37, 5, 45, 13, 53, 21, 61, 29,
               36, 4, 44, 12, 52, 20, 60, 28,
               35, 3, 43, 11, 51, 19, 59, 27,
               34, 2, 42, 10, 50, 18, 58, 26,
               33, 1, 41, 9, 49, 17, 57, 25 ]
def desFinalPermutation (ip_blocks):
    final_ip_block=''
    for i in range(len(ip_blocks)):
        x=final_perm[i]-1
        y=ip_blocks[x]
        final_ip_block+=y
    return final_ip_block
final_ip_block=[]
print('After Final Permutation')
for x in ip_blocks:
    final_ip_block.append(desFinalPermutation(x))
print(final_ip_block)
```

4. Write a method called **getSubkeys** that takes the secret key as input and returns the sub-keys for the 16 rounds of DES.

[6 Marks]

Solution

```
def hex2bin(s):
    mp = {'0' : "0000",
          '1' : "0001",
          '2' : "0010",
          '3' : "0011",
          '4' : "0100",
          '5' : "0101",
          '6' : "0110",
          '7' : "0111",
          '8' : "1000",
          '9' : "1001",
          'A' : "1010",
          'B' : "1011",
          'C' : "1100",
          'D' : "1101",
          'E' : "1110",
          'F' : "1111" }
    bin = ""
    for i in range(len(s)):
        bin = bin + mp[s[i]]
    return bin
def PC1(key):
    pc1key=''
    for i in range(len(keyp)):
        x=keyp[i]-1
        y=key[x]
        pc1key+=y
    return pc1key
def PC2(key):
    pc2key=''
    for i in range(len(pc2_table)):
        x=pc2_table[i]-1
        y=key[x]
        pc2key+=y
    return pc2key
# shifting the bits towards left by nth shifts
def shift_left(k, nth_shifts):
    s = ""
    for i in range(nth_shifts):
        for j in range(1,len(k)):
            s = s + k[j]
```

```python
            s = s + k[0]
            k = s
            s = ""
    return k
def getSubkeys ():
    f=open("secretkey.txt",'r')
    str=f.read()
    key=hex2bin(str)
    print("Secret Key: "+key)
    arr_of_subkeys=[]
    arr_of_PC1=(PC1(key))
    C=arr_of_PC1[0:28]
    D=arr_of_PC1[28:56]
    for i in range(16):
        # Shifting the bits by nth shifts by checking from shift table
        C = shift_left(C, shift_table[i])
        D = shift_left(D, shift_table[i])
        combine_subkey = C + D
        round_sub_key=PC2(combine_subkey)
        print("Round",i+1,"\t subkey:",round_sub_key)
    # print (arr_of_PC1)
    return arr_of_subkeys
getSubkeys()
```

Tasks Execution [1 Mark]

Screenshots

64-bit Chunks of Data

```
PS C:\Users\dell\Documents\Fall 2021\NS\NS lab 3>  c:; cd 'c:\Users\dell\Documents\Fall 2021\NS\NS lab 3'; & 'C:\Users\dell\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\dell\.vscode\extensions\ms-python.python-2021.10.1317843341\pythonFiles\lib\python\debugpy\launcher' '62850' '--' 'c:\Users\dell\Documents\Fall 2021\NS\NS lab 3\DES.py'
Plain text 64 bits chunks
['1001000110010111011001101100110111111000001010111110111111111001011', '0110011001001000000111010011010001101001111001110000011010011110 0', '111000000110000011000000100010011001011101101110111110000001110000 11', '0110011000011101001110111010000011101001100101111100011101001000', '00110011011011111110010100000100010010001011010011100000101000 10', '0010011000011110100110000110000010001011101110110001111100101111', '00111100001110100110100111011111101110100000101001111101001100 00', '00000000000000000000001110111011001001100001111001011001001 01001']
```

After Initial Permutation:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

PS C:\Users\dell\Documents\Fall 2021\NS\NS lab 3>  c:; cd 'c:\Users\dell\Documents\Fall 2021\NS\NS lab 3'; & 'C:\Users\dell\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\dell\.vscode\extensions\ms-python.python-2021.10.1317843341\pythonFiles\lib\python\debugpy\launcher' '63207' '--' 'c:\Users\dell\Documents\Fall 2021\NS\NS lab 3\DES.py'
______________
Plain Text:     Hello World this is a Demo plain text for DES (Data Encryption Standard)
_____________
Plain text 64 bits chunks Before Intial Permutations
['1001000110010111011001101100110111111000001010111110111111111001011', '0110011001001000000111010011010001101001111001110000011010011110 0', '111000000110000011000000100010011001011101101110111110000001110000 11', '0110011000011101001110111010000011101001100101111100011101001000', '00110011011011111110010100000100010010001011010011100000101000 10', '0010011000011110100110000110000010001011101110110001111100101111', '00111100001110100110100111011111101110100000101001111101001100 00', '00000000000000000000001110111011001001100001111001011001001 01001']
Plain text 64 bits chunks after Intial Permutation:
['1111110001110011011011101110101111011011000101001100100011100110', '0011101110010100111000010101000000110000100011011110111000110101', '111010110011000000101100111100101101001100110001000110001011001 0', '1101000100100110011000110111011001111000000111011001011001100101', '01010110001000001001011100000011111100100111001110001001010000011', '0000100001100110110000111111000000110100101010011111011011110011', '0100110011011011010010010100110000011000110101110111111100111010', '0100000001101000001100001100110000000100010011000111010000011110 0']
PS C:\Users\dell\Documents\Fall 2021\NS\NS lab 3>
```

After Final Permutation Without Key

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\dell\Documents\Fall 2021\NS\NS lab 3> & C:/Users/dell/AppData/Local/Programs/Python/Pyt
n37/python.exe "c:/Users/dell/Documents/Fall 2021/NS/NS lab 3/DES.py"
Plaintext:      Hello World DES lab
Before Intial Permutation
['1001000110010111011001101100110111110000010101111101111111001011', '0110011001001000001000100100
1101001110000011011001100000111000', '0000000000000000000000000000000000000000000000000000000000000000
']
After Intial Permutation
['1111110001110011011101110110101111011011000101001100100011100110', '0100101110000000011110010001
0000101000010010101110000100011010 1', '0000000000000000000000000000000000000000000000000000000000000001000
0000']
After Final Permutation
['1001000110010111011001101100110111110000010101111101111111001011', '0110011001001000001000100100
1011010011100000110110011000011100 0', '0000000000000000000000000000000000000000000000000000000000000000000
0010']
PS C:\Users\dell\Documents\Fall 2021\NS\NS lab 3>
```

16 unique subkeys for 16 rounds

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\dell\Documents\Fall 2021\NS\NS lab 3> & C:/Users/dell/AppData/Local/Programs/Pyth
on/Python37/python.exe "c:/Users/dell/Documents/Fall 2021/NS/NS lab 3/DES.py"
Secret Key: 1010101010111011000010010001100000100111001101101100110011011101
Round 1         subkey: 000110010100110011010000011100101101111010001100
Round 2         subkey: 010001010110100001011000000110101011110011001110
Round 3         subkey: 000001101110110110100100101011001111010110110101
Round 4         subkey: 110110100010110100000011001010110110111011100011
Round 5         subkey: 011010011010011000101001111111101100100100010011
Round 6         subkey: 110000011001010010001110100001110100011101011110
Round 7         subkey: 011100001000101011010010110111011011001111000000
Round 8         subkey: 001101001111100000100010111100001100011001101101
Round 9         subkey: 100001001011101101000100011100111101110011001100
Round 10        subkey: 000000100111011001010111000010001011010110111111
Round 11        subkey: 011011010101010110000010101111011111001010010 1
Round 12        subkey: 110000101100000111101001011010100100101111110011
Round 13        subkey: 100110011100001100010011100101111100100100011111
Round 14        subkey: 001001010001101110001011110001110001011111010000
Round 15        subkey: 001100110011000011000101110110011010001101101101
Round 16        subkey: 000110000001110001011101011101011100011001101101
PS C:\Users\dell\Documents\Fall 2021\NS\NS lab 3>
```

Grade Criteria

This lab is graded. Min marks: 0. Max marks: 10.

Activity	Minimum	Maximum
Documentation with clearly defined understanding of the lab task and approach	Fail	Pass
Lab Tasks	0	09
Screenshots	0	01