```
/*************************************************************
FileName:
     SM4.h
Version:
     SM4_V1.0
Date:
     Sep 13,2016
Description:
     This headfile provide macro defination, parameter definition and function declaration needed
in SM4 algorithm implement.
Function List:
     1. SM4_KeySchedule //Generate the required round keys
     2. SM4_Encrypt     //Encryption function
     3. SM4_Decrypt     //Decryption function
History:
     Date:Sep 13,2016
     Author:Mao Yingying,Huo Lili
     Modification:Adding notes to all the functions
*************************************************************/


#include<stdio.h>

//rotate n bits to the left in a 32bit buffer
#define SM4_Rotl32(buf, n) (((buf)<<n)|((buf)>>(32-n)))

unsigned int SM4_CK[32] ={0x00070e15, 0x1c232a31, 0x383f464d, 0x545b6269,
                          0x70777e85, 0x8c939aa1, 0xa8afb6bd, 0xc4cbd2d9,
                          0xe0e7eef5, 0xfc030a11, 0x181f262d, 0x343b4249,
                          0x50575e65, 0x6c737a81, 0x888f969d, 0xa4abb2b9,
                          0xc0c7ced5, 0xdce3eaf1, 0xf8ff060d, 0x141b2229,
                          0x30373e45, 0x4c535a61, 0x686f767d, 0x848b9299,
                          0xa0a7aeb5, 0xbcc3cad1, 0xd8dfe6ed, 0xf4fb0209,
                          0x10171e25, 0x2c333a41, 0x484f565d, 0x646b7279};

unsigned char SM4_Sbox[256] =

{0xd6,0x90,0xe9,0xfe,0xcc,0xe1,0x3d,0xb7,0x16,0xb6,0x14,0xc2,0x28,0xfb,0x2c,0x05,
0x2b,0x67,0x9a,0x76,0x2a,0xbe,0x04,0xc3,0xaa,0x44,0x13,0x26,0x49,0x86,0x06,0x99,
0x9c,0x42,0x50,0xf4,0x91,0xef,0x98,0x7a,0x33,0x54,0x0b,0x43,0xed,0xcf,0xac,0x62,
0xe4,0xb3,0x1c,0xa9,0xc9,0x08,0xe8,0x95,0x80,0xdf,0x94,0xfa,0x75,0x8f,0x3f,0xa6,
0x47,0x07,0xa7,0xfc,0xf3,0x73,0x17,0xba,0x83,0x59,0x3c,0x19,0xe6,0x85,0x4f,0xa8,
0x68,0x6b,0x81,0xb2,0x71,0x64,0xda,0x8b,0xf8,0xeb,0x0f,0x4b,0x70,0x56,0x9d,0x35,
0x1e,0x24,0x0e,0x5e,0x63,0x58,0xd1,0xa2,0x25,0x22,0x7c,0x3b,0x01,0x21,0x78,0x87,
```

```
0xd4,0x00,0x46,0x57,0x9f,0xd3,0x27,0x52,0x4c,0x36,0x02,0xe7,0xa0,0xc4,0xc8,0x9e,
0xea,0xbf,0x8a,0xd2,0x40,0xc7,0x38,0xb5,0xa3,0xf7,0xf2,0xce,0xf9,0x61,0x15,0xa1,
0xe0,0xae,0x5d,0xa4,0x9b,0x34,0x1a,0x55,0xad,0x93,0x32,0x30,0xf5,0x8c,0xb1,0xe3,
0x1d,0xf6,0xe2,0x2e,0x82,0x66,0xca,0x60,0xc0,0x29,0x23,0xab,0x0d,0x53,0x4e,0x6f,
0xd5,0xdb,0x37,0x45,0xde,0xfd,0x8e,0x2f,0x03,0xff,0x6a,0x72,0x6d,0x6c,0x5b,0x51,
0x8d,0x1b,0xaf,0x92,0xbb,0xdd,0xbc,0x7f,0x11,0xd9,0x5c,0x41,0x1f,0x10,0x5a,0xd8,
0x0a,0xc1,0x31,0x88,0xa5,0xcd,0x7b,0xbd,0x2d,0x74,0xd0,0x12,0xb8,0xe5,0xb4,0xb0,
0x89,0x69,0x97,0x4a,0x0c,0x96,0x77,0x7e,0x65,0xb9,0xf1,0x09,0xc5,0x6e,0xc6,0x84,
0x18,0xf0,0x7d,0xec,0x3a,0xdc,0x4d,0x20,0x79,0xee,0x5f,0x3e,0xd7,0xcb,0x39,0x48};

unsigned int SM4_FK[4] = {0xA3B1BAC6, 0x56AA3350, 0x677D9197, 0xB27022DC};


/*************************************************************
Function:
      void SM4_KeySchedule(unsigned char MK[],  unsigned int rk[]);
Description:
      Generate round keys
Calls:
Called By:
      SM4_Encrypt;
      SM4_Decrypt;
Input:
      MK[]: Master key
Output:
      rk[]: round keys
Return:null
Others:
*************************************************************/
void SM4_KeySchedule(unsigned char MK[],unsigned int rk[])
{
    unsigned int tmp,buf,K[36];
    int i;

    for(i=0;i<4;i++)
     {
              K[i]=SM4_FK[i]^( (MK[4*i]<<24) | (MK[4*i+1]<<16)
                              |(MK[4*i+2]<<8) | (MK[4*i+3]) );
     }

    for(i=0;i<32;i++)
     {
        tmp =K[i+1]^K[i+2]^K[i+3]^ SM4_CK[i];
```

```
        //nonlinear operation
        buf= (SM4_Sbox[(tmp >> 24) & 0xFF]) << 24
             |(SM4_Sbox[(tmp >> 16) & 0xFF]) << 16
             |(SM4_Sbox[(tmp >> 8)  & 0xFF]) << 8
             |(SM4_Sbox[tmp & 0xFF]);

         //linear operation
         K[i+4]=K[i]^((buf)^(SM4_Rotl32((buf),13))^(SM4_Rotl32((buf),23)));

         rk[i]=K[i+4];
    }
}


/*************************************************************
Function:
      void SM4_Encrypt(unsigned char MK[],unsigned char PlainText[],unsigned char
CipherText[]);
Description:
      Encryption function
Calls:
      SM4_KeySchedule
Called By:
Input:
      MK[]: Master key
      PlainText[]: input text
Output:
      CipherText[]: output text
Return:null
Others:
*************************************************************/
void SM4_Encrypt(unsigned char MK[],unsigned char PlainText[],unsigned char CipherText[])
{
    unsigned int rk[32],X[36],tmp,buf;
    int i,j;

    SM4_KeySchedule(MK,rk);

    for(j=0;j<4;j++)
     {
           X[j]=(PlainText[j*4]<<24) |(PlainText[j*4+1]<<16)
               |(PlainText[j*4+2]<<8)|(PlainText[j*4+3]);
     }
```

```
    for(i=0;i<32;i++)
     {
               tmp = X[i+1]^X[i+2]^X[i+3]^rk[i];

               //nonlinear operation
                buf=  ( SM4_Sbox[(tmp >> 24) & 0xFF]) << 24
                      |(SM4_Sbox[(tmp >> 16) & 0xFF]) << 16
                      |(SM4_Sbox[(tmp >> 8)  & 0xFF]) << 8
                      |(SM4_Sbox[tmp & 0xFF]);

               //linear operation
               X[i+4]=X[i]^(buf^SM4_Rotl32((buf),2)^ SM4_Rotl32((buf),10)
                          ^ SM4_Rotl32((buf),18)^ SM4_Rotl32((buf),24));
         }

    for(j=0;j<4;j++)
     {
              CipherText[4*j]=(X[35-j]>> 24)& 0xFF;
              CipherText[4*j+1]=(X[35-j]>> 16)& 0xFF;
              CipherText[4*j+2]=(X[35-j]>> 8)& 0xFF;
              CipherText[4*j+3]=(X[35-j])& 0xFF;
     }
}


/************************************************************
Function:
     void SM4_Decrypt(unsigned char MK[],unsigned char CipherText[], unsigned char PlainText[]);
Description:
     Decryption function
Calls:
     SM4_KeySchedule
Called By:
Input:
     MK[]: Master key
     CipherText[]: input text
Output:
     PlainText[]: output text
Return:null
Others:
************************************************************/
void SM4_Decrypt(unsigned char MK[],unsigned char CipherText[],unsigned char PlainText[])
{
    unsigned int rk[32],X[36],tmp,buf;
```

```
    int i,j;

    SM4_KeySchedule(MK,rk);

    for(j=0;j<4;j++)
     {
            X[j]=(CipherText[j*4]<<24) |(CipherText[j*4+1]<<16)|
                 (CipherText[j*4+2]<<8)|(CipherText[j*4+3]);
     }

    for(i=0;i<32;i++)
        {
            tmp = X[i+1]^X[i+2]^X[i+3]^rk[31-i];

            //nonlinear operation
            buf= (SM4_Sbox[(tmp >> 24) & 0xFF]) << 24
                |(SM4_Sbox[(tmp >> 16) & 0xFF]) << 16
                |(SM4_Sbox[(tmp >> 8)  & 0xFF]) << 8
                |(SM4_Sbox[tmp & 0xFF]);
            //linear operation
            X[i+4]=X[i]^(buf^SM4_Rotl32((buf),2)^ SM4_Rotl32((buf),10)
                        ^ SM4_Rotl32((buf),18)^ SM4_Rotl32((buf),24));
      }

    for(j=0;j<4;j++)
     {
              PlainText[4*j]=(X[35-j]>> 24)& 0xFF;
              PlainText[4*j+1]=(X[35-j]>>16)& 0xFF;
              PlainText[4*j+2]=(X[35-j]>> 8)& 0xFF;
              PlainText[4*j+3]=(X[35-j])& 0xFF;
     }
}
```