```
///*************************************************************************
//  File name:    SM9_Key_ex.h
//  Version:      SM9_Key_ex_V1.0
//  Date:         Jan 1,2017
//  Description:  implementation of SM9 Key Exchange Protocol
//                all operations based on BN curve line function
//  Function List:
//        1.bytes128_to_ecn2        //convert 128 bytes into ecn2
//        2.zzn12_ElementPrint      //print all element of struct zzn12
//        3.LinkCharZzn12           //link two different types(unsigned char and zzn12)to
one(unsigned char)
//        4.Test_Point              //test if the given point is on SM9 curve
//        5.SM9_KeyEx_KDF           //calculate KDF(IDA||IDB||RA||RB||g1||g2||g3)
//        6.SM9_KeyEx_Hash          //calculate
Hash(hashid||g1||Hash(g2||g3||IDA||IDB||RA||RB))
//        7.SM9_H1                  //function H1 in SM9 standard 5.4.2.2
//        8.SM9_Init                //initiate SM9 curve
//        9.SM9_GenerateEncryptKey  //generate encrypted private and public key
//        10.SM9_KeyEx_InitA_I      //calculate RA (Step A1-A4)
//        11.SM9_KeyEx_ReB_I        //calculate RB ,a hash Value SB and a shared key SKB(Step
B1-A7)
//        12.SM9_KeyEx_InitA_II     //initiator A calculate the secret key SKA and a hash
                                    //SA which responder B might verifies(Step A5-A7)
//        13.SM9_KeyEx_ReB_II       //Step B10 (optional) verifies the hash value SA received
from initiator A
//        14.SM9_SelfCheck()        //SM9 slef-check

//
// Notes:
//   This SM9 implementation source code can be used for academic, non-profit making or
non-commercial use only.
//   This SM9 implementation is created on MIRACL. SM9 implementation source code provider does
not provide MIRACL library, MIRACL license or any permission to use MIRACL library. Any commercial
use of MIRACL requires a license which may be obtained from Shamus Software Ltd.

//*************************************************************************/


#include<malloc.h>
#include<math.h>
#include "miracl.h"
#include "R-ate.h"

#define BNLEN          32      //BN curve with 256bit is used in SM9 algorithm
```

```
#define SM9_ASK_MEMORY_ERR          0x00000001   //ask for memory fail
#define SM9_MEMBER_ERR              0x00000002   //the order of group G error
#define SM9_MY_ECAP_12A_ERR         0x00000003   //R-ate pairing generated error
#define SM9_NOT_VALID_G1            0x00000004   //not valid element of G1
#define SM9_G1BASEPOINT_SET_ERR     0x00000005   //base point of G1 seted error
#define SM9_G2BASEPOINT_SET_ERR     0x00000006   //base point of G2 seted error
#define SM9_GEPUB_ERR               0x00000007   //pubkey error
#define SM9_GEPRI_ERR               0x00000008   //privare key error
#define SM9_ERR_CMP_S1SB            0x00000009   //S1!=SB
#define SM9_ERR_CMP_S2SA            0x0000000A   //S2!=SA
#define SM9_ERR_RA                  0x0000000B   //RA error
#define SM9_ERR_RB                  0x0000000C   //RB error
#define SM9_ERR_SA                  0x0000000D   //SA error
#define SM9_ERR_SB                  0x0000000E   //SB error


unsigned char SM9_q[32] =
{0xB6,0x40,0x00,0x00,0x02,0xA3,0xA6,0xF1,0xD6,0x03,0xAB,0x4F,0xF5,0x8E,0xC7,0x45,
0x21,0xF2,0x93,0x4B,0x1A,0x7A,0xEE,0xDB,0xE5,0x6F,0x9B,0x27,0xE3,0x51,0x45,0x7D};
unsigned char SM9_N[32] =
{0xB6,0x40,0x00,0x00,0x02,0xA3,0xA6,0xF1,0xD6,0x03,0xAB,0x4F,0xF5,0x8E,0xC7,0x44,
0x49,0xF2,0x93,0x4B,0x18,0xEA,0x8B,0xEE,0xE5,0x6E,0xE1,0x9C,0xD6,0x9E,0xCF,0x25};

unsigned char SM9_P1x[32]=
{0x93,0xDE,0x05,0x1D,0x62,0xBF,0x71,0x8F,0xF5,0xED,0x07,0x04,0x48,0x7D,0x01,0xD6,
0xE1,0xE4,0x08,0x69,0x09,0xDC,0x32,0x80,0xE8,0xC4,0xE4,0x81,0x7C,0x66,0xDD,0xDD};
unsigned char SM9_P1y[32]=
{0x21,0xFE,0x8D,0xDA,0x4F,0x21,0xE6,0x07,0x63,0x10,0x65,0x12,0x5C,0x39,0x5B,0xBC,
0x1C,0x1C,0x00,0xCB,0xFA,0x60,0x24,0x35,0x0C,0x46,0x4C,0xD7,0x0A,0x3E,0xA6,0x16};

unsigned char SM9_P2[128]=
{0x85,0xAE,0xF3,0xD0,0x78,0x64,0x0C,0x98,0x59,0x7B,0x60,0x27,0xB4,0x41,0xA0,0x1F,
0xF1,0xDD,0x2C,0x19,0x0F,0x5E,0x93,0xC4,0x54,0x80,0x6C,0x11,0xD8,0x80,0x61,0x41,
0x37,0x22,0x75,0x52,0x92,0x13,0x0B,0x08,0xD2,0xAA,0xB9,0x7F,0xD3,0x4E,0xC1,0x20,
0xEE,0x26,0x59,0x48,0xD1,0x9C,0x17,0xAB,0xF9,0xB7,0x21,0x3B,0xAF,0x82,0xD6,0x5B,
0x17,0x50,0x9B,0x09,0x2E,0x84,0x5C,0x12,0x66,0xBA,0x0D,0x26,0x2C,0xBE,0xE6,0xED,
0x07,0x36,0xA9,0x6F,0xA3,0x47,0xC8,0xBD,0x85,0x6D,0xC7,0x6B,0x84,0xEB,0xEB,0x96,
0xA7,0xCF,0x28,0xD5,0x19,0xBE,0x3D,0xA6,0x5F,0x31,0x70,0x15,0x3D,0x27,0x8F,0xF2,
0x47,0xEF,0xBA,0x98,0xA7,0x1A,0x08,0x11,0x62,0x15,0xBB,0xA5,0xC9,0x99,0xA7,0xC7};

unsigned char SM9_t[32] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x60,0x00,0x00,0x00,0x00,0x58,0xF9,0x8A};
unsigned char SM9_a[32] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char SM9_b[32] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x05};


epoint *P1;
ecn2 P2;
big N;  //order of group, N(t)
big para_a,para_b,para_t,para_q;


BOOL bytes128_to_ecn2(unsigned char Ppubs[],ecn2 *res);
void zzn12_ElementPrint(zzn12 x);
void LinkCharZzn12(unsigned char *message,int len,zzn12 w,unsigned char *Z,int Zlen);
int Test_Point(epoint* point);
int SM9_KeyEx_KDF(unsigned char *IDA,unsigned char *IDB,epoint *RA,epoint *RB,zzn12 g1,zzn12
g2,zzn12 g3,int klen,unsigned char K[]);
int SM9_KeyEx_Hash(unsigned char hashid[],unsigned char *IDA,unsigned char *IDB,epoint
*RA,epoint *RB,zzn12 g1,zzn12 g2,zzn12 g3,unsigned char hash[]);
int SM9_H1(unsigned char Z[],int Zlen,big n,big h1);
int SM9_Init();
int SM9_GenerateEncryptKey(unsigned char hid[],unsigned char *ID,int IDlen,big ke,unsigned char
Ppubs[],unsigned char deB[]);
int SM9_KeyEx_InitA_I(unsigned char hid[],unsigned char *IDB,unsigned char randA[],
                      unsigned char Ppub[],unsigned char deA[],epoint *RA);
int SM9_KeyEx_ReB_I(unsigned char hid[],unsigned char *IDA,unsigned char *IDB,unsigned char
randB[],unsigned char Ppub[],unsigned char deB[],epoint *RA,epoint *RB,unsigned char SB[],zzn12
*g1,zzn12 *g2,zzn12 *g3);
int SM9_KeyEx_InitA_II(unsigned char *IDA,unsigned char *IDB,unsigned char randA[],unsigned char
Ppub[],unsigned char deA[],epoint *RA,epoint *RB,unsigned char SB[],unsigned char SA[]);
int SM9_KeyEx_ReB_II(unsigned char *IDA,unsigned char *IDB,zzn12 g1,zzn12 g2,zzn12 g3,epoint
*RA,epoint *RB,unsigned char SA[]);
int SM9_SelfCheck();
```