

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
DES
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<malloc.h>
#include<stdlib.h>
#include<math.h>
void hex_to_bin(char *,char *);
char* bin_to_hex(char *);
void permutation(char *,char *);
void make_half(char *,char *,char *);
void single_shift(char *,char *);
void double_shift(char *,char *);
void make_key(char *,char *,char *);
void permutation_32(char *,char *);
void permutation_48(char *,char *);
void permutation_64(char *,char *,char *);
void des_round(char *,char *,char *,char *,char
*,char *,char *);
void des_round_decry(char *,char *,char *,char
*,char *,char *,char *);
void copy(char *,char *);
void permut_48(char *,char *);
void xor(char *,char *,char *);
void xor_32(char *,char *,char *);
void common_permutation(char *,char *);
void hex_to_plain(char *,char *,int);
int switch_case(char );
char SB[32];
char *bin[]={
    "0000",
    "0001",
    "0010",
    "0011",
    "0100",
    "0101",
    "0110",
    "0111",
    "1000",
    "1001",
    "1010",
    "1011",
    "1100",
    "1101",
    "1110",
    "1111"
};
char
hex[16]={'0','1','2','3','4','5','6','7','8','9'
,'A','B','C','D','E','F'};
int PC1[8][7]={
    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,
    63,55,47,39,31,23,15,
    7,62,54,46,38,30,22,
    14,6,61,53,45,37,29,
    21,13,5,28,20,12,4
};
int PC2[8][6]={
    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 IP[8][8]={
    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};
int E_bit[8][6]={
    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};
char *look_up[]={
    "00",
    "01",
    "10",
    "11"};
int sb_permutation[8][4]={
    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};
int s1[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};
int s2[4][16]={
    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};
int s3[4][16]={
    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};
int s4[4][16]={
    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};
int s5[4][16]={
    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};
int s6[4][16]={
    12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
    10,15,4,2,7,12,9,5,6,1,12,14,0,11,3,8,
    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};
int s7[4][16]={
    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,
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12};
int s8[4][16]={
    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 ip_inverse[8][8]={
    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
};
void main()
{
    char input[200],initial_hex[400];
    int i,j,k=0,len,r,x,m,temp;
    int d,e,f;
    char hex_arr[25][16];
    char input_hex[16],input_bin[64];
    char
key_hex[16]='1','3','3','4','5','7','7','9','9'
,'B','B','C','D','F','F','1';
    char key_bin[64],key_PC1[56];
    char
ch,*decryption,*encryption,encryption_final[400]
,decryption_final_hex[400],decryption_final_plai
n[200];
    char
encrypted[64],decrypted[64],encry_permut[64],dec
ry_permut[64];
    int length,p=-1,q=-1;
    char C0[28],D0[28],
        C1[28],D1[28],CD1[56],
        C2[28],D2[28],CD2[56],
        C3[28],D3[28],CD3[56],
        C4[28],D4[28],CD4[56],
        C5[28],D5[28],CD5[56],
        C6[28],D6[28],CD6[56],
        C7[28],D7[28],CD7[56],
        C8[28],D8[28],CD8[56],
        C9[28],D9[28],CD9[56],
        C10[28],D10[28],CD10[56],
        C11[28],D11[28],CD11[56],
        C12[28],D12[28],CD12[56],
        C13[28],D13[28],CD13[56],
        C14[28],D14[28],CD14[56],
        C15[28],D15[28],CD15[56],
        C16[28],D16[28],CD16[56];
    char L0[32],R0[32],ER0[48];
    char K1[48],L1[32],R1[32],ER1[48],F1[48],
        K2[48],L2[32],R2[32],ER2[48],F2[48],
        K3[48],L3[32],R3[32],ER3[48],F3[48],
        K4[48],L4[32],R4[32],ER4[48],F4[48],
        K5[48],L5[32],R5[32],ER5[48],F5[48],
        K6[48],L6[32],R6[32],ER6[48],F6[48],
        K7[48],L7[32],R7[32],ER7[48],F7[48],
        K8[48],L8[32],R8[32],ER8[48],F8[48],
        K9[48],L9[32],R9[32],ER9[48],F9[48],
        K10[48],L10[32],R10[32],ER10[48],F10[48],
        K11[48],L11[32],R11[32],ER11[48],F11[48],
        K12[48],L12[32],R12[32],ER12[48],F12[48],
        K13[48],L13[32],R13[32],ER13[48],F13[48],
        K14[48],L14[32],R14[32],ER14[48],F14[48],
        K15[48],L15[32],R15[32],ER15[48],F15[48],
        K16[48],L16[32],R16[32],ER16[48],F16[48];
    printf(">Enter plain text : ");
    gets(input);
    len=strlen(input);
    for(i=0;i<len;i++)
    {
        while(input[i]!=0)
        {
            r=input[i]%16;
            input[i]=input[i]/16;
            if(r>9)
            {
                x=r-10;
                r=65+x;
                initial_hex[k]=r;
            }
            else
                initial_hex[k]=r+48;
            k++;
        }
    }
    for(i=0;i<k;i=i+2)
    {
        temp=initial_hex[i];
        initial_hex[i]=initial_hex[i+1];
        initial_hex[i+1]=temp;
    }
    d=k/16;
    e=k%16;
    f=0;
    for(i=0;i<=d;i++)
    {
        if(i<d)
        {
            for(j=0;j<=15;j++)
                hex_arr[i][j]=initial_hex[f++];
        }
        else if(k%16==0)
            break;
        else
        {
            for(j=0;j<=15;j++)
            {
                if(j<e)
                    hex_arr[i][j]=initial_hex[f++];
                else
                {
                    hex_arr[i][j]='2';
                    hex_arr[i][++j]='0';
                }
            }
        }
    }
    if(k%16!=0)
        d++;
    hex_to_bin(key_hex,key_bin);
    printf("\n>Key in Hexadecimal used for
encryption : ");
    for(i=0;i<16;i++)
        printf("%c",key_hex[i]);
        for(m=0;m<d;m++)
        {
            for(i=0;i<16;i++)
                input_hex[i]=hex_arr[m][i];
            hex_to_bin(input_hex,input_bin);
            permutation(key_bin,key_PC1);
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
make_half(key_PC1,C0,D0);
single_shift(C0,C1);
single_shift(D0,D1);
single_shift(C1,C2);
single_shift(D1,D2);
double_shift(C2,C3);
double_shift(D2,D3);
double_shift(C3,C4);
double_shift(D3,D4);
double_shift(C4,C5);
double_shift(D4,D5);
double_shift(C5,C6);
double_shift(D5,D6);
double_shift(C6,C7);
double_shift(D6,D7);
double_shift(C7,C8);
double_shift(D7,D8);
single_shift(C8,C9);
single_shift(D8,D9);
double_shift(C9,C10);
double_shift(D9,D10);
double_shift(C10,C11);
double_shift(D10,D11);
double_shift(C11,C12);
double_shift(D11,D12);
double_shift(C12,C13);
double_shift(D12,D13);
double_shift(C13,C14);
double_shift(D13,D14);
double_shift(C14,C15);
double_shift(D14,D15);
single_shift(C15,C16);
single_shift(D15,D16);
make_key(C1,D1,CD1);
permutation_48(CD1,K1);
make_key(C2,D2,CD2);
permutation_48(CD2,K2);
make_key(C3,D3,CD3);
permutation_48(CD3,K3);
make_key(C4,D4,CD4);
permutation_48(CD4,K4);
make_key(C5,D5,CD5);
permutation_48(CD5,K5);
make_key(C6,D6,CD6);
permutation_48(CD6,K6);
make_key(C7,D7,CD7);
permutation_48(CD7,K7);
make_key(C8,D8,CD8);
permutation_48(CD8,K8);
make_key(C9,D9,CD9);
permutation_48(CD9,K9);
make_key(C10,D10,CD10);
permutation_48(CD10,K10);
make_key(C11,D11,CD11);
permutation_48(CD11,K11);
make_key(C12,D12,CD12);
permutation_48(CD12,K12);
make_key(C13,D13,CD13);
permutation_48(CD13,K13);
make_key(C14,D14,CD14);
permutation_48(CD14,K14);
make_key(C15,D15,CD15);
permutation_48(CD15,K15);
make_key(C16,D16,CD16);
permutation_48(CD16,K16);
permutation_64(input_bin,L0,R0);
des_round(L1,R1,L0,R0,ER0,K1,F1);
des_round(L2,R2,L1,R1,ER1,K2,F2);
des_round(L3,R3,L2,R2,ER2,K3,F3);
des_round(L4,R4,L3,R3,ER3,K4,F4);
des_round(L5,R5,L4,R4,ER4,K5,F5);
des_round(L6,R6,L5,R5,ER5,K6,F6);
des_round(L7,R7,L6,R6,ER6,K7,F7);
des_round(L8,R8,L7,R7,ER7,K8,F8);
des_round(L9,R9,L8,R8,ER8,K9,F9);
des_round(L10,R10,L9,R9,ER9,K10,F10);
des_round(L11,R11,L10,R10,ER10,K11,F11);
des_round(L12,R12,L11,R11,ER11,K12,F12);
des_round(L13,R13,L12,R12,ER12,K13,F13);
des_round(L14,R14,L13,R13,ER13,K14,F14);
des_round(L15,R15,L14,R14,ER14,K15,F15);
des_round(L16,R16,L15,R15,ER15,K16,F16);
for(i=0;i<32;i++)
{
    encrypted[i]=R16[i];
    encrypted[i+32]=L16[i];
}
common_permutation(encrypted,encry_permut);
encryption=bin_to_hex(encry_permut);
for(i=0;i<16;i++)
{
    encryption_final[++p]=*(encryption+i);
}
des_round_decry(L16,R16,L15,R15,ER15,K16,F16);
des_round_decry(L15,R15,L14,R14,ER14,K15,F15);
des_round_decry(L14,R14,L13,R13,ER13,K14,F14);
des_round_decry(L13,R13,L12,R12,ER12,K13,F13);
des_round_decry(L12,R12,L11,R11,ER11,K12,F12);
des_round_decry(L11,R11,L10,R10,ER10,K11,F11);
des_round_decry(L10,R10,L9,R9,ER9,K10,F10);
des_round_decry(L9,R9,L8,R8,ER8,K9,F9);
des_round_decry(L8,R8,L7,R7,ER7,K8,F8);
des_round_decry(L7,R7,L6,R6,ER6,K7,F7);
des_round_decry(L6,R6,L5,R5,ER5,K6,F6);
des_round_decry(L5,R5,L4,R4,ER4,K5,F5);
des_round_decry(L4,R4,L3,R3,ER3,K4,F4);
des_round_decry(L3,R3,L2,R2,ER2,K3,F3);
des_round_decry(L2,R2,L1,R1,ER1,K2,F2);
des_round_decry(L1,R1,L0,R0,ER0,K1,F1);
for(i=0;i<32;i++)
{
    decrypted[i]=L0[i];
    decrypted[i+32]=R0[i];
}
common_permutation(decrypted,decry_permut);
decryption=bin_to_hex(decry_permut);
for(i=0;i<16;i++)
{
    decryption_final_hex[++q]=*(decryption+i);
}
encryption_final[p+1]='\0';
printf("\n\nEncrypted Output : ");
printf("%s",encryption_final);
decryption_final_hex[q+1]='\0';
printf("\n\nDecrypted Output in Hexadecimal:
");
printf("%s",decryption_final_hex);
hex_to_plain(decryption_final_hex,decryption_fin
al_plain,q+1);
printf("\n\nDecrypted Output in Plain Text: ");
printf("%s\n",decryption_final_plain);
getch();
}
void hex_to_bin(char *input,char *in)
{
    short i,j,k,lim=0;
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
for(i=0;i<16;i++)
{
    for(j=0;j<16;j++)
    {
        if(*(input+i)==hex[j])
        {
            for(k=0;k<4;k++)
            {
                *(in+lim)=bin[j][k];
                lim++;
            }
        }
    }
}
char* bin_to_hex(char *bit)
{
    char tmp[5],*out;
    short lim=0,i,j;
    out=(char*)malloc(16*sizeof(char));
    for(i=0;i<64;i=i+4)
    {
        tmp[0]=bit[i];
        tmp[1]=bit[i+1];
        tmp[2]=bit[i+2];
        tmp[3]=bit[i+3];
        tmp[4]='\0';
        for(j=0;j<16;j++)
        {
            if((strcmp(tmp,bin[j]))==0)
            {
                out[lim++]=hex[j];
                break;
            }
        }
    }
    out[lim]='\0';
    return out;
}
void hex_to_plain(char *in,char *out,int t)
{
    int i,j=0,z,sum;
    char temp[3];
    for(i=0;i<t;i=i+2)
    {
        sum=0;
        temp[0]=in[i];
        if(temp[0]>=65 && temp[0]<=71)
            z=switch_case(temp[0]);
        else
            z=temp[0]-48;
        sum=sum+z*16;
        temp[1]=in[i+1];
        if(temp[1]>=65 && temp[1]<=71)
            z=switch_case(temp[1]);
        else
            z=temp[1]-48;
        sum=sum+z*1;
        temp[2]='\0';
        *(out+j)=sum;
        j++;
    }
    *(out+j]='\0';
}
int switch_case(char a)
{
    switch(a)
    {
        case 'A':
            return(10);
            break;
        case 'B':
            return(11);
            break;
        case 'C':
            return(12);
            break;
        case 'D':
            return(13);
            break;
        case 'E':
            return(14);
            break;
        case 'F':
            return(15);
            break;
    }
}
void permutation(char *key_bin,char *key_PC1)
{
    short i,j,k=0,temp;
    for(i=0;i<8;i++)
    {
        for(j=0;j<7;j++)
        {
            temp=PC1[i][j]-1;
            *(key_PC1+k)=*(key_bin+temp);
            k++;
        }
    }
}
void make_half(char *key_PC1,char *a,char *b)
{
    int i,j=0;
    for(i=0;i<56;i++)
    {
        if(i<28)
            *(a+i)=*(key_PC1+i);
        else
        {
            *(b+j)=*(key_PC1+i);
            j++;
        }
    }
}
void single_shift(char *p,char *q)
{
    int i;
    *(q+27)=*(p+0);
    for(i=0;i<27;i++)
        *(q+i)=*(p+(i+1));
}
void double_shift(char *p,char *q)
{
    int i;
    *(q+26)=*(p+0);
    *(q+27)=*(p+1);
    for(i=0;i<26;i++)
        *(q+i)=*(p+(i+2));
}
void make_key(char *a,char *b,char *c)
{
    int i;
    for(i=0;i<28;i++)
        *(c+i)=*(a+i);
}
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
for(i=28;i<56;i++)
    *(c+i)=*(b+(i-28));
}
void permutation_48(char *CD,char *K)
{
    short i,j,m=0,temp;
    for(i=0;i<8;i++)
    {
        for(j=0;j<6;j++)
        {
            temp=PC2[i][j]-1;
            *(K+m)=*(CD+temp);
            m++;
        }
    }
}
void permutation_64(char *in,char *L,char *R)
{
    int i,j,m=0,temp;
    for(i=0;i<4;i++)
    {
        for(j=0;j<8;j++)
        {
            temp=IP[i][j]-1;
            *(L+m)=*(in+temp);
            m++;
        }
    }
    m=0;
    for(i=4;i<8;i++)
    {
        for(j=0;j<8;j++)
        {
            temp=IP[i][j]-1;
            *(R+m)=*(in+temp);
            m++;
        }
    }
}
void des_round(char *L1,char *R1,char *L0,char
*R0,char *ER0,char *K1,char *F1)
{
    char t[3],tp[5],f[32];
    int temp,i,row,column,j,limit=0;
    copy(L1,R0);
    permut_48(R0,ER0);
    xor(K1,ER0,F1);
    for(i=0;i<48;i=i+6)
    {
        t[0]=F1[i];
        t[1]=F1[i+5];
        t[2]='\0';
        for(j=0;j<4;j++)
        {
            if(strcmp(t,look_up[j])==0)
            {
                row=j;
                break;
            }
        }
        tp[0]=F1[i+1];
        tp[1]=F1[i+2];
        tp[2]=F1[i+3];
        tp[3]=F1[i+4];
        tp[4]='\0';
        for(j=0;j<16;j++)
        {
            if(strcmp(tp,bin[j])==0)
            {
                column=j;
                break;
            }
        }
        switch(i)
        {
            case 0:
                temp=s1[row][column];
                break;
            case 6:
                temp=s2[row][column];
                break;
            case 12:
                temp=s3[row][column];
                break;
            case 18:
                temp=s4[row][column];
                break;
            case 24:
                temp=s5[row][column];
                break;
            case 30:
                temp=s6[row][column];
                break;
            case 36:
                temp=s7[row][column];
                break;
            case 42:
                temp=s8[row][column];
                break;
        }
        for(j=0;j<4;j++)
        {
            SB[limit]=bin[temp][j];
            limit++;
        }
        SB[limit]='\0';
        permutation_32(SB,f);
        SB[0]='\0';
        xor_32(L0,f,R1);
    }
}
void des_round_decry(char *L1,char *R1,char
*L0,char *R0,char *ER0,char *K1,char *F1)
{
    char tp[5],f[32];
    short temp,i,row,column,j,limit=0;
    copy(L1,R0);
    permut_48(R0,ER0);
    xor(K1,ER0,F1);
    for(i=0;i<48;i=i+6)
    {
        tp[0]=F1[i];
        tp[1]=F1[i+5];
        tp[2]='\0';
        for(j=0;j<4;j++)
        {
            if(strcmp(tp,look_up[j])==0)
            {
                row=j;
                break;
            }
        }
        tp[0]=F1[i+1];
        tp[1]=F1[i+2];
        tp[2]=F1[i+3];
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
tp[3]=F1[i+4];
tp[4]='\0';
for(j=0;j<16;j++)
{
    if(strcmp(tp,bin[j])==0)
    {
        column=j;
        break;
    }
}
switch(i)
{
    case 0:
        temp=s1[row][column];
        break;
    case 6:
        temp=s2[row][column];
        break;
    case 12:
        temp=s3[row][column];
        break;
    case 18:
        temp=s4[row][column];
        break;
    case 24:
        temp=s5[row][column];
        break;
    case 30:
        temp=s6[row][column];
        break;
    case 36:
        temp=s7[row][column];
        break;
    case 42:
        temp=s8[row][column];
        break;
}
for(j=0;j<4;j++)
{
    SB[limit]=bin[temp][j];
    limit++;
}
SB[limit]='\0';
permutation_32(SB,f);
SB[0]='\0';
xor_32(L0,f,R1);
}
void copy(char *L,char *R)
{
    int i;
    for(i=0;i<32;i++)
        *(L+i)=*(R+i);
}
void permut_48(char *R,char *ER)
{
    short i,j,m=0,temp;
    for(i=0;i<8;i++)
    {
        for(j=0;j<6;j++)
        {
            temp=E_bit[i][j]-1;
            *(ER+m)=*(R+temp);
            m++;
        }
    }
}
void xor(char *K,char *ER,char *F)
{
    int i,m=0;
    for(i=0;i<48;i++)
    {
        if((*(K+i)=='1' && *(ER+i)=='1') ||
        (*(K+i)=='0' && *(ER+i)=='0'))
        {
            *(F+m)='0';
            m++;
        }
        else
        {
            *(F+m)='1';
            m++;
        }
    }
}
void xor_32(char *L0,char *f,char *R1)
{
    short i,m=0;
    for(i=0;i<32;i++)
    {
        if((*(L0+i)=='1' && *(f+i)=='1') ||
        (*(L0+i)=='0' && *(f+i)=='0'))
        {
            *(R1+m)='0';
            m++;
        }
        else
        {
            *(R1+m)='1';
            m++;
        }
    }
}
void permutation_32(char *SB1,char *f)
{
    short i,j,m=0,temp;
    for(i=0;i<8;i++)
    {
        for(j=0;j<4;j++)
        {
            temp=sb_permutation[i][j]-1;
            *(f+m)=*(SB1+temp);
            m++;
        }
    }
}
void common_permutation(char *in,char *out)
{
    short i,j,temp,m=0;
    for(i=0;i<8;i++)
    {
        for(j=0;j<8;j++)
        {
            temp=ip_inverse[i][j]-1;
            out[m]=in[temp];
            m++;
        }
    }
}
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
AES - Encryption
#include<stdio.h>
#define Nb 4
int Nr =0,Nk =0;
unsigned char in[16], out[16], state[4][4];
unsigned char RoundKey[240];
unsigned char Key[32];
int getSBoxValue(int num)
{
    int sbox[256] = {
        0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f,
        0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab,
        0x76, //0
        0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47,
        0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72,
        0xc0, //1
        0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7,
        0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31,
        0x15, //2
        0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05,
        0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2,
        0x75, //3
        0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a,
        0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f,
        0x84, //4
        0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1,
        0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58,
        0xcf, //5
        0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33,
        0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f,
        0xa8, //6
        0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38,
        0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3,
        0xd2, //7
        0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44,
        0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19,
        0x73, //8
        0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90,
        0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b,
        0xdb, //9
        0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24,
        0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4,
        0x79, //A
        0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e,
        0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae,
        0x08, //B
        0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4,
        0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b,
        0x8a, //C
        0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6,
        0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d,
        0x9e, //D
        0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e,
        0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28,
        0xdf, //E
        0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42,
        0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb,
        0x16 }; //F
    return sbox[num];
}
int Rcon[255] = {
    0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
    0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d,
    0x9a,
    0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
    0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91,
    0x39,
    0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
    0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d,
    0x3a,
    0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
    0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c,
    0xd8,
    0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6,
    0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,
    0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61,
    0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,
    0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01,
    0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
    0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e,
    0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
    0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4,
    0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
    0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8,
    0xcb
};
void KeyExpansion()
{
    int i,j;
    unsigned char temp[4],k;
    for(i=0;i<Nk;i++)
    {
        RoundKey[i*4]=Key[i*4];
        RoundKey[i*4+1]=Key[i*4+1];
        RoundKey[i*4+2]=Key[i*4+2];
        RoundKey[i*4+3]=Key[i*4+3];
    }
    while (i < (Nb * (Nr+1)))
    {
        for(j=0;j<4;j++)
        {
            temp[j]=RoundKey[(i-1) * 4 + j];
        }
        if (i % Nk == 0)
        {
            k = temp[0];
            temp[0] = temp[1];
            temp[1] = temp[2];
            temp[2] = temp[3];
            temp[3] = k;
        }
    }
}
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
{
temp[0]=getSBoxValue(temp[0]);
temp[1]=getSBoxValue(temp[1]);
temp[2]=getSBoxValue(temp[2]);
temp[3]=getSBoxValue(temp[3]);
}
temp[0] = temp[0] ^ Rcon[i/Nk];
}
else if (Nk > 6 && i % Nk == 4)
{
{
temp[0]=getSBoxValue(temp[0]);
temp[1]=getSBoxValue(temp[1]);
temp[2]=getSBoxValue(temp[2]);
temp[3]=getSBoxValue(temp[3]);
}
}
RoundKey[i*4+0]= RoundKey[(i-Nk)*4+0] ^ temp[0];
RoundKey[i*4+1]= RoundKey[(i-Nk)*4+1] ^ temp[1];
RoundKey[i*4+2]= RoundKey[(i-Nk)*4+2] ^ temp[2];
RoundKey[i*4+3]= RoundKey[(i-Nk)*4+3] ^ temp[3];
i++;
}
}
void AddRoundKey(int round)
{
int i,j;
for(i=0;i<4;i++)
{
for(j=0;j<4;j++)
{
state[j][i] ^=
RoundKey[round * Nb * 4 + i * Nb + j];
}
}
}
void SubBytes()
{
int i,j;
for(i=0;i<4;i++)
{
for(j=0;j<4;j++)
{
state[i][j] =
getSBoxValue(state[i][j]);
}
}
}
void ShiftRows()
{
unsigned char temp;
temp=state[1][0];
state[1][0]=state[1][1];
state[1][1]=state[1][2];
state[1][2]=state[1][3];
state[1][3]=temp;
temp=state[2][0];
state[2][0]=state[2][2];
state[2][2]=temp;
temp=state[2][1];
state[2][1]=state[2][3];
state[2][3]=temp;
temp=state[3][0];
state[3][0]=state[3][3];
state[3][3]=state[3][2];
state[3][2]=state[3][1];
state[3][1]=temp;
}
```

```
#define xtime(x) ((x<<1)^(((x>>7) & 1)*0x1b))
void MixColumns()
{
int i;
unsigned char Tmp,Tm,t;
for(i=0;i<4;i++)
{
t=state[0][i];
Tmp = state[0][i] ^ state[1][i] ^
state[2][i] ^ state[3][i] ;
Tm = state[0][i] ^ state[1][i] ; Tm =
xtime(Tm); state[0][i] ^= Tm ^ Tmp ;
Tm = state[1][i] ^ state[2][i] ; Tm =
xtime(Tm); state[1][i] ^= Tm ^ Tmp ;
Tm = state[2][i] ^ state[3][i] ; Tm =
xtime(Tm); state[2][i] ^= Tm ^ Tmp ;
Tm = state[3][i] ^ t ; Tm = xtime(Tm);
state[3][i] ^= Tm ^ Tmp ;
}
}
void Cipher()
{
int i,j,round=0;
for(i=0;i<4;i++)
{
for(j=0;j<4;j++)
{
state[j][i] = in[i*4 + j];
}
}
AddRoundKey(0);
for(round=1;round<Nr;round++)
{
SubBytes();
ShiftRows();
MixColumns();
AddRoundKey(round);
}
SubBytes();
ShiftRows();
AddRoundKey(Nr);
for(i=0;i<4;i++)
{
for(j=0;j<4;j++)
{
out[i*4+j]=state[j][i];
}
}
}
}
int main()
{
int i;
while(Nr!=128 && Nr!=192 && Nr!=256)
{
printf("Enter the length of Key(128, 192 or 256
only): ");
scanf("%d",&Nr);
}
Nk = Nr / 32;
Nr = Nk + 6;
unsigned char temp[16] = {0x00 ,0x01 ,0x02
,0x03 ,0x04 ,0x05 ,0x06 ,0x07 ,0x08 ,0x09
,0x0a ,0x0b ,0x0c ,0x0d ,0x0e ,0x0f};
unsigned char temp2[16]= {0x00 ,0x11 ,0x22
,0x33 ,0x44 ,0x55 ,0x66 ,0x77 ,0x88 ,0x99
,0xaa ,0xbb ,0xcc ,0xdd ,0xee ,0xff};
for(i=0;i<Nk*4;i++)
{
```



```
    Key[i]=temp[i];
    in[i]=temp2[i];
}
KeyExpansion();
Cipher();
printf("\nText after encryption:\n");
for(i=0;i<Nk*4;i++)
{
    printf("%02x ",out[i]);
}
return 1;
}
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
AES - Decryption
#include<stdio.h>
#define Nb 4
int Nr=0;
int Nk=0;
unsigned char in[16], out[16], state[4][4];
unsigned char RoundKey[240];
unsigned char Key[32];
int getSBoxInvert(int num)
{
    int rsbox[256] =
    { 0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5,
      0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7,
      0xfb,
      0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff,
      0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9,
      0xcb,
      0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23,
      0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3,
      0x4e,
      0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24,
      0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1,
      0x25,
      0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98,
      0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6,
      0x92,
      0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9,
      0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d,
      0x84,
      0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3,
      0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45,
      0x06,
      0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f,
      0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a,
      0x6b,
      0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc,
      0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6,
      0x73,
      0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35,
      0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf,
      0x6e,
      0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5,
      0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe,
      0x1b,
      0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79,
      0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a,
      0xf4,
      0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7,
      0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec,
      0x5f,
      0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a,
      0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c,
      0xef,
      0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5,
      0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99,
      0x61,
      0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6,
      0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c,
      0x7d };
    return rsbox[num];
}
int getSBoxValue(int num)
{
    int sbox[256] = {
        0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f,
        0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab,
        0x76,
```

```
0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47,
0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72,
0xc0,
0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7,
0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31,
0x15,
0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05,
0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2,
0x75,
0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a,
0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f,
0x84,
0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1,
0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58,
0xcf,
0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33,
0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f,
0xa8,
0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38,
0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3,
0xd2,
0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44,
0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19,
0x73,
0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90,
0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b,
0xdb,
0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24,
0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4,
0x79,
0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e,
0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae,
0x08,
0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4,
0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b,
0x8a,
0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6,
0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d,
0x9e,
0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e,
0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28,
0xdf,
0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42,
0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb,
0x16 };
    return sbox[num];
}
int Rcon[255] = {
    0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
    0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d,
    0x9a,
    0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
    0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91,
    0x39,
    0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
    0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d,
    0x3a,
    0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
    0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c,
    0xd8,
    0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,
    0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa,
    0xef,
    0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,
    0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66,
    0xcc,
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d,
0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80,
0x1b,
0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f,
0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4,
0xb3,
0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72,
0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a,
0x94,
0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74,
0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10,
0x20,
0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab,
0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97,
0x35,
0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5,
0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2,
0x9f,
0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83,
0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02,
0x04,
0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36,
0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc,
0x63,
0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d,
0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3,
0xbd,
0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33,
0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb
};
void KeyExpansion()
{
    int i,j;
    unsigned char temp[4],k;
    for(i=0;i<Nk;i++)
    {
        RoundKey[i*4]=Key[i*4];
        RoundKey[i*4+1]=Key[i*4+1];
        RoundKey[i*4+2]=Key[i*4+2];
        RoundKey[i*4+3]=Key[i*4+3];
    }
    while (i < (Nb * (Nr+1)))
    {
        for(j=0;j<4;j++)
        {
            temp[j]=RoundKey[(i-1) * 4 + j];
        }
        if (i % Nk == 0)
        {
            {
                k = temp[0];
                temp[0] = temp[1];
                temp[1] = temp[2];
                temp[2] = temp[3];
                temp[3] = k;
            }
            {
                temp[0]=getSBoxValue(temp[0]);
                temp[1]=getSBoxValue(temp[1]);
                temp[2]=getSBoxValue(temp[2]);
                temp[3]=getSBoxValue(temp[3]);
            }
        }
        temp[0] = temp[0] ^ Rcon[i/Nk];
    }
    else if (Nk > 6 && i % Nk == 4)
    {
        {
            temp[0]=getSBoxValue(temp[0]);
```

```
temp[1]=getSBoxValue(temp[1]);
temp[2]=getSBoxValue(temp[2]);
temp[3]=getSBoxValue(temp[3]);
}
}
RoundKey[i*4+0] = RoundKey[(i-Nk)*4+0] ^
temp[0];
RoundKey[i*4+1] = RoundKey[(i-Nk)*4+1] ^
temp[1];
RoundKey[i*4+2] = RoundKey[(i-Nk)*4+2] ^
temp[2];
RoundKey[i*4+3] = RoundKey[(i-Nk)*4+3] ^
temp[3];
i++;
}
}
void AddRoundKey(int round)
{
    int i,j;
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            state[j][i] ^=
RoundKey[round * Nb * 4 + i * Nb + j];
        }
    }
}
void InvSubBytes()
{
    int i,j;
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            state[i][j] =
getSBoxInvert(state[i][j]);
        }
    }
}
void InvShiftRows()
{
    unsigned char temp;
    temp=state[1][3];
    state[1][3]=state[1][2];
    state[1][2]=state[1][1];
    state[1][1]=state[1][0];
    state[1][0]=temp;
    temp=state[2][0];
    state[2][0]=state[2][2];
    state[2][2]=temp;
    temp=state[2][1];
    state[2][1]=state[2][3];
    state[2][3]=temp;

    temp=state[3][0];
    state[3][0]=state[3][1];
    state[3][1]=state[3][2];
    state[3][2]=state[3][3];
    state[3][3]=temp;
}
#define xtime(x) ((x<<1) ^ ((x>>7) & 1) *
0x1b))
#define Multiply(x,y) (((y & 1) * x) ^ ((y>>1 &
1) * xtime(x)) ^ ((y>>2 & 1) * xtime(xtime(x)))
^ ((y>>3 & 1) * xtime(xtime(xtime(x)))) ^ ((y>>4
& 1) * xtime(xtime(xtime(xtime(x))))))
void InvMixColumns()
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
{
    int i;
    unsigned char a,b,c,d;
    for(i=0;i<4;i++)
    {
        a = state[0][i];
        b = state[1][i];
        c = state[2][i];
        d = state[3][i];
        state[0][i] = Multiply(a, 0x0e) ^
Multiply(b, 0x0b) ^ Multiply(c, 0x0d) ^
Multiply(d, 0x09);
        state[1][i] = Multiply(a, 0x09) ^
Multiply(b, 0x0e) ^ Multiply(c, 0x0b) ^
Multiply(d, 0x0d);
        state[2][i] = Multiply(a, 0x0d) ^
Multiply(b, 0x09) ^ Multiply(c, 0x0e) ^
Multiply(d, 0x0b);
        state[3][i] = Multiply(a, 0x0b) ^
Multiply(b, 0x0d) ^ Multiply(c, 0x09) ^
Multiply(d, 0x0e);
    }
}
void InvCipher()
{
    int i,j,round=0;
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            state[j][i] = in[i*4 + j];
        }
    }
    AddRoundKey(Nr);
    for(round=Nr-1;round>0;round--)
    {
        InvShiftRows();
        InvSubBytes();
        AddRoundKey(round);
        InvMixColumns();
    }
    InvShiftRows();
    InvSubBytes();
    AddRoundKey(0);
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            out[i*4+j]=state[j][i];
        }
    }
}
int main()
{
    int i;
    while(Nr!=128 && Nr!=192 && Nr!=256)
    {
        printf("Enter the length of
Key(128, 192 or 256 only): ");
        scanf("%d",&Nr);
    }
    Nk = Nr / 32;
    Nr = Nk + 6;
    unsigned char temp[32] = {0x00 ,0x01
,0x02 ,0x03 ,0x04 ,0x05 ,0x06 ,0x07 ,0x08
,0x09 ,0x0a ,0x0b ,0x0c ,0x0d ,0x0e
,0x0f};
    unsigned char temp2[16]= {0x69 ,0xc4
,0xe0 ,0xd8 ,0x6a ,0x7b ,0x04 ,0x30 ,0xd8
,0xcd ,0xb7 ,0x80 ,0x70 ,0xb4 ,0xc5
,0x5a};
    for(i=0;i<Nk*4;i++)
    {
        Key[i]=temp[i];
        in[i]=temp2[i];
    }
    KeyExpansion();
    InvCipher();
    printf("\nText after decryption:\n");
    for(i=0;i<Nb*4;i++)
    {
        printf("%02x ",out[i]);
    }
    printf("\n\n");
}
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
RSA
#include<stdio.h>
#include<conio.h>
int phi,M,n,e,d,C,FLAG;
int check()
{
    int i;
    for(i=3;e%i==0 && phi%i==0;i+2)
    {
        FLAG = 1;
        return;
    }
    FLAG = 0;
}
void encrypt()
{
    int i;
    C = 1;
    for(i=0;i< e;i++)
        C=C*M%n;
    C = C%n;
    printf("\n\tEncrypted keyword : %d",C);
}
void decrypt()
{
    int i;
    M = 1;
    for(i=0;i< d;i++)
        M=M*C%n;
    M = M%n;
    printf("\n\tDecrypted keyword : %d",M);
}
void main()
{
    int p,q,s;
    printf("Enter Two Relatively Prime Numbers\t: ");
    scanf("%d%d",&p,&q);
    n = p*q;
    phi=(p-1)*(q-1);
    printf("\n\tF(n)\t= %d",phi);
    do
    {
        printf("\n\nEnter e\t: ");
        scanf("%d",&e);
        check();
    }while(FLAG==1);
    d = 1;
    do
    {
        s = (d*e)%phi;
        d++;
    }while(s!=1);
    d = d-1;
    printf("\n\tPublic Key\t: {%d,%d}",e,n);
    printf("\n\tPrivate Key\t: {%d,%d}",d,n);
    printf("\n\nEnter The Plain Text\t: ");
    scanf("%d",&M);
    encrypt();
    printf("\n\nEnter the Cipher text\t: ");
    scanf("%d",&C);
    decrypt();
    getch();
}
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```
SHA-1
#include<stdio.h>
#include<string.h>
#include<malloc.h>
#include<math.h>
#include<stdlib.h>
#define rotateleft(x,n) ((x<<n) | (x>>(32-n)))
#define rotateright(x,n) ((x>>n) | (x<<(32-n)))
void SHA1(unsigned char * str1)
{
    unsigned long int
    h0,h1,h2,h3,h4,a,b,c,d,e,f,k,temp;
    int m,i,j;
    h0 = 0x67452301;
    h1 = 0xEFCDAB89;
    h2 = 0x98BADCFE;
    h3 = 0x10325476;
    h4 = 0xC3D2E1F0;
    unsigned char * str;
    str = (unsigned char *)malloc(strlen((const char
    *)str1)+100);
    strcpy((char *)str,(const char *)str1);
    int current_length = strlen((const char *)str);
    int original_length = current_length;
    str[current_length] = 0x80;
    str[current_length + 1] = '\0';
    char ic = str[current_length];
    current_length++;
    int ib = current_length % 64;
    if(ib<56)
    ib = 56-ib;
    else
    ib = 120 - ib;
    for(i=0;i < ib;i++)
    {
        str[current_length]=0x00;
        current_length++;
    }
    str[current_length + 1]='\0';
    for(i=0;i<6;i++)
    {
        str[current_length]=0x0;
        current_length++;
    }
    str[current_length] = (original_length * 8) /
    0x100 ;
    current_length++;
    str[current_length] = (original_length * 8) %
    0x100;
    current_length++;
    str[current_length+i]='\0';
    int number_of_chunks = current_length/64;
    unsigned long int word[80];
    for(i=0;i<number_of_chunks;i++)
    {
        for(j=0;j<16;j++)
        {
            word[j] = str[i*64 + j*4 + 0] * 0x1000000 +
            str[i*64 + j*4 + 1] * 0x10000 + str[i*64 + j*4 +
            2] * 0x100 + str[i*64 + j*4 + 3];
        }
        for(j=16;j<80;j++)
        {
            word[j] = rotateleft((word[j-3] ^ word[j-8] ^
            word[j-14] ^ word[j-16]),1);
        }
        a = h0;
        b = h1;
        c = h2;
        d = h3;
        e = h4;
        for(m=0;m<80;m++)
        {
            if(m<=19)
            {
                f = (b & c) | ((~b) & d);
                k = 0x5A827999;
            }
            else if(m<=39)
            {
                f = b ^ c ^ d;
                k = 0x6ED9EBA1;
            }
            else if(m<=59)
            {
                f = (b & c) | (b & d) | (c & d);
                k = 0x8F1BBCDC;
            }
            else
            {
                f = b ^ c ^ d;
                k = 0xCA62C1D6;
            }
            temp = (rotateleft(a,5) + f + e + k + word[m]) &
            0xFFFFFFFF;
            e = d;
            d = c;
            c = rotateleft(b,30);
            b = a;
            a = temp;
        }
        h0 = h0 + a;
        h1 = h1 + b;
        h2 = h2 + c;
        h3 = h3 + d;
        h4 = h4 + e;
    }
    printf("\n\n");
    printf("Hash: %x %x %x %x %x",h0, h1, h2, h3,
    h4);
    printf("\n\n");
}

void main()
{
    unsigned char * msg = "The quick brown fox jumps
    over the lazy dog";
    printf("-----\n");
    printf("Input String to be Encrypted using SHA-1
    : \n\t%s",msg);
    SHA1((unsigned char *)msg);
}
```

## Thực hành – Mật mã học cơ sở - HTT – Học viện CNBCVT cơ sở tại TP.HCM

```

MD5
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
typedef union uwb {
    unsigned w;
    unsigned char b[4];
} MD5union;
typedef unsigned DigestArray[4];
unsigned func0( unsigned abcd[] ){
    return ( abcd[1] & abcd[2]) | (~abcd[1] &
abcd[3]);}
unsigned func1( unsigned abcd[] ){
    return ( abcd[3] & abcd[1]) | (~abcd[3] &
abcd[2]);}
unsigned func2( unsigned abcd[] ){
    return abcd[1] ^ abcd[2] ^ abcd[3];}
unsigned func3( unsigned abcd[] ){
    return abcd[2] ^ (abcd[1] |~ abcd[3]);}
typedef unsigned (*DgstFctn)(unsigned a[]);
unsigned *calctable( unsigned *k)
{
    double s, pwr;
    int i;
    pwr = pow( 2, 32);
    for (i=0; i<64; i++) {
        s = fabs(sin(1+i));
        k[i] = (unsigned)( s * pwr );
    }
    return k;
}
unsigned rol( unsigned r, short N )
{
    unsigned mask1 = (1<<N) -1;
    return ((r>>(32-N)) & mask1) | ((r<<N) &
~mask1);
}
unsigned *md5( const char *msg, int mlen)
{
    static DigestArray h0 = { 0x67452301,
0xEFCDAB89, 0x98BADCFE, 0x10325476 };
    static DgstFctn ff[] = { &func0, &func1,
&func2, &func3 };
    static short M[] = { 1, 5, 3, 7 };
    static short O[] = { 0, 1, 5, 0 };
    static short rot0[] = { 7,12,17,22};
    static short rot1[] = { 5, 9,14,20};
    static short rot2[] = { 4,11,16,23};
    static short rot3[] = { 6,10,15,21};
    static short *rots[] = {rot0, rot1, rot2,
rot3 };
    static unsigned kspace[64];
    static unsigned *k;
    static DigestArray h;
    DigestArray abcd;
    DgstFctn fctn;
    short m, o, g;
    unsigned f;
    short *rotn;
    union {
        unsigned w[16];
        char b[64];
    }mm;
    int os = 0;
    int grp, grps, q, p;
    unsigned char *msg2;
    if (k==NULL) k= calctable(kspace);

    for (q=0; q<4; q++) h[q] = h0[q];
    {
        grps = 1 + (mlen+8)/64;
        msg2 = malloc( 64*grps);
        memcpy( msg2, msg, mlen);
        msg2[mlen] = (unsigned char)0x80;
        q = mlen + 1;
        while (q < 64*grps){ msg2[q] = 0; q++ ;
        }
        {
            MD5union u;
            u.w = 8*mlen;
            q -= 8;
            memcpy(msg2+q, &u.w, 4 );
        }
        for (grp=0; grp<grps; grp++)
        {
            memcpy( mm.b, msg2+os, 64);
            for(q=0;q<4;q++) abcd[q] = h[q];
            for (p = 0; p<4; p++) {
                fctn = ff[p];
                rotn = rots[p];
                m = M[p]; o= O[p];
                for (q=0; q<16; q++) {
                    g = (m*q + o) % 16;
                    f = abcd[1] + rol( abcd[0]+
fctn(abcd) + k[q+16*p] + mm.w[g], rotn[q%4]);
                    abcd[0] = abcd[3];
                    abcd[3] = abcd[2];
                    abcd[2] = abcd[1];
                    abcd[1] = f;
                }
            }
            for (p=0; p<4; p++)
                h[p] += abcd[p];
            os += 64;
        }
        return h;
    }
}
int main( int argc, char *argv[] )
{
    int j,k;
    const char *msg = "This code has been
provided for students";
    printf("Input String to be Encrypted using
MD5 : \n\t%s",msg);
    unsigned *d = md5(msg, strlen(msg));
    MD5union u;
    printf("\n\n\nThe MD5 code for input string
is : \n");
    printf("\t= 0x");
    for (j=0;j<4;j++){
        u.w = d[j];
        for (k=0;k<4;k++) printf("%02x",u.b[k]);
    }
    printf("\n");
    printf("\n\t MD5 Encyption Successfully
Completed!!!\n\n");
    getch();
    system("pause");
    return 0;
}

```