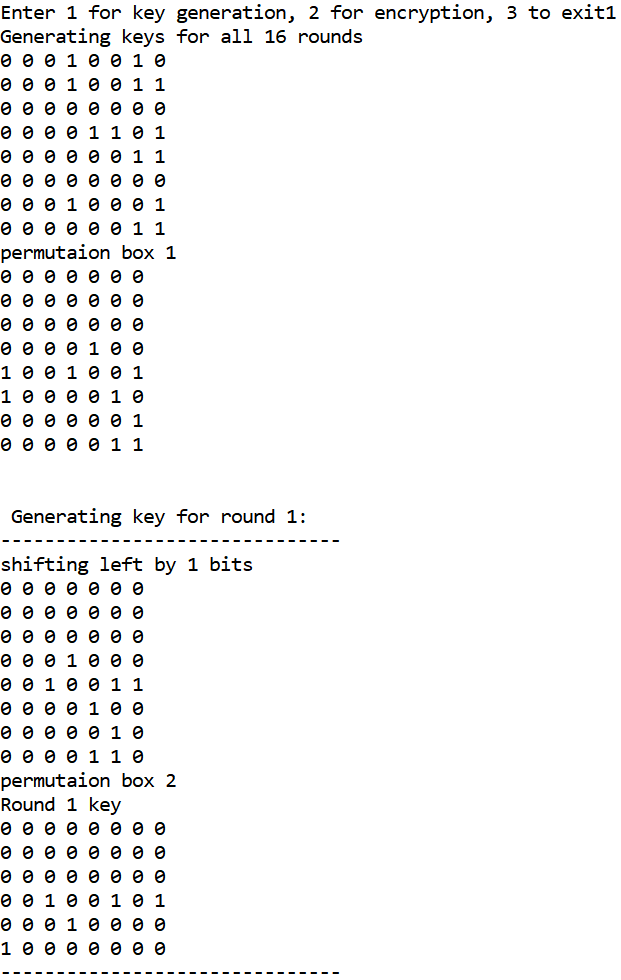
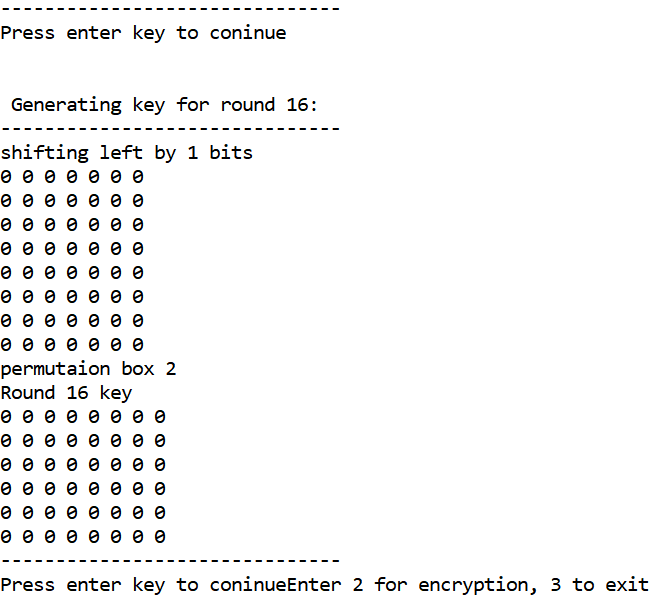
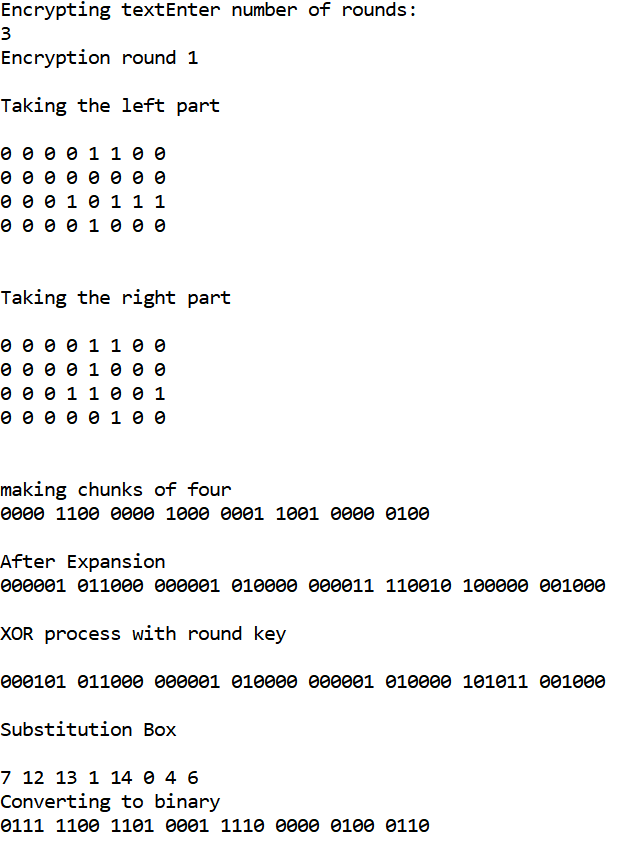
Plain text = MAXIMIZE

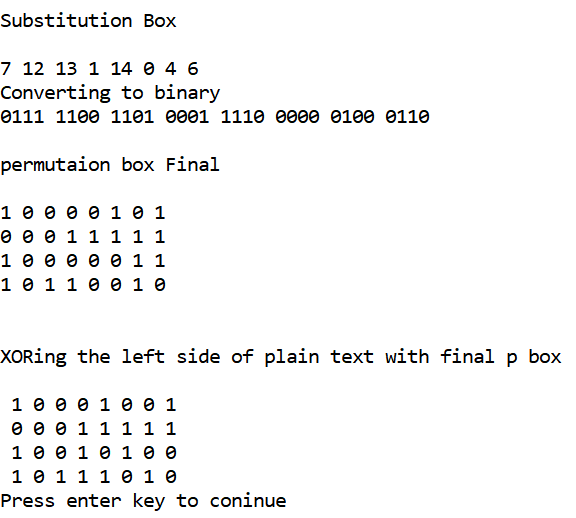
Key = STANDARD

# Working of system:









# Code:

#include <stdio.h>

char plain[8] = "MAXIMIZE";

char key[8] = "STANDARD";

char p1[64];

char shifted[56]; //holds the previous 56 rotated bits

char keys[16][48]; //to store 16 round keys of 48 length each

char bigarray[48]; //stores expanded bits

char xor[48]; //xor with round key

int sBox[8]; //sbox results

char sBoxBin[32];

int index=0;

int rhs[32];

int lhs[32];

char binaries[26][9] = {"00000000\0","00000001\0","00000010\0","00000011\0","00000100\0","00000101\0","00000110\0","00000111\0",

"00001000\0","00001001\0","00001010\0","00001011\0","00001100\0","00001101\0","00001110\0","00001111\0",

"00010000\0","00010001\0","00010010\0","00010011\0","00010100\0","00010101\0","00010110\0","00010111\0",

"00011000\0","00011001\0"};

char bin4[16][5] = {"0000\0","0001\0","0010\0","0011\0","0100\0","0101\0","0110\0","0111\0","1000\0","1001\0","1010\0",

"1011\0","1100\0","1101\0","1110\0","1111\0"};

int p1\_box[56] = {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,

63,55,47,39,21,23,15,

7,62,54,46,38,30,22,

14,6,61,53,45,37,29,

21,13,5,28,20,12,4};

int p2\_box[48] ={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};

int rotation[16] ={1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

int sboxes[8][4][16]= {

{

{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}

},

{

{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}

},

{

{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}

},

{

{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}

},

{

{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}

},

{

{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}

},

{

{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}

},

{

{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}

}

};

int Pbox[32] = {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};

**void generateKey(char permuted1[56], int roundNum)**{

printf("\n\n Generating key for round %d:",roundNum);

printf("\n-------------------------------\n");

int rot = rotation[roundNum-1]; //the bits to be shifted left

printf("shifting left by %d bits\n",rot);

for (int i=rot ; i<28 ; i++){

//skip first bit and shift bits left

shifted[i-rot] = permuted1[i];

}

if (rot==2){ // if 2 shifts left then append 1 zero more at each half end

shifted[26]='0';

shifted[54]='0';

}

shifted[27]='0'; //append 0 at end

for (int i=28+rot ; i<56 ; i++){

shifted[i-rot] = permuted1[i];

}

shifted[55]='0';

for (int i=0 ; i<56 ; i++){

printf("%c ",shifted[i]);

if ((i+1)%7==0)

printf("\n");

}

// permutation choice2 box

printf("permutaion box 2\nRound %d key\n",roundNum);

for (int i=0 ; i<48 ; i++){

keys[roundNum-1][i] = shifted[p2\_box[i]-1];

printf("%c ",keys[roundNum-1][i]);

if ((i+1)%8==0)

printf("\n");

}

printf("-------------------------------\n");

}

**void encryptText(int lhs[32], int rhs[32], int roundNum)**{

// -------------------- left and right part of the text ------------------------

printf("\n\nTaking the left part\n\n");

for (int i=0 ; i<32 ; i++){

printf ("%c ",lhs[i]);

if ((i+1)%8==0)

printf("\n");

}

printf("\n\nTaking the right part\n\n");

for (int i=0 ; i<32 ; i++){

printf ("%c ",rhs[i]);

if ((i+1)%8==0)

printf("\n");

}

//------------------------- expansion step---------------------------------------

printf("\n\nmaking chunks of four\n");

index=0;

int l=0;

for(int p=0;p<32;p++){

printf("%c",rhs[p]);

if((p+1)%4==0){

printf(" ");

}

}

int pos=1; //increments according to chunks

printf("\n\nAfter Expansion\n");

for (int i=4 ; i<32 ; i+=4){

bigarray[i+pos++]=rhs[i]; // copy from right chunk

bigarray[i+pos++]=rhs[i-1]; //copy from left chunk

}

//copy middle bits

pos=1;

for (int i=0 ; i<32 ; i++){

bigarray[pos++]=rhs[i];

if (pos==5 || pos==11 || pos==17 || pos==23 || pos==29 || pos==35 || pos==41){

pos+=2; //skip 2 middle bits

}

}

bigarray[0]=rhs[31];

bigarray[47]=rhs[0];

for(int p=0;p<48;p++){

printf("%c",bigarray[p]);

if(p==5||p==11||p==17||p==23||p==29||p==35||p==41||p==47){

printf(" ");

}

}

//------------------------- XOR with round key---------------------------------------

printf("\n\nXOR process with round key\n\n");

for(int i=0;i<48;i++){

if(bigarray[i]!=keys[roundNum][i])

xor[i]='1';

else xor[i]='0';

}

for(int i=0;i<48;i++){

printf("%c",xor[i]);

if(i==5||i==11||i==17||i==23||i==29||i==35||i==41||i==47){

printf(" ");

}

}

//------------------------- S-Box ---------------------------------------

printf("\n\nSubstitution Box\n\n");

int row=0,col=0,box\_index=0;

for (int i=0 ; i<48 ; i+=6){

row=checkrow(xor[i],xor[i+5]);

col=checkcol(xor[i+1],xor[i+2],xor[i+3],xor[i+4]);

sBox[box\_index]=sboxes[box\_index][row][col];

printf("%d ",sBox[box\_index]);

box\_index++;

}

printf("\nConverting to binary\n");

index=0;

for (int i=0 ; i<8 ; i++){

for (int j=0 ; j<4 ; j++){

// store the binary bits in 32 bit array

sBoxBin[index] = bin4[sBox[i]][j];

printf("%c",sBoxBin[index]);

index++;

}

printf(" ");

}

//------------------------- P-Box ---------------------------------------

printf("\n\npermutaion box Final\n \n");

char permutedFinal[32];

for (int i=0 ; i<32 ; i++){

permutedFinal[i] = sBoxBin[Pbox[i]-1];

printf("%c ",permutedFinal[i]);

if ((i+1)%8==0)

printf("\n");

}

//------------------------- XOR with LHS ---------------------------------------

// the xor-ed rhs becomes the new rhs and previous rhs becomes the new lhs

printf("\n\nXORing the left side of plain text with final p box\n\n");

char newRhs[32];

for(int i=0;i<32;i++){

if(permutedFinal[i]!=lhs[i])

newRhs[i]='1';

else newRhs[i]='0';

}

for(int i=0;i<32;i++){

printf(" %c",newRhs[i]);

if(i==7||i==15||i==23||i==31){

printf("\n");

}

}

//------------------------- New lhs and rhs ---------------------------------------

for (int i=0 ; i<32 ; i++){

lhs[i] = rhs[i];

}

for (int i=0 ; i<32 ; i++){

rhs[i] = newRhs[i];

}

}

**void main(){**

printf("Enter 1 for key generation, 2 for encryption, 3 to exit");

int option = 0;

scanf("%d",&option);

getchar();

do{

if (option==1){

printf("Generating keys for all 16 rounds\n");

int index = 0;

// for each character in the key text

for (int i=0 ; i<sizeof(key) ; i++){

// get the decimal value for the character

int decimal = key[i]-65;

// store the binaries in a single array

for (int j=0 ; j<8 ; j++){

p1[index] = binaries[decimal][j];

printf("%c ",p1[index]);

// index to hold the location of 64 bit string

index++;

}

printf("\n");

}

printf("permutaion box 1\n");

char permuted1[56];

for (int i=0 ; i<56 ; i++){

//printf("%d ",p1\_box[i-1]);

permuted1[i] = p1[p1\_box[i]-1];

//printf("%d",i);

printf("%c ",permuted1[i]);

if ((i+1)%7==0)

printf("\n");

}

// pass the 56-bit permuted key

generateKey(permuted1, 1);

printf("Press enter key to coninue");

for (int i=2 ; i<17 ; i++){

if (getchar()=='\n'){

generateKey(shifted,i);

printf("Press enter key to coninue");

}

}

printf("\nEnter 2 for encryption, 3 to exit");

scanf("%d",&option);

}

else if (option==2){

printf("Encrypting text\n");

for (int i=0 ; i<sizeof(plain)/2 ; i++){

// get the decimal value for the character

int decimal = plain[i]-65;

// store the binaries in a single array

for (int j=0 ; j<8 ; j++){

lhs[index] = binaries[decimal][j];

// index to hold the location of 64 bit string

index++;

}

}

index=0;

for (int i=(sizeof(plain)/2); i<sizeof(plain) ; i++){

// get the decimal value for the character

int decimal = plain[i]-65;

// store the binaries in a single array

for (int j=0 ; j<8 ; j++){

rhs[index] = binaries[decimal][j];

// index to hold the location of 64 bit string

index++;

}

}

printf("Enter number of rounds:\n");

int round;

scanf("%d",&round);

getchar();

printf("Encryption round %d",1);

encryptText(lhs,rhs,1);

printf("Press enter key to coninue");

for (int i=2 ; i<=round ; i++){

if (getchar()=='\n'){

printf("Encryption round %d",i);

encryptText(lhs,rhs,i);

printf("Press enter key to coninue");

}

}

printf("\nEnter 3 to exit");

scanf("%d",&option);

}

}while(option!=3);

}

**int checkrow(char a,char b){**

int d1 = a-'0'; //convert from char to digit

int d2 = b-'0';

return (2\*d1)+(d2);

}

**int checkcol(char a,char b,char c,char d){**

int a1 = a-'0';

int b1 = b-'0';

int c1 = c-'0';

int d1 = d-'0';

return (8\*a1)+(4\*b1)+(2\*c1)+(1\*d1);

}

DATA SECUTITY AND ENCRRYPTION

ASSIGNMENT-4

Submitted to:

Dr. Adnan Ahmed

Submitted by:

Ume-Kalsoom DDP-FA14-BSE-017

Samurah Shafique DDP-FA14-BSE-092