Name: Sohel Momin

Roll No:1513126/a4

CSS EXPERIMENT 5

**Program:**

%% Data Encryption Standard

% Takes a string from user and encrypts it using DES with a random generated 64-bit key

% It displays encrypted string and decrypts it using DES with the same key

% D\_E\_S.m, permutation.m, sbox\_perm.m should be in the same folder !!!

clc;clear variables;close all;

msg = double(input('Enter message -> ','s'));

bits = 8; % the no. of bits used to represent a symbol. For representing all

% ASCII values we need minimum 7 bits, I chose 8 here for

% convenience

%% Appending spaces(ascii=32) to original msg to make bit stream perfectly divisble into 64-bit blocks

while mod(length(msg)\*bits,64) ~= 0

msg = horzcat(msg,32);

end

msg\_bit = zeros(1,length(msg)\*bits);

for i=1:length(msg)

msg\_bit((i-1)\*bits+1:i\*bits) = de2bi(msg(i),'left-msb',bits);

end

%% Random 64-bit key generated

key = round(rand(8,7));

key(:,8) = mod(sum(key,2),2); % parity column

key = reshape(key',1,64);

%% Encrypting message bits block by block ( of size 64 bits)

enc\_bit = zeros(1,length(msg\_bit));

for i=1:length(msg\_bit)/64

enc\_bit((i-1)\*64+1:i\*64) = D\_E\_S(msg\_bit((i-1)\*64+1:i\*64),key,'enc'); % calling DES for encryption

end

%% Displaying Encrypted message

enc\_msg = zeros(1,length(enc\_bit)/bits);

for i = 1:length(enc\_bit)/bits

enc\_msg(i) = bi2de(enc\_bit((i-1)\*bits+1:i\*bits),'left-msb');

end

fprintf('\nThe encrypted message is -> %s\n',char(enc\_msg));

%% Decrypting encrypted bits block by block ( of size 64 bits)

dec\_bit = zeros(1,length(enc\_bit));

for i=1:length(enc\_bit)/64

dec\_bit((i-1)\*64+1:i\*64) = D\_E\_S(enc\_bit((i-1)\*64+1:i\*64),key,'dec'); % calling DES for decryption

end

%% Displaying Encrypted message

dec\_msg = zeros(1,length(dec\_bit)/bits);

for i = 1:length(dec\_bit)/bits

dec\_msg(i) = bi2de(dec\_bit((i-1)\*bits+1:i\*bits),'left-msb');

end

fprintf('\nThe decrypted message is -> %s\n',char(dec\_msg));

**Functions defined:**

**1) D\_E\_S.m**

function [output64] = D\_E\_S(input64,key64,mode)

% The input,key should be of 64 bits !!!

% The mode should be either 'enc' or 'dec' for encryption/decryption

% respectively !!!

% permutation.m, sbox\_perm.m should be in the same folder !!!

%% DES Fixed Variables (taken from https://en.wikipedia.org/wiki/DES\_supplementary\_material )

initial\_perm\_key = [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];

final\_perm\_key = [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];

expansion\_p\_key = [32 1 2 3 4 5 ...

4 5 6 7 8 9 ...

8 9 10 11 12 13 ...

12 13 14 15 16 17 ...

16 17 18 19 20 21 ...

20 21 22 23 24 25 ...

24 25 26 27 28 29 ...

28 29 30 31 32 1];

straight\_p\_key =[16 7 20 21 29 12 28 17 ...

1 15 23 26 5 18 31 10 ...

2 8 24 14 32 27 3 9 ...

19 13 30 6 22 11 4 25];

PC1L\_key = [57 49 41 33 25 17 9 ...

1 58 50 42 34 26 18 ...

10 2 59 51 43 35 27 ...

19 11 3 60 52 44 36];

PC1R\_key = [63 55 47 39 31 23 15 ...

7 62 54 46 38 30 22 ...

14 6 61 53 45 37 29 ...

21 13 5 28 20 12 4];

PC2\_key = [14 17 11 24 1 5 3 28 ...

15 6 21 10 23 19 12 4 ...

26 8 16 7 27 20 13 2 ...

41 52 31 37 47 55 30 40 ...

51 45 33 48 44 49 39 56 ...

34 53 46 42 50 36 29 32];

%% Round key generation (for all 16 rounds,stored in 'round\_keys')

shift = [1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1];

round\_key = zeros(16,48);

% dropping parity bits, separating into left, right keys

l\_key28 = permutation(key64, PC1L\_key);

r\_key28 = permutation(key64, PC1R\_key);

% shifting left,right keys and compressing them to 48 bits

for i = 1:16

l\_key28 = [l\_key28(1+shift(i):end),l\_key28(1:shift(i))];

r\_key28 = [r\_key28(1+shift(i):end),r\_key28(1:shift(i))];

round\_key(i,:) = permutation([l\_key28,r\_key28],PC2\_key);

end

%% Initial permutation of the 64-bit input and splitting in left,right parts of 32 bits

input64 = permutation(input64, initial\_perm\_key);

if mode == 'enc'

lpt32 = input64(1:32);

rpt32 = input64(33:64);

elseif mode == 'dec'

lpt32 = input64(33:64);

rpt32 = input64(1:32);

else

fprintf('Enter correct mode - (enc/dec) \n');

return

end

%% Feistel/DES function (f)

for i=1:16

swap\_left = rpt32;

% EXPANSION P BOX

rpt48 = permutation(rpt32, expansion\_p\_key);

% KEY MIXING

rpt\_xor48 = xor(rpt48,round\_key(i,:));

if mode == 'dec'

rpt\_xor48 = xor(rpt48,round\_key(17-i,:)); % Applying keys in reverse for decryption

end

% SUBSTITUTION BOX

rpt\_sbox32 = sbox\_perm(rpt\_xor48);

% STRAIGHT P BOX

rpt32 = permutation(rpt\_sbox32, straight\_p\_key);

% MIXING AND SWAPPING

swap\_right = xor(rpt32,lpt32);

lpt32 = swap\_left;

rpt32 = swap\_right;

end

%% Combining left,right parts of 32 bits and Final permutation of the 64-bit output

output64 = [lpt32,rpt32];

if mode == 'dec'

output64 = [rpt32,lpt32]; % Swapping order for decryption

end

output64 = permutation(output64, final\_perm\_key);

end

**2) permutation.m**

function [output] = permutation(array,key)

% This function permutates the given array according to given key

output = zeros(size(key)); % size of key will be the size of output

for i=1:length(key)

output(i) = array(key(i));

end

end

**3) sbox\_perm.m**

function [output] = sbox\_perm(xored)

% xored input - 1 x 48

% output - 1 x 32

% This function gives corresponding sbox permutation of 32 bits for given

% 48 bits xored input(from previous DES step)

sbox{1} = [14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7;...

0 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8;...

4 1 14 8 13 6 2 11 15 12 9 7 3 10 5 0;...

15 12 8 2 4 9 1 7 5 11 3 14 10 0 6 13];

sbox{2} = [15 1 8 14 6 11 3 4 9 7 2 13 12 0 5 10;...

3 13 4 7 15 2 8 14 12 0 1 10 6 9 11 5 ;...

0 14 7 11 10 4 13 1 5 8 12 6 9 3 2 15;...

13 8 10 1 3 15 4 2 11 6 7 12 0 5 14 9];

sbox{3} = [10 0 9 14 6 3 15 5 1 13 12 7 11 4 2 8;...

13 7 0 9 3 4 6 10 2 8 5 14 12 11 15 1;...

13 6 4 9 8 15 3 0 11 1 2 12 5 10 14 7;...

1 10 13 0 6 9 8 7 4 15 14 3 11 5 2 12];

sbox{4} = [7 13 14 3 0 6 9 10 1 2 8 5 11 12 4 15;...

13 8 11 5 6 15 0 3 4 7 2 12 1 10 14 9;...

10 6 9 0 12 11 7 13 15 1 3 14 5 2 8 4;...

3 15 0 6 10 1 13 8 9 4 5 11 12 7 2 14];

sbox{5} = [2 12 4 1 7 10 11 6 8 5 3 15 13 0 14 9;...

14 11 2 12 4 7 13 1 5 0 15 10 3 9 8 6;...

4 2 1 11 10 13 7 8 15 9 12 5 6 3 0 14;...

11 8 12 7 1 14 2 13 6 15 0 9 10 4 5 3];

sbox{6} = [12 1 10 15 9 2 6 8 0 13 3 4 14 7 5 11;...

10 15 4 2 7 12 9 5 6 1 13 14 0 11 3 8;...

9 14 15 5 2 8 12 3 7 0 4 10 1 13 11 6;...

4 3 2 12 9 5 15 10 11 14 1 7 6 0 8 13];

sbox{7} = [4 11 2 14 15 0 8 13 3 12 9 7 5 10 6 1;...

13 0 11 7 4 9 1 10 14 3 5 12 2 15 8 6;...

1 4 11 13 12 3 7 14 10 15 6 8 0 5 9 2;...

6 11 13 8 1 4 10 7 9 5 0 15 14 2 3 12];

sbox{8} = [13 2 8 4 6 15 11 1 10 9 3 14 5 0 12 7;...

1 15 13 8 10 3 7 4 12 5 6 11 0 14 9 2;...

7 11 4 1 9 12 14 2 0 6 10 13 15 3 5 8;...

2 1 14 7 4 10 8 13 15 12 9 0 3 5 6 11];

output = zeros(1,32);

for i=1:8

start = (i-1)\*6 + 1;

finish = i\*6;

row = bi2de([xored(start),xored(finish)],'left-msb')+ 1;

column = bi2de(xored(start+1 : finish-1),'left-msb')+ 1;

value = de2bi(sbox{i}(row,column),'left-msb',4); % getting 4 bits from each box

output((i-1)\*4+1 : i\*4) = value;

end

end

**Output**

**Enter message ->** The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

**The encrypted message is ->** ¶Àë óæäT <­äe>>£ ÿÒÄ\*§¾s¥i~¦/ï¥³ roõÕÌ×½¿  
¤ËÆÂàÛ+G£A)â?FZW1Ò:¿¶¦KhOùV¶Ev¿þñ'\*Üü-ssÃ@¥ÓÿÕmB¢I­9vK÷íGg¿>ÏÑÈãÂ@

**The decrypted message is ->** The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).