DES Assignment

CSE436, Computer and Networks Security

Name: Manal Ahmed Mahamed ID: 1601449

Date: 3 /2 /2021

This report consists of:

1) Code explanation

2) How to run the exe with screenshots provided for example

➢ Code explanation

1) Key Formatting Helper Functions:

There are 3 Helper functions for the 3 operations done on the key input.

1st we get the 64 bits key as an input from the user and use permutation choice 1 table to get only 56 bits as an output by using the helper function key_permutation1(key).

2nd we do the left shift using left shift round table, we take the key we want to shift and the number of the round to determine how many left shifts we do by using the helper function key_circular_shift(toshift,n).

3rd get the 16-56 bits keys as an input and use permutation choice 2 table to get only 16-48 bits keys as an output by using the helper function key_permutation2(keys).

2) Key Generator Function:

Key_Generator(key) →The function that generates the 16 keys, it takes the 64 bits key as input and outputs the 16-48 bits keys in a numpy array by using the helper functions above.

First it calls the key_permutation1(key) then it divides the 56 bits key to left and right parts each is 28 bits then it loops 16 times to create the 16 keys by calling the key_circular_shift(toshift,n) then after the loop finishes it uses the key_permutation2(keys) to get the 16-48 bits keys.

3) **Data Formatting Helper Functions:**

There is 11 helper function that are used.

HexaToBinary(s) → the first function that operates on the input data from the user, as we take the input in hexadecimal format and the operations are done on binary bits so we need to convert it first, it maps the hexadecimal to its corresponding binary bits

initial_permutation(data) → we use the initial permutation table to rearrange the input data, the input length (64bits) is the same as the output length its only rearrangement, we loop on the numbers in the IP table and take the bit from the data which correspond to that number.

Final_permutation(data) → we use the final permutation table to rearrange the output data, the input length (64bits) is the same as the output length its only rearrangement, we loop on the numbers in the IP inverse table and take the bit from the data which correspond to that number.

4) **Function F Helper Functions:**

E_Box_Operation(righ_part) → to do the expansion on the right part of the 32 bits right part of the data so we can XOR it with the 48 bits key, we use the expansion box table given.

xor(arg1,arg2) → we do need the xor for the function F and for the round itself.

SBox_Looping(sinput,x) & sbox(sboxin) → to perform the S box operation on the output of the xoring between the key and the right part so we git 32 bits again

F_permutation(topermute) → we use the function F permutation table to rearrange the output of the S box, the input length (32bits) is the same as the output length its only rearrangement.

5) **Function F:**

F(right,key) → 1st it calls the expansion helper function to be done on the right part 2nd it calls the Xor function to xor the output of the expansion with the key 3rd it calls the S box functions and lastly it does the function F permutation and returns its 32 bits output

6) **Round Funtion:**

round(data,rkey) → 1st it takes the input data and splits it into right and left parts each is 32bits long the new left part is equal to the pervious right part 2nd it calls the Function F function 3rd it calls the Xor function to be done on the output of the function and the left part and assign it to the new right part 4th it returns the new right and left parts

7) **DES Encryption:**

DES_Encryption(data,key16,no_of_encryption) → first we need to know the number of encryptions, the user determine it and we loop on that number to do the DES encryption, 1st we call IP function to be done on the data then for 16 times we call the round function and give it its correct key from the 16 key we have and finally do our FP (IP inverse) and output our encrypted block in string format.

8) **DES Decryption:**

DES_Decryption(data,key16) → it is the same as encryption but we only use the keys in reverse order

- **How to run the exe with screenshots provided for example**
- **Example1 (encryption with 1 run)**

  -First enter your key, for example: 0000000000000000

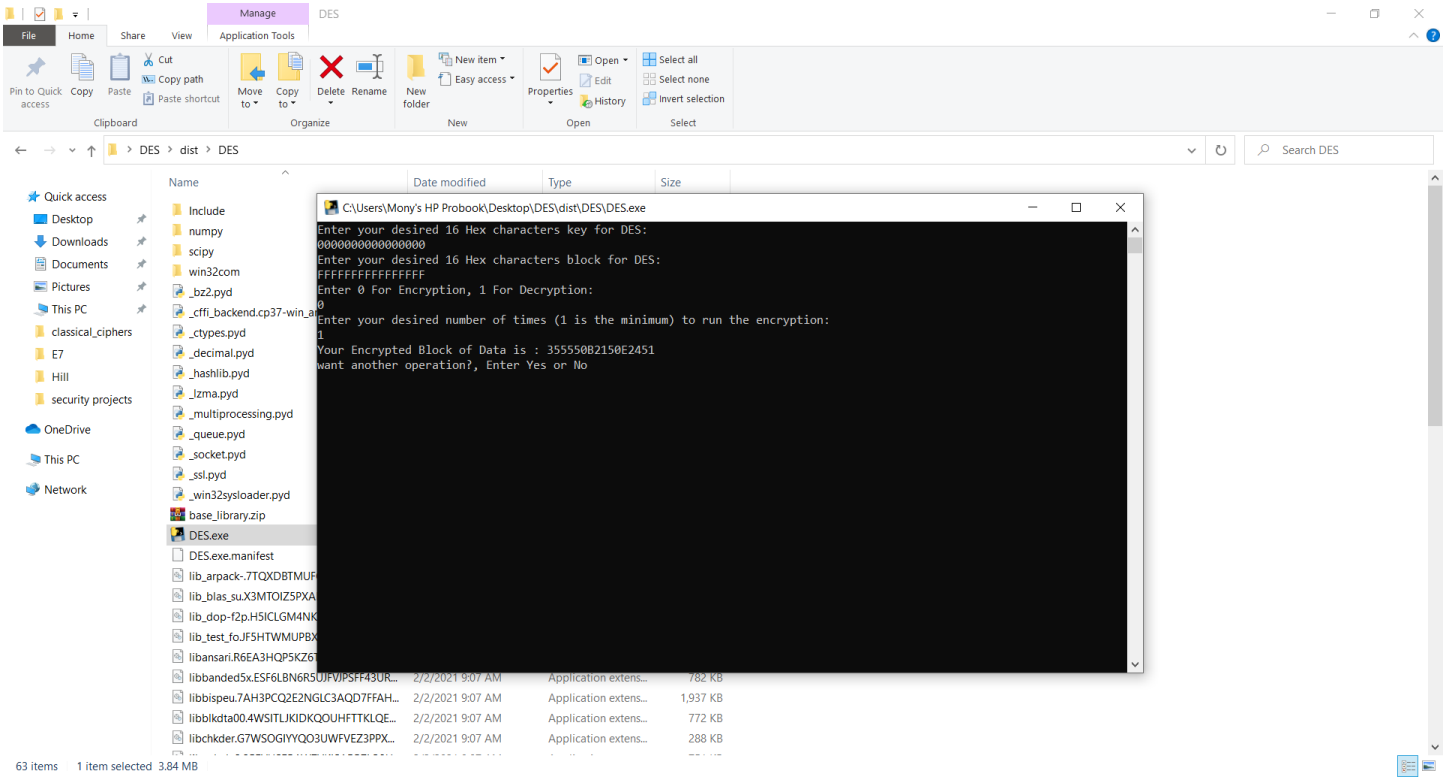
  -Then enter the block of data, for example: FFFFFFFFFFFFFFFF

  -Then choose whether you want encryption or decryption, for encryption enter 0 and for decryption enter 1, for example choose 0

  -enter the number of runs, for example 1.

  -The output will be printed for you as shown in the screenshot below:

Screenshot for encryption 1 run



➢ Example2(encryption with 2 runs)

-First enter yes to do another operation

-enter your key, for example: 0000000000000000

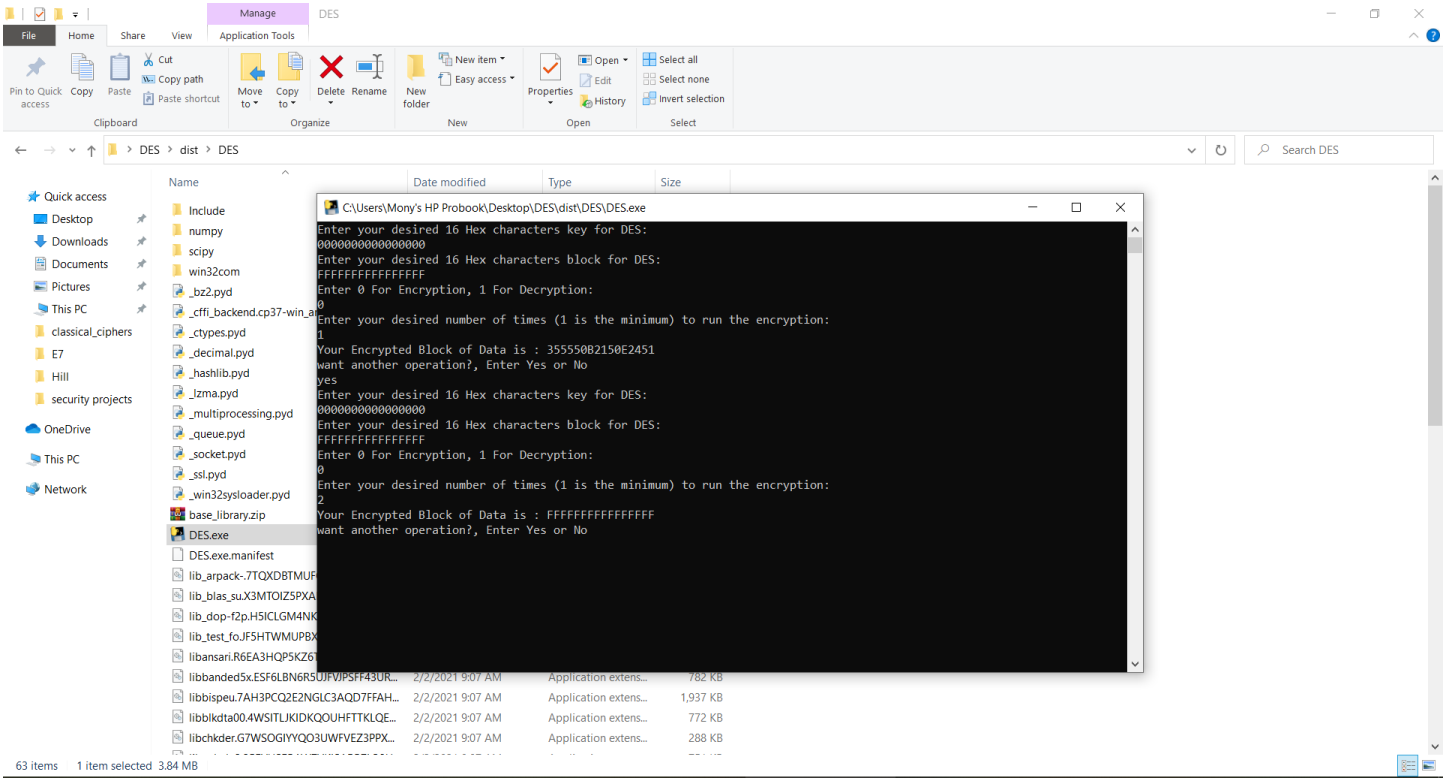
-Then enter the block of data, for example: FFFFFFFFFFFFFFFF

-Then choose whether you want encryption or decryption, for encryption enter 0 and for decryption enter 1, for example choose 0

-enter the number of runs, for example 2.

-The output will be printed for you as shown in the screenshot below:

Screenshot for encryption 2 runs



➢ Example3(decryption)

-First enter yes do to another operation

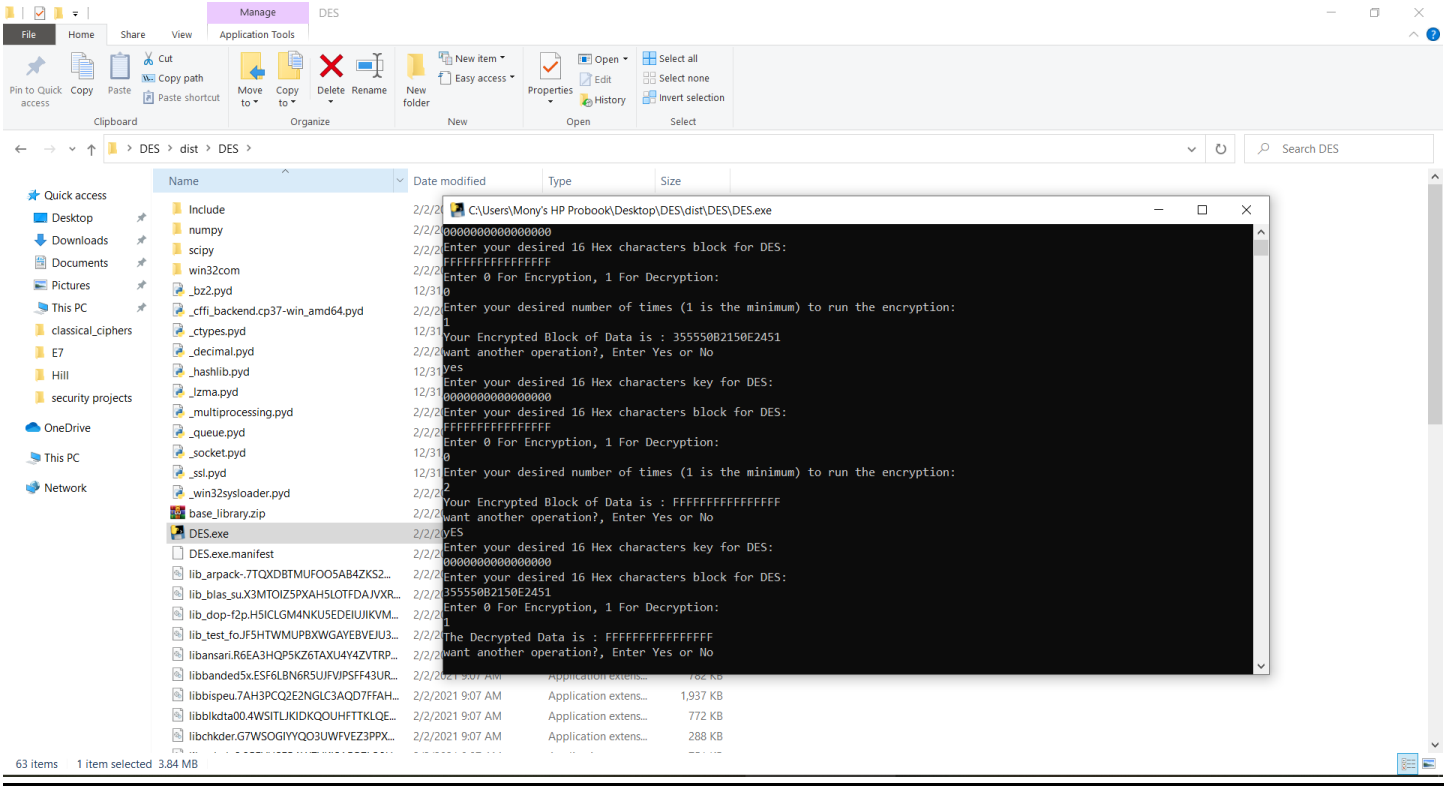
-enter your key, for example: 0000000000000000

-Then enter the block of data, for example: FFFFFFFFFFFFFFFF

-Then choose whether you want encryption or decryption, for encryption enter 0 and for decryption enter 1, for example choose 1

-The output will be printed for you as shown in the screenshot below:

Screenshot for decryption



Then type No and the exe will be closed.