Assignment Answer

CSE436, Computer And Networks Security

Name: Hala Mohamed Shaheen ID: 1601655

Assignment No: (2)

Date: 3 / 2 /2021

DES Implementation functions:

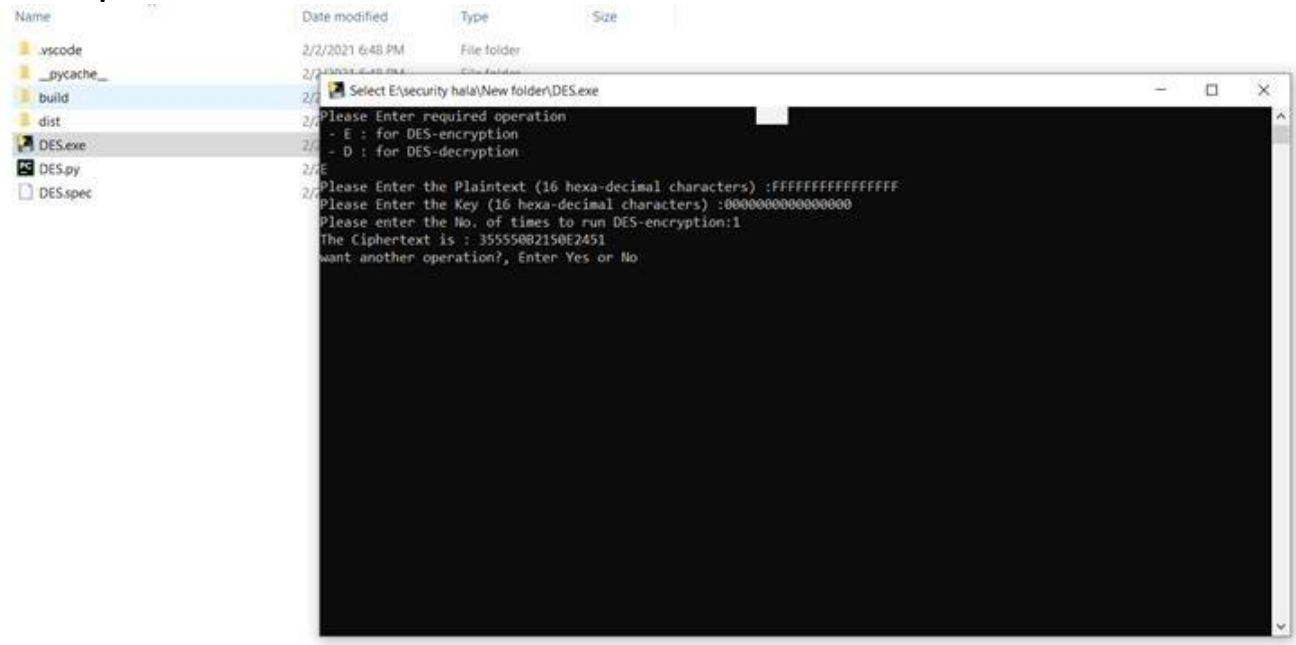
- **key_permutation1(key):** First stage of Key Generation : permuted-choice 1:
  input : 64-bit key , And *output* : corresponding 56-bit after applying permutation choice 1 as referenced from table pc1.
- **key_circular_shift(toshift,n):** Second stage of Key Generation : left circular shift:
  input : key that need to be left-shifted and round number specifying the no. of shifts.
  Output: shifted key same length as input
- **key_permutation2(keys):** Last stage of Key Generation : permuted-choice 2.
  input : 16 56-bit keys
  output: 16 48-bit keys after applying permutation choice 2 as referenced from table pc2.
- **Key_Generator(key):** Key Generation Function :
  input : 64-bit key
  output : 16 48-bit keys
  explanation : responsible of generation of 16 48-bit keys from input 64-bit key
- **HexaToBinary(s):** Helper FN :
  input : string of hexa-decimal number
  output : string of corresponding binary number
- **initial_permutation(data):** First stage of DES structure : initial-permutation
  input : 64-bit plain-text
  output: 64-bit initially permuted data (different arrangement of input)
- **E_Box_Operation(righ_part):** Expansion permutation
  input : 32-bit right part of initially permuted plain-text
  output : 48-bit after applying expansionas referenced from EBox table
- **xor(arg1,arg2):** we need to xor the right part with the key, also the left part with the output of the function F
- **SBox_Looping(sinput,x):** Helper Fn :
  input : 6-bit data and sbox number
  output : 4-bit data according to sbox
- **sbox(sboxin):**
  input : 48-bit data
  output : corresponding 32-bit data after applying substitution choice

- **F_permutation(topermute):** last stage in function F
  input : 32-bit data
  output : corresponding 32-bit data after applying permutation as referenced in
- **F(right,key):**
  input : 32-bit right part of initialy permutated plain-text
  output : corresponding 32-bit after applying F function
  explanation : this function is composed of several stages
    1- Expansion permutation
    2- XORing with round key
    3- Substitution choice
    4- F permutation
- **round(data,rkey):**
  input : 64-bit data
  output : corresponding 64-bit data
  explanation :
    1- 64-bit input is divided to 2 32-bit left and right parts
    2- F function is applied to right part
    3- new right : output of XORing output of F function with left part
    4- new left : old right part
- **Final_permutation(data):** inverse-permutation :
  input : 64-bit data after applying 16 DES rounds then swapping the two-halves
  output : 64-bit cipher-text
- **BinaryToHexa(s):** Helper FN :
  input : string of hexa-decimal number
  output : string of corresponding binary number
- **DES_Encryption(data,key16,no_of_encryption):**
  input : 64-bit plain-text , no_of times of encryption
  output : corresponding 64-bit cipher-text
  explanation :
    1- initial permutation of input 64-bit plain-text
    2- 16 DES rounds
    3- 32-bit swap of the output of 16th round
    4- inverse initial permutation of swap output which represents cipher-text
- **DES_Decryption(data,key16):**
  input : 64-bit cipher-text
  output : corresponding 64-bit plain-text
  explanation : similar to encryption with only 1 difference the round keys are reversed
- **main():**
  1. Asks the user to enter the operation (Encryption or Decryption)
  2. Asks the user to enter the plain text
  3. Asks the user to enter the Key
  4. Asks the user to enter how many times to run encryption if the operation is Encryption.

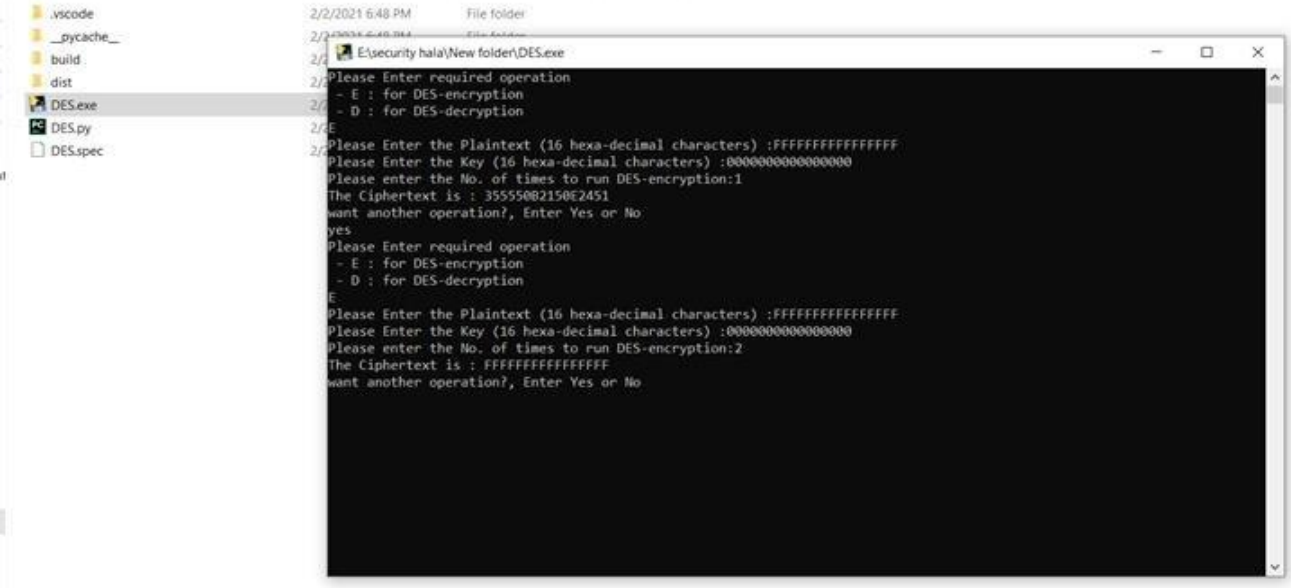
Examples:

1. Encryption

Input text : FFFFFFFFFFFFFFFF
Input Key : 0000000000000000
Repeat only 1 time
The output is : 355550B2150E2451



Input text : FFFFFFFFFFFFFFFF
Input Key : 0000000000000000
Repeat only 2 time
The output is : FFFFFFFFFFFFFFFF



2. Decryption

Input text : 355550B2150E2451
Input Key : 0000000000000000
The output is : FFFFFFFFFFFFFFFF