[toc]

《密码系统设计》实验

实验二 密码算法实现

4-6 学时实践要求（30 分）

1. 在 Ubuntu或openEuler中（推荐 openEuler）中调试运行教材提供的源代码，至少运行SM2，SM3，SM4代码，使用GmSSL命令验证你代码的正确性，使用Markdown记录详细记录实践过程，每完成一项功能或者一个函数git commit 一次。（15分）

- 下载并整理教材的源代码
  - 下载并解压云班课/资源/课程参考资料/Windows.C.C++.加密解密实战.rocsrc.zip
  - 结合教材确定代码所属内容（见GitHub仓库中“实验/代码/教材源代码”）
- 调试运行sm3代码
  - 4-1代码编译运行结果（一段式SM3算法）
    - 原先代码报错

```
test.cpp: In function ‘long unsigned int SL(long unsigned int,
int)’:
test.cpp:17:26: error: expected initializer before ‘x’
17 |         unsigned __int64 x = X;
   |                          ^
test.cpp:18:9: error: ‘x’ was not declared in this scope
18 |         x = x << (n % 32);
   |         ^
test.cpp: In function ‘void SM3Hash(unsigned char*, int, unsigned
char*)’:
test.cpp:154:9: error: ‘memset’ was not declared in this scope
154 |         memset(m1, 0, m1l);
    |         ^~~~~
test.cpp:9:1: note: ‘memset’ is defined in header ‘<cstring>’; did
you forget to ‘#include <cstring>’?
   8 | #include <memory>
+++ |+#include <cstring>
   9 |
test.cpp:155:9: error: ‘memcpy’ was not declared in this scope
155 |         memcpy(m1, m, l / 8);
    |         ^~~~~
test.cpp:155:9: note: ‘memcpy’ is defined in header ‘<cstring>’;
did you forget to ‘#include <cstring>’?
test.cpp: At global scope:
test.cpp:223:1: error: ‘::main’ must return ‘int’
223 | void main()
    | ^~~~
```

    - 代码修改结果如下：

```
#include <stdio.h>
#include <memory>
#include <cstring>
#include <cstdint>  // 包含标准整数类型

unsigned char IV[256 / 8] = {
0x73,0x80,0x16,0x6f,0x49,0x14,0xb2,0xb9,0x17,0x24,0x42,0xd7,0xda,0
x8a,0x06,0x00,0xa9,0x6f,0x30,0xbc,0x16,0x31,0x38,0xaa,0xe3,0x8d,0x
ee,0x4d,0xb0,0xfb,0x0e,0x4e };

// 循环左移
unsigned long SL(unsigned long X, int n)
{
        uint64_t x = X;  // 修改为标准类型
        x = x << (n % 32);
        unsigned long l = (unsigned long)(x >> 32);
        return x | l;
}

unsigned long Tj(int j)
{
        if (j <= 15)
        {
                return 0x79cc4519;
        }
        else
        {
                return 0x7a879d8a;
        }
}

unsigned long FFj(int j, unsigned long X, unsigned long Y,
unsigned long Z)
{
        if (j <= 15)
        {
                return X ^ Y ^ Z;
        }
        else
        {
                return (X & Y) | (X & Z) | (Y & Z);
        }
}

unsigned long GGj(int j, unsigned long X, unsigned long Y,
unsigned long Z)
{
        if (j <= 15)
        {
                return X ^ Y ^ Z;
        }
        else
```

```
        {
                return (X & Y) | (~X & Z);
        }
}

unsigned long P0(unsigned long X)
{
        return X ^ SL(X, 9) ^ SL(X, 17);
}

unsigned long P1(unsigned long X)
{
        return X ^ SL(X, 15) ^ SL(X, 23);
}

// 扩展
void EB(unsigned char Bi[512 / 8], unsigned long W[68], unsigned
long W1[64])
{
        // Bi 分为W0~W15
        for (int i = 0; i < 16; ++i)
        {
                W[i] = Bi[i * 4] << 24 | Bi[i * 4 + 1] << 16 |
Bi[i * 4 + 2] << 8 | Bi[i * 4 + 3];
        }

        for (int j = 16; j <= 67; ++j)
        {
                W[j] = P1(W[j - 16] ^ W[j - 9] ^ SL(W[j - 3], 15))
^ SL(W[j - 13], 7) ^ W[j - 6];
        }

        for (int j = 0; j <= 63; ++j)
        {
                W1[j] = W[j] ^ W[j + 4];
        }
}

// 压缩函数
void CF(unsigned char Vi[256 / 8], unsigned char Bi[512 / 8],
unsigned char Vi1[256 / 8])
{
        // Bi 扩展为132个字
        unsigned long W[68] = { 0 };
        unsigned long W1[64] = { 0 };

        EB(Bi, W, W1);

        // 串联 ABCDEFGH = Vi
        unsigned long R[8] = { 0 };
        for (int i = 0; i < 8; ++i)
        {
                R[i] = ((unsigned long)Vi[i * 4]) << 24 |
((unsigned long)Vi[i * 4 + 1]) << 16 | ((unsigned long)Vi[i * 4 +
```

```
2]) << 8 | ((unsigned long)Vi[i * 4 + 3]);
        }

        unsigned long A = R[0], B = R[1], C = R[2], D = R[3], E =
R[4], F = R[5], G = R[6], H = R[7];

        unsigned long SS1, SS2, TT1, TT2;
        for (int j = 0; j <= 63; ++j)
        {
                SS1 = SL(SL(A, 12) + E + SL(Tj(j), j), 7);
                SS2 = SS1 ^ SL(A, 12);
                TT1 = FFj(j, A, B, C) + D + SS2 + W1[j];
                TT2 = GGj(j, E, F, G) + H + SS1 + W[j];
                D = C;
                C = SL(B, 9);
                B = A;
                A = TT1;
                H = G;
                G = SL(F, 19);
                F = E;
                E = P0(TT2);
        }

        // Vi1 = ABCDEFGH 串联
        R[0] = A, R[1] = B, R[2] = C, R[3] = D, R[4] = E, R[5] =
F, R[6] = G, R[7] = H;
        for (int i = 0; i < 8; ++i)
        {
                Vi1[i * 4] = (R[i] >> 24) & 0xFF;
                Vi1[i * 4 + 1] = (R[i] >> 16) & 0xFF;
                Vi1[i * 4 + 2] = (R[i] >> 8) & 0xFF;
                Vi1[i * 4 + 3] = (R[i]) & 0xFF;
        }
        // Vi1 = ABCDEFGH ^ Vi
        for (int i = 0; i < 256 / 8; ++i)
        {
                Vi1[i] ^= Vi[i];
        }
}

//参数 m 是原始数据，ml 是数据长度，r 是输出参数,存放hash结果
void SM3Hash(unsigned char* m, int ml, unsigned char r[32])
{
        int l = ml * 8;
        int k = 448 - 1 - l % 512; // 添加k个0，k 是满足 l + 1 + k
≡ 448mod512 的最小的非负整数
        if (k <= 0)
        {
                k += 512;
        }

        int n = (l + k + 65) / 512;

        int m1l = n * 512 / 8; // 填充后的长度，512位的倍数
```

```
        unsigned char* m1 = new unsigned char[m1l];
        memset(m1, 0, m1l);
        memcpy(m1, m, l / 8);

        m1[l / 8] = 0x80; // 消息后补1

        // 再添加一个64位比特串，该比特串是长度l的二进制表示
        unsigned long l1 = l;
        for (int i = 0; i < 64 / 8 && l1 > 0; ++i)
        {
                m1[m1l - 1 - i] = l1 & 0xFF;
                l1 = l1 >> 8;
        }

        //将填充后的消息m’按512比特进行分组：m’ = B(0)B(1)· · · B(n-1),其中n=(l+k+65)/512。
        unsigned char** B = new unsigned char*[n];
        for (int i = 0; i < n; ++i)
        {
                B[i] = new unsigned char[512 / 8];
                memcpy(B[i], m1 + (512 / 8)*i, 512 / 8);
        }

        delete[] m1;

        unsigned char** V = new unsigned char*[n + 1];
        for (int i = 0; i <= n; ++i)
        {
                V[i] = new unsigned char[256 / 8];
                memset(V[i], 0, 256 / 8);
        }

        // 初始化 V0 = VI
        memcpy(V[0], IV, 256 / 8);

        // 压缩函数，V 与扩展的B
        for (int i = 0; i < n; ++i)
        {
                CF(V[i], B[i], V[i + 1]);
        }

        for (int i = 0; i < n; ++i)
        {
                delete[] B[i];
        }
        delete[] B;

        // V[n]是结果
        memcpy(r, V[n], 32);

        for (int i = 0; i <= n; ++i)
        {
                delete[] V[i];
        }
```

```
        delete[] V;
}

// 打印缓冲区内容
void dumpbuf(unsigned char* buf, int len)
{
        for (int i = 0; i < len; ++i) {
                printf("%02x", buf[i]);
                if ((i + 1) % 16 == 0)
                printf("\n");
                else
                        printf(" ");
        }
        printf("\n");
}

int main()
{
        // 输入数据
        unsigned char message[] = "abc";
        int message_len = strlen((char*)message);

        // 输出缓冲区
        unsigned char hash_result[32] = { 0 };

        // 调用 SM3 哈希函数
        SM3Hash(message, message_len, hash_result);

        // 输出哈希结果
        printf("SM3 Hash:\n");
        dumpbuf(hash_result, 32);

        return 0;
}
```

- 代码编译运行过程与结果

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-1# nano test.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-1# g++ -o test
test.cpp -m32
test.cpp:4:10: fatal error: pch.h: No such file or directory
    4 | #include "pch.h"
      |          ^~~~~~~
compilation terminated.
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-1# nano test.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-1# g++ -o test
test.cpp -m32
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-1# ./test
SM3 Hash:
66 c7 f0 f4 62 ee ed d9 d1 f2 d4 6b dc 10 e4 e2
41 67 c4 87 5c f2 f7 a2 29 7d a0 2b 8f 4b a8 e0
```

- 4-2代码编译运行结果（手工实现三段式SM3算法）

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-2# nano sm3.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-2# ls
sm3.cpp  sm3.h  test  test.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-2# g++ -o test sm3.cpp
test.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-2# ./test
Message: abc
Hash:   37bc43d1 1cab393d 7899ef62 24f568ec 18a8fd85 1d165c50 0c375402
0f466a04
Message:
abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcd
Hash:   a8f95215 08e03054 1325267f d822077a e5c2fd1f 32b54ebb bf8c1c79
064c9b6d
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-2# g++ -o test sm3.cpp
test.cpp -m32
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-2# ./test
Message: abc
Hash:   66c7f0f4 62eeedd9 d1f2d46b dc10e4e2 4167c487 5cf2f7a2 297da02b
8f4ba8e0
Message:
abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcd
Hash:   debe9ff9 2275b8a1 38604889 c18e5a4d 6fdb70e5 387e5765 293dcba3
9c0c5732
```

- 4-3代码编译运行结果（基于openssl实现sm3）

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-3# nano test.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-3# nano sm3hash.h
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-3# nano sm3hash.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-3# g++ -o test test.cpp
sm3hash.cpp -lssl -lcrypto
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-3# ./test
raw data: abc
hash length: 32 bytes.
hash value:
0x66  0xc7  0xf0  0xf4  0x62  0xee  0xed  0xd9  0xd1  0xf2  0xd4  0x6b
0xdc  0x10  0xe4  0xe2  0x41  0x67  0xc4  0x87  0x5c  0xf2  0xf7  0xa2
0x29  0x7d  0xa0  0x2b  0x8f  0x4b  0xa8  0xe0

raw data:
0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64
0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64
0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64
0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64
0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64  0x61  0x62  0x63  0x64
0x61  0x62  0x63  0x64
```

```
hash length: 32 bytes.
hash value:
0xde  0xbe  0x9f  0xf9  0x22  0x75  0xb8  0xa1  0x38  0x60  0x48  0x89
0xc1  0x8e  0x5a  0x4d  0x6f  0xdb  0x70  0xe5  0x38  0x7e  0x57  0x65
0x29  0x3d  0xcb  0xa3  0x9c  0xc  0x57  0x32
```

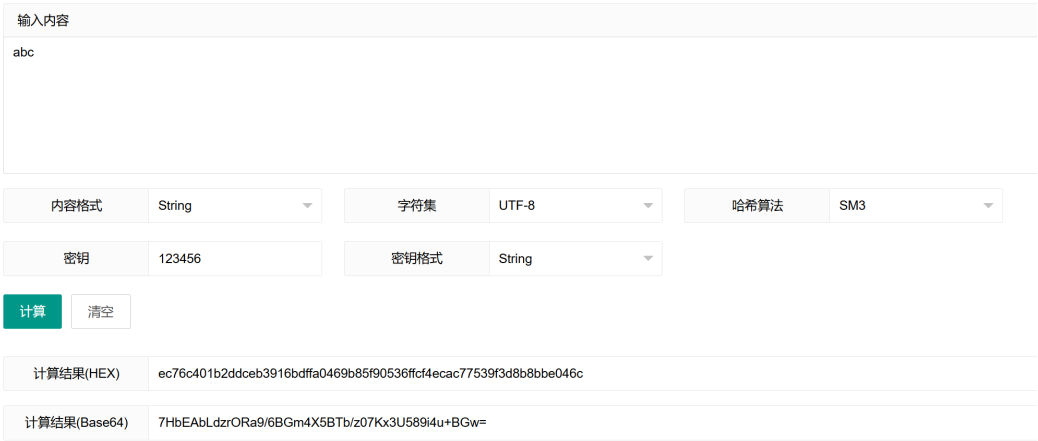
- 4-4代码编译运行结果（实现HMAC-SM3算法）

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-4# ls
sm3.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-4# nano sm3.h
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-4# nano test.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-4# g++ -o test test.cpp
sm3.cpp -m32
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-4# ./test
Message: abc
HMAC:   ec76c401 b2ddceb3 916bdffa 0469b85f 90536ffc f4ecac77 539f3d8b
8bbe046c
```

- 验证部分
  - 使用GmSSL命令得到'abc'和'abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcd'的哈希值

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-3# echo -n "abc" |
gmssl sm3
66c7f0f462eeedd9d1f2d46bdc10e4e24167c4875cf2f7a2297da02b8f4ba8e0
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm3/4-2# echo -n
"abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcd"
| gmssl sm3
debe9ff92275b8a138604889c18e5a4d6fdb70e5387e5765293dcba39c0c5732
```

  - 由于4-4中源代码的密钥长度不符合gmssl命令的要求，所以在在线网站中验证结果



- 调试运行sm4代码
  - 3-6代码编译运行结果（16字节版）

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-6# nano sm4.h
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-6# nano sm4.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-6# nano test.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-6# g++ -o test test.cpp
sm4.cpp -m32
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-6# ./test
sm4(16��)�L�ɹ���

root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-6# nano sm4.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-6# g++ -o test test.cpp
sm4.cpp -m32
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-6# ./test
sm4(16字节)自检成功
```

- 3-7代码编译运行结果（实现SM4-ECBCBCCFBOFB 算法(大数据版)）

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-7# nano sm4.h
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-7# nano sm4.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-7# nano sm4check.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-7# nano test.cpp
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-7# g++ -o test test.cpp
sm4check.cpp sm4.cpp -m32
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm4/3-7# ./test
ecb enc(len=16) memcmp ok
ecb dec(len=16) memcmp ok
ecb enc/dec(len=32) memcmp ok
ecb enc/dec(len=64) memcmp ok
ecb enc/dec(len=128) memcmp ok
ecb enc/dec(len=256) memcmp ok
ecb enc/dec(len=512) memcmp ok
ecb enc/dec(len=1024) memcmp ok
ecb enc/dec(len=2048) memcmp ok
ecb enc/dec(len=4096) memcmp ok
cbc enc(len=32) memcmp ok
cbc dec(len=32) memcmp ok
cbc enc/dec(len=32) memcmp ok
cbc enc/dec(len=64) memcmp ok
cbc enc/dec(len=128) memcmp ok
cbc enc/dec(len=256) memcmp ok
cbc enc/dec(len=512) memcmp ok
cbc enc/dec(len=1024) memcmp ok
cbc enc/dec(len=2048) memcmp ok
cbc enc/dec(len=4096) memcmp ok
cfb enc/dec(len=16) memcmp ok
cfb enc/dec(len=32) memcmp ok
cfb enc/dec(len=64) memcmp ok
cfb enc/dec(len=128) memcmp ok
cfb enc/dec(len=256) memcmp ok
cfb enc/dec(len=512) memcmp ok
```

```
cfb enc/dec(len=1024) memcmp ok
cfb enc/dec(len=2048) memcmp ok
cfb enc/dec(len=4096) memcmp ok
ofb enc/dec(len=16) memcmp ok
ofb enc/dec(len=32) memcmp ok
ofb enc/dec(len=64) memcmp ok
ofb enc/dec(len=128) memcmp ok
ofb enc/dec(len=256) memcmp ok
ofb enc/dec(len=512) memcmp ok
ofb enc/dec(len=1024) memcmp ok
ofb enc/dec(len=2048) memcmp ok
ofb enc/dec(len=4096) memcmp ok
```

    - 验证：32位与64位编译均自检成功
    - 其他：sm4的程序似乎与系统位数无关，编译的结果都能通过自检函数
- 调试运行sm2代码
    - 代码修改
        - 由于之前已经编译后miracl库，所以不再使用源代码中的miracl.h等文件并修改代码中对于库的头文件的引用
        - 修复重复定义报错：在SM2_ENC.c中加入para_p, para_a等变量的定义，在SM2_ENC.h中给原先变量加上extern守卫。14-3的代码也是这样处理的。
        - 修复下载的源代码中的一些乱码
    - 14-2代码编译运行结果(sm2加解密)
        - 编译与运行过程

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-2# gcc -o test kdf.c
SM2_ENC.c test.c -m32 -lmiracl_32
In file included from kdf.c:1:
kdf.c: In function ‘CF’:
kdf.h:21:31: warning: result of ‘2055708042 << 16’ requires 48
bits to represent, but ‘int’ only has 32 bits [-Wshift-overflow=]
21 | #define SM3_rotl32(x,n) (((x) << n) | ((x) >> (32 - n)))
    |                                ^~
kdf.c:89:29: note: in expansion of macro ‘SM3_rotl32’
89 |                         T = SM3_rotl32(SM3_T2, 16);
    |                             ^~~~~~~~~~
test.c: In function ‘main’:
test.c:5:9: warning: implicit declaration of function
‘SM2_ENC_SelfTest’ [-Wimplicit-function-declaration]
    5 |         SM2_ENC_SelfTest();
    |         ^~~~~~~~~~~~~~~~
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-2# ./test
原文
0x65,0x6e,0x63,0x72,0x79,0x70,0x74,0x69,
0x6f,0x6e,0x20,0x73,0x74,0x61,0x6e,0x64,
0x61,0x72,0x64,

密文:
0x4,0xeb,0xfc,0x71,0x8e,0x8d,0x17,0x98,
0x62,0x4,0x32,0x26,0x8e,0x77,0xfe,0xb6,
```

```
0x41,0x5e,0x2e,0xde,0xe,0x7,0x3c,0xf,
0x4f,0x64,0xe,0xcd,0x2e,0x14,0x9a,0x73,
0xe8,0x58,0xf9,0xd8,0x1e,0x54,0x30,0xa5,
0x7b,0x36,0xda,0xab,0x8f,0x95,0xa,0x3c,
0x64,0xe6,0xee,0x6a,0x63,0x9,0x4d,0x99,
0x28,0x3a,0xff,0x76,0x7e,0x12,0x4d,0xf0,
0x59,0x98,0x3c,0x18,0xf8,0x9,0xe2,0x62,
0x92,0x3c,0x53,0xae,0xc2,0x95,0xd3,0x3,
0x83,0xb5,0x4e,0x39,0xd6,0x9,0xd1,0x60,
0xaf,0xcb,0x19,0x8,0xd0,0xbd,0x87,0x66,
0x21,0x88,0x6c,0xa9,0x89,0xca,0x9c,0x7d,
0x58,0x8,0x73,0x7,0xca,0x93,0x9,0x2d,
0x65,0x1e,0xfa,

解密结果:
0x65,0x6e,0x63,0x72,0x79,0x70,0x74,0x69,
0x6f,0x6e,0x20,0x73,0x74,0x61,0x6e,0x64,
0x61,0x72,0x64,
解密成功
```

- 14-3代码编译运行结果(sm2签名验签)
  - 编译与运行结果

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# nano SM2_sv.h
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# nano SM2_sv.c
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# nano kdf.h
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# nano kdf.c
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# nano test.c
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# gcc -o test
test.c SM2_sv.c kdf.c -lmiracl_32 -m32
SM2_sv.c:3:10: fatal error: KDF.h: No such file or directory
    3 | #include "KDF.h"
      |          ^~~~~~
compilation terminated.
In file included from kdf.c:1:
kdf.c: In function ‘CF’:
kdf.h:22:31: warning: result of ‘2055708042 << 16’ requires 48
bits to represent, but ‘int’ only has 32 bits [-Wshift-overflow=]
22 | #define SM3_rotl32(x,n) (((x) << n) | ((x) >> (32 - n)))
      |                               ^~
kdf.c:92:29: note: in expansion of macro ‘SM3_rotl32’
92 |                         T = SM3_rotl32(SM3_T2, 16);
      |                             ^~~~~~~~~~
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# nano SM2_sv.c
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# gcc -o test
test.c SM2_sv.c kdf.c -lmiracl_32 -m32
In file included from kdf.c:1:
kdf.c: In function ‘CF’:
kdf.h:22:31: warning: result of ‘2055708042 << 16’ requires 48
bits to represent, but ‘int’ only has 32 bits [-Wshift-overflow=]
22 | #define SM3_rotl32(x,n) (((x) << n) | ((x) >> (32 - n)))
```

```
    |                                ^~
kdf.c:92:29: note: in expansion of macro ‘SM3_rotl32’
92 |                         T = SM3_rotl32(SM3_T2, 16);
    |                             ^~~~~~~~~~
/usr/bin/ld: /tmp/cc3Q7CVn.o:(.bss+0x0): multiple definition of
`Gx'; /tmp/ccu5rqId.o:(.bss+0x0): first defined here
/usr/bin/ld: /tmp/cc3Q7CVn.o:(.bss+0x4): multiple definition of
`Gy'; /tmp/ccu5rqId.o:(.bss+0x4): first defined here
/usr/bin/ld: /tmp/cc3Q7CVn.o:(.bss+0x8): multiple definition of
`p'; /tmp/ccu5rqId.o:(.bss+0x8): first defined here
/usr/bin/ld: /tmp/cc3Q7CVn.o:(.bss+0xc): multiple definition of
`a'; /tmp/ccu5rqId.o:(.bss+0xc): first defined here
/usr/bin/ld: /tmp/cc3Q7CVn.o:(.bss+0x10): multiple definition of
`b'; /tmp/ccu5rqId.o:(.bss+0x10): first defined here
/usr/bin/ld: /tmp/cc3Q7CVn.o:(.bss+0x14): multiple definition of
`n'; /tmp/ccu5rqId.o:(.bss+0x14): first defined here
/usr/bin/ld: /tmp/cc3Q7CVn.o:(.bss+0x18): multiple definition of
`G'; /tmp/ccu5rqId.o:(.bss+0x18): first defined here
/usr/bin/ld: /tmp/cc3Q7CVn.o:(.bss+0x1c): multiple definition of
`nG'; /tmp/ccu5rqId.o:(.bss+0x1c): first defined here
/usr/bin/ld: /tmp/ccXUWcaA.o:(.bss+0x0): multiple definition of
`Gx'; /tmp/ccu5rqId.o:(.bss+0x0): first defined here
/usr/bin/ld: /tmp/ccXUWcaA.o:(.bss+0x4): multiple definition of
`Gy'; /tmp/ccu5rqId.o:(.bss+0x4): first defined here
/usr/bin/ld: /tmp/ccXUWcaA.o:(.bss+0x8): multiple definition of
`p'; /tmp/ccu5rqId.o:(.bss+0x8): first defined here
/usr/bin/ld: /tmp/ccXUWcaA.o:(.bss+0xc): multiple definition of
`a'; /tmp/ccu5rqId.o:(.bss+0xc): first defined here
/usr/bin/ld: /tmp/ccXUWcaA.o:(.bss+0x10): multiple definition of
`b'; /tmp/ccu5rqId.o:(.bss+0x10): first defined here
/usr/bin/ld: /tmp/ccXUWcaA.o:(.bss+0x14): multiple definition of
`n'; /tmp/ccu5rqId.o:(.bss+0x14): first defined here
/usr/bin/ld: /tmp/ccXUWcaA.o:(.bss+0x18): multiple definition of
`G'; /tmp/ccu5rqId.o:(.bss+0x18): first defined here
/usr/bin/ld: /tmp/ccXUWcaA.o:(.bss+0x1c): multiple definition of
`nG'; /tmp/ccu5rqId.o:(.bss+0x1c): first defined here
collect2: error: ld returned 1 exit status
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# rm SM2_sv.c
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# nano SM2_sv.c
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# rm SM2_sv.h
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# nano SM2_sv.h
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# gcc -o test
test.c SM2_sv.c kdf.c -lmiracl_32 -m32
In file included from kdf.c:1:
kdf.c: In function ‘CF’:
kdf.h:22:31: warning: result of ‘2055708042 << 16’ requires 48
bits to represent, but ‘int’ only has 32 bits [-Wshift-overflow=]
22 | #define SM3_rotl32(x,n) (((x) << n) | ((x) >> (32 - n)))
    |                                ^~
kdf.c:92:29: note: in expansion of macro ‘SM3_rotl32’
92 |                         T = SM3_rotl32(SM3_T2, 16);
    |                             ^~~~~~~~~~
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-3# ./test
SM2 签名验签成功
```

- 验证：自检均通过，无需用命令验证

2. 在密标委网站http://www.gmbz.org.cn/main/bzlb.html查找SM2，SM3，SM4相关标准，分析代码实现与标准的对应关系。（10分）

- 相关标准文件链接
  - SM3： SM3 密码杂凑算法
  - SM4: SM4分组密码算法
  - SM2: 五个标准文件如下所示
    - SM2 椭圆曲线公钥密码算法第1部分：总则
    - SM2 椭圆曲线公钥密码算法第2部分：数字签名算法
    - SM2 椭圆曲线公钥密码算法第3部分：密钥交换协议
    - SM2 椭圆曲线公钥密码算法第4部分：公钥加密算法
    - SM2 椭圆曲线公钥密码算法第5部分：参数定义
- 分析代码实现与标准的对应关系
  - SM3
    1. 上下文结构

```
标准定义：SM3 的上下文结构包含状态、已处理的比特总数和数据缓冲区。
实现对应：sm3_context 结构体符合标准定义，包含 total（已处理比特数）、state（当前哈希状态）和 buffer（用于存储输入数据的缓冲区）。
```

    2. 初始化

```
标准定义：SM3 的初始化状态为特定的常数值。
实现对应：sm3_starts 函数设置了与标准一致的初始状态值。
```

    3. 消息扩展

```
标准定义：SM3 规定了如何扩展输入消息为 68 个字（64 个输入字 + 4 个扩展字）。
实现对应：sm3_process 函数中的消息扩展逻辑，使用公式生成 W 数组与 W1 数组，符合标准的扩展要求。
```

    4. 压缩函数

```
标准定义：SM3 使用特定的逻辑运算和常量进行压缩。
实现对应：FF0, FF1, GG0, GG1 等宏的定义和使用符合标准的压缩函数逻辑，计算状态更新的步骤也遵循标准的流程。
```

    5. 填充和结束

标准定义：SM3 在处理完输入后，需要进行填充以确保数据长度为 512 位的倍数，并处理最后的状态输出。
实现对应：sm3_update 和 sm3_finish 函数处理了输入填充和最终哈希值的输出，确保符合标准的要求。

6. 输出

标准定义：SM3 输出一个 256 位（32 字节）的哈希值。
实现对应：sm3_finish 函数输出 32 字节的结果，符合标准输出格式。

7. 文件处理

标准定义：处理文件的能力以支持大数据量输入。
实现对应：sm3_file 函数提供了从文件读取数据并进行哈希计算的功能，符合标准的实际应用需求。

- SM4

1. 常量定义

实现:SM4_CK 和 SM4_FK 数组。
对应标准:SM4 标准规定了轮常量（CK）和初始轮密钥常量（FK），这些常量在加密过程中是必需的。实现中定义的数组直接使用标准中的常量值，确保一致性。

2. S-Box

实现：SM4_Sbox 数组，包含 256 字节的替代值。
对应标准:SM4 标准明确规定了 S-Box 的内容，用于进行字节的非线性替换。实现中的 S-Box 数组严格遵循标准中的映射。

3. 密钥调度

实现:SM4_KeySchedule 函数。
对应标准:SM4 标准描述了如何从128位的初始密钥生成32个轮密钥。实现中使用标准中指定的算法和常量（CK 和 FK），按照标准的流程生成轮密钥。

4. 加密过程

实现:SM4_Encrypt 函数。
对应标准:SM4 标准中描述了加密的具体流程，包括明文分为4个字块，经过32轮处

理。实现中严格按照标准规定的步骤和顺序进行位运算、S-Box 替换和轮密钥加。

5. 解密过程

> 实现:SM4_Decrypt 函数。
> 对应标准:SM4 标准中说明了解密过程与加密相反，使用轮密钥的逆序（rk[31 - i]）。实现中遵循这一点，确保解密流程与标准一致。

6. 循环位移

> 实现:SM4_Rotl32 宏。
> 对应标准:SM4 标准中要求对数据进行循环左移操作以实现位变换。实现中的函数通过位操作实现这一要求，确保符合标准。

7. 数据格式和输出

> 实现:加密和解密函数均处理 128 位（16 字节）数据块。
> 对应标准:SM4 标准规定了块加密的要求，输入和输出均为128位数据块。实现中严格按照这一格式处理数据。

- SM2

1. 椭圆曲线参数设置与初始化（对应标准中椭圆曲线参数定义部分）
   - **代码部分**：
     - 在SM2_Init函数中，通过bytes_to_big函数将预定义的字节数组形式的椭圆曲线参数（如SM2_p、SM2_a、SM2_b、SM2_n、SM2_Gx、SM2_Gy、SM2_h）转换为big类型，并初始化椭圆曲线相关对象。例如：

```
bytes_to_big(SM2_NUMWORD, SM2_p, para_p);
// 其他参数类似的转换和初始化操作

ecurve_init(para_a, para_b, para_p, MR_PROJECTIVE);
if (!epoint_set(para_Gx, para_Gy, 0, G))
{
    return ERR_ECURVE_INIT;
}
```

     - 还通过ecurve_mult等操作来验证点G的阶是否为n。
     - **标准对应**：在SM2标准中，首先需要明确定义椭圆曲线的参数，包括有限域的特征p、曲线方程系数a、b，基点G的坐标Gx、Gy以及群的阶n和余因子h等。这里的代码操作就是在程序中按照标准要求准确设置和初始化这些关键的椭圆曲线参数及相关对象，以构建符合SM2标准的椭圆曲线环境。
2. 公钥生成（对应标准中公钥生成流程部分）

- **代码部分**：
  - SM2_KeyGeneration函数实现了从私钥生成公钥的功能。它接受一个私钥（big类型的priKey），通过ecurve_mult操作计算公钥（pubKey = [priKey]G），然后调用Test_PubKey函数来验证生成的公钥是否符合要求。

```
int SM2_KeyGeneration(big priKey, epoint *pubKey)
{
    ecurve_mult(priKey, G, pubKey);
    if (Test_PubKey(pubKey)!= 0)
        return 1;
    else
        return 0;
}
```

- **标准对应**：按照SM2标准，公钥是通过私钥与基点G进行椭圆曲线点乘运算得到的，并且生成的公钥需要满足一系列的有效性条件，如不能是无穷远点、坐标需在有限域内、在椭圆曲线上且阶符合要求等。这里的代码通过调用相关函数准确地实现了标准规定的公钥生成流程及后续的有效性验证步骤。

3. 加密过程（对应标准中加密算法流程部分）
   - **代码部分**：
     - 在SM2_Encrypt函数中，按照以下步骤进行加密操作：
       - 步骤 1：首先从密文C中提取C1的坐标并转换为big类型，然后通过Test_Point函数验证C1是否在椭圆曲线上。

```
bytes_to_big(SM2_NUMWORD, C, C1x);
bytes_to_big(SM2_NUMWORD, C + SM2_NUMWORD, C1x);
epoint_set(C1x, C1y, 0, C1);
i = Test_Point(C1);
if (i!= 0)
    return i;
```

       - 步骤2：计算C1 = [k]G，通过bytes_to_big将随机数randK转换为big类型的rand，然后利用ecurve_mult计算得到C1，并将其坐标转换为字节数组存入密文C的相应位置。

```
bytes_to_big(SM2_NUMWORD, randK, rand);
ecurve_mult(rand, G, C1);
epoint_get(C1, C1x, C1y);
big_to_bytes(SM2_NUMWORD, C1x, C, 1);
big_to_bytes(SM2_NUMWORD, C1y, C + SM2_NUMWORD,
1);
```

       - 步骤3：计算S = [h]pubKey并检查是否为无穷远点。

```
ecurve_mult(para_h, pubKey, S);
if (point_at_infinity(S))
    return ERR_INFINITY_POINT;
```

- 步骤4：计算`[k]PB = (x2, y2)`，同样使用`ecurve_mult`操作。

```
ecurve_mult(rand, pubKey, kP);
epoint_get(kP, x2, y2);
```

- 步骤5：通过`KDF`函数（这里是`SM3_KDF`）基于`x2`和`y2`生成密钥材料，并检查是否全为零。

```
big_to_bytes(SM2_NUMWORD, x2, x2y2, 1);
big_to_bytes(SM2_NUMWORD, y2, x2y2 + SM2_NUMWORD,
1);
SM3_KDF(x2y2, SM2_NUMWORD * 2, klen, C +
SM2_NUMWORD * 3);
if (Test_Null(C + SM2_NUMWORD * 3, klen)!= 0)
    return ERR_ARRAY_NULL;
```

- 步骤6：计算`C2 = M ^ t`，通过逐字节异或操作实现。

```
for (i = 0; i < klen; i++)
{
    C[SM2_NUMWORD * 3 + i] = M[i] ^ C[SM2_NUMWORD
* 3 + i];
}
```

- 步骤7：计算`C3 = hash(x2, M, y2)`，通过`SM3`相关函数（`SM3_init`、`SM3_process`、`SM3_done`）完成哈希计算并将结果存入密文`C`的相应位置。

```
SM3_init(&md);
SM3_process(&md, x2y2, SM2_NUMWORD);
SM3_process(&md, M, klen);
SM3_process(&md, x2y2 + SM2_NUMWORD, SM2_NUMWORD);
SM3_done(&md, C + SM2_NUMWORD * 2);
```

- **标准对应**：SM2标准的加密过程规定了一系列明确的步骤，包括生成随机数、计算密文的各个组成部分（如`C1`、`C2`、`C3`），其中涉及到椭圆曲线点乘、密钥派生函数（KDF）应用以及哈希函数计算等操作。上述代码中的各个步骤严格按照标准所规定的加密流程来实现，确保了加密操作符合SM2标准的要求。

4. 解密过程（对应标准中解密算法流程部分）
    - **代码部分**：
        - 在SM2_Decrypt函数中，执行以下解密步骤：
            - 步骤1：首先从密文C中提取C1的坐标并转换为big类型，然后通过Test_Point函数验证C1是否在椭圆曲线上。

```
bytes_to_big(SM2_NUMWORD, C, C1x);
bytes_to_big(SM2_NUMWORD, C + SM2_NUMWORD, C1x);
epoint_set(C1x, C1y, 0, C1);
i = Test_Point(C1);
if (i!= 0)
    return i;
```

            - 步骤2：计算S = [h]C1并检查是否为无穷远点。

```
ecurve_mult(para_h, C1, S);
if (point_at_infinity(S))
    return ERR_INFINITY_POINT;
```

            - 步骤3：计算[dB]C1 = (x2, y2)，使用ecurve_mult操作，然后将坐标转换为字节数组用于后续操作。

```
ecurve_mult(dB, C1, dBC1);
epoint_get(dBC1, x2, y2);
big_to_bytes(SM2_NUMWORD, x2, x2y2, 1);
big_to_bytes(SM2_NUMWORD, y2, x2y2 + SM2_NUMWORD,
1);
```

            - 步骤4：通过KDF函数（SM3_KDF）基于x2和y2生成密钥材料，并检查是否全为零。

```
SM3_KDF(x2y2, SM2_NUMWORD * 2, Clen - SM2_NUMWORD
* 3, M);
if (Test_Null(M, Clen - SM2_NUMWORD * 3)!= 0)
    return ERR_ARRAY_NULL;
```

            - 步骤5：计算M = C2 ^ t，通过逐字节异或操作还原出明文。

```
for (i = 0; i < Clen - SM2_NUMWORD * 3; i++)
    M[i] = M[i] ^ C[SM2_NUMWORD * 3 + i];
```

- 步骤6：计算`hash(x2, m, y2)`并与密文C中的`C3`部分进行比较，以验证解密的正确性。

```
SM3_init(&md);
SM3_process(&md, x2y2, SM2_NUMWORD);
SM3_process(&md, M, Clen - SM2_NUMWORD * 3);
SM3_process(&md, x2y2 + SM2_NUMWORD, SM2_NUMWORD);
SM3_done(&md, hash);
if (memcmp(hash, C + SM2_NUMWORD * 2,
SM2_NUMWORD)!= 0)
    return ERR_C3_MATCH;
else
    return 0;
```

- **标准对应**：SM2标准的解密过程同样有明确规定的步骤，包括从密文中提取相关信息、验证密文组件的有效性、通过椭圆曲线点乘还原出中间值、利用KDF生成密钥材料以及通过哈希验证来确保解密结果的正确性等。代码中的这些步骤紧密遵循标准所设定的解密流程，实现了符合SM2标准要求的解密操作。

5. 自测试功能（对应标准中验证实现正确性的需求部分）
   - **代码部分**：
     - `SM2_ENC_SelfTest`函数实现了自测试功能。它首先初始化`MIRACL`系统相关设置，然后按照以下步骤进行测试：
       - 读取标准的私钥、公钥、随机数、明文和密文等数据作为测试用例。
       - 调用`SM2_Init`函数初始化椭圆曲线环境。
       - 通过`SM2_KeyGeneration`函数基于标准私钥生成公钥，并与标准公钥进行比较验证公钥生成的正确性。
       - 使用`SM2_Encrypt`函数基于标准随机数、生成的公钥和标准明文进行加密操作，并将结果与标准密文进行比较验证加密的正确性。
       - 最后通过`SM2_Decrypt`函数基于标准私钥和生成的密文进行解密操作，并将结果与标准明文进行比较验证解密的正确性。
   - **标准对应**：这里的自测试函数数据与标准文件附录一致，符合SM2标准对于功能正确性验证的相关要求。

3. 使用GmSSL,UKey交叉验证实现的正确性（5 分）

4. 实验记录中提交 gitee 课程项目链接，提交本次实验相关 git log运行结果

- gittee链接
- git log记录：

```
root@Youer:~/shiyan/shiyan02/shiyan2-2# git log
commit b1564294aacd5b20bbef42d34d437b99d8cd1064 (HEAD -> master,
origin/master, origin/HEAD)
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Thu Oct 31 18:08:30 2024 +0800

    shiyan2-2:finish sm2 14-3
```

```
commit d5529c2b1d6301e4dee7dae3f2cfa85368da802d
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Thu Oct 31 17:59:36 2024 +0800

    shiyan2-2: finish sm2 14-2

commit d3a72a28d5ea164ac67f951e0fd456cd2a6f7c55
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Wed Oct 30 22:41:32 2024 +0800

    shiyan2-2:fail to sm2 14-2

commit 8b984d586697f9b76a0284d9d0f1984630a6bf84
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Wed Oct 30 22:25:58 2024 +0800

    shiyan2-2:finish sm4 3-6 & 3-7

commit 79d752afc6ba9256ed585ef34cdf24863ff14329
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Wed Oct 30 21:25:29 2024 +0800

    shiyan2-2:finish sm3 4-4

commit fafdcfa790230f9715233b8e0249b182d4edf72d
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Wed Oct 30 20:58:05 2024 +0800

    shiyan2-2: finish 4-1

commit 94aca6e5831369d0cdf4a096b959e0f2e4df6ec0
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Wed Oct 30 14:56:42 2024 +0800

    shiyan2-2:sm3 4-2 & 4-3
```

5. 提交要求：

- 提交实践过程Markdown和转化的PDF文件
- 代码，文档托管到gitee或github等，推荐 gitclone
- 记录实验过程中遇到的问题，解决过程，反思等内容，用于后面实验报告
- 实验中遇到的问题
    - sm3的教材代码结果不正确
        - 现象：'abc'的sm3哈希值为37bc43d1 1cab393d 7899ef62 24f568ec 18a8fd85 1d165c50 0c375402 0f466a04
        - 原因是程序需要32位环境运行
        - 同样的问题

- 如何判断代码是32位还是64位抑或是无关
  - 在这个sm3算法程序中，其位操作和位移的界限是32位的，这是一个很明确的信号

- SM2编译报错
  - 现象：在SM2_ENC.h中为变量加了extern后会出现无定义报错，不加就会出现重复定义报错，非常神奇。签名验签中也有类似问题。
    - 重复定义报错：

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-2# gcc -o test
kdf.c SM2_ENC.c test.c -m32 -lmiracl_32
In file included from kdf.c:1:
kdf.c: In function ‘CF’:
kdf.h:21:31: warning: result of ‘2055708042 << 16’ requires
48 bits to represent, but ‘int’ only has 32 bits [-Wshift-
overflow=]
21 | #define SM3_rotl32(x,n) (((x) << n) | ((x) >> (32 - n)))
|                                ^~
kdf.c:89:29: note: in expansion of macro ‘SM3_rotl32’
89 |                         T = SM3_rotl32(SM3_T2, 16);
|                              ^~~~~~~~~~
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0x0): multiple definition
of `para_p'; /tmp/cccW9LVJ.o:(.bss+0x0): first defined here
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0x4): multiple definition
of `para_a'; /tmp/cccW9LVJ.o:(.bss+0x4): first defined here
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0x8): multiple definition
of `para_b'; /tmp/cccW9LVJ.o:(.bss+0x8): first defined here
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0xc): multiple definition
of `para_n'; /tmp/cccW9LVJ.o:(.bss+0xc): first defined here
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0x10): multiple definition
of `para_Gx'; /tmp/cccW9LVJ.o:(.bss+0x10): first defined here
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0x14): multiple definition
of `para_Gy'; /tmp/cccW9LVJ.o:(.bss+0x14): first defined here
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0x18): multiple definition
of `para_h'; /tmp/cccW9LVJ.o:(.bss+0x18): first defined here
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0x1c): multiple definition
of `G'; /tmp/cccW9LVJ.o:(.bss+0x1c): first defined here
/usr/bin/ld: /tmp/ccOFEzUh.o:(.bss+0x20): multiple definition
of `mip'; /tmp/cccW9LVJ.o:(.bss+0x20): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0x0): multiple definition
of `para_p'; /tmp/cccW9LVJ.o:(.bss+0x0): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0x4): multiple definition
of `para_a'; /tmp/cccW9LVJ.o:(.bss+0x4): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0x8): multiple definition
of `para_b'; /tmp/cccW9LVJ.o:(.bss+0x8): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0xc): multiple definition
of `para_n'; /tmp/cccW9LVJ.o:(.bss+0xc): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0x10): multiple definition
of `para_Gx'; /tmp/cccW9LVJ.o:(.bss+0x10): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0x14): multiple definition
of `para_Gy'; /tmp/cccW9LVJ.o:(.bss+0x14): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0x18): multiple definition
of `para_h'; /tmp/cccW9LVJ.o:(.bss+0x18): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0x1c): multiple definition
```

```
of `G'; /tmp/cccW9LVJ.o:(.bss+0x1c): first defined here
/usr/bin/ld: /tmp/ccZEitAg.o:(.bss+0x20): multiple definition
of `mip'; /tmp/cccW9LVJ.o:(.bss+0x20): first defined here
collect2: error: ld returned 1 exit status
```

- 无定义报错：

```
root@Youer:~/shiyan/shiyan02/shiyan2-2/sm2/14-2# gcc -o test
kdf.c SM2_ENC.c test.c -m32 -lmiracl_32
In file included from kdf.c:1:
kdf.c: In function ‘CF’:
kdf.h:21:31: warning: result of ‘2055708042 << 16’ requires
48 bits to represent, but ‘int’ only has 32 bits [-Wshift-
overflow=]
21 | #define SM3_rotl32(x,n) (((x) << n) | ((x) >> (32 - n)))
|                                ^~
kdf.c:89:29: note: in expansion of macro ‘SM3_rotl32’
89 |                          T = SM3_rotl32(SM3_T2, 16);
|                           ^~~~~~~~~~
/usr/bin/ld: /tmp/cczBODut.o: in function `Test_Point':
SM2_ENC.c:(.text+0x68): undefined reference to `para_p'
/usr/bin/ld: SM2_ENC.c:(.text+0x81): undefined reference to
`para_a'
/usr/bin/ld: SM2_ENC.c:(.text+0x9b): undefined reference to
`para_p'
/usr/bin/ld: SM2_ENC.c:(.text+0xc9): undefined reference to
`para_b'
/usr/bin/ld: SM2_ENC.c:(.text+0xe3): undefined reference to
`para_p'
/usr/bin/ld: SM2_ENC.c:(.text+0xfd): undefined reference to
`para_p'
/usr/bin/ld: /tmp/cczBODut.o: in function `Test_PubKey':
SM2_ENC.c:(.text+0x1c6): undefined reference to `para_p'
/usr/bin/ld: SM2_ENC.c:(.text+0x1e2): undefined reference to
`para_p'
/usr/bin/ld: SM2_ENC.c:(.text+0x21e): undefined reference to
`para_n'
/usr/bin/ld: /tmp/cczBODut.o: in function `SM2_Init':
SM2_ENC.c:(.text+0x2c3): undefined reference to `para_p'
/usr/bin/ld: SM2_ENC.c:(.text+0x2d8): undefined reference to
`para_a'
/usr/bin/ld: SM2_ENC.c:(.text+0x2ed): undefined reference to
`para_b'
/usr/bin/ld: SM2_ENC.c:(.text+0x302): undefined reference to
`para_n'
/usr/bin/ld: SM2_ENC.c:(.text+0x317): undefined reference to
`para_Gx'
/usr/bin/ld: SM2_ENC.c:(.text+0x32c): undefined reference to
`para_Gy'
/usr/bin/ld: SM2_ENC.c:(.text+0x341): undefined reference to
`para_h'
/usr/bin/ld: SM2_ENC.c:(.text+0x34e): undefined reference to
```

```
`G'
/usr/bin/ld: SM2_ENC.c:(.text+0x35e): undefined reference to
`para_p'
/usr/bin/ld: SM2_ENC.c:(.text+0x37b): undefined reference to
`para_a'
/usr/bin/ld: SM2_ENC.c:(.text+0x398): undefined reference to
`para_b'
/usr/bin/ld: SM2_ENC.c:(.text+0x3b5): undefined reference to
`para_n'
/usr/bin/ld: SM2_ENC.c:(.text+0x3d2): undefined reference to
`para_Gx'
/usr/bin/ld: SM2_ENC.c:(.text+0x3ef): undefined reference to
`para_Gy'
/usr/bin/ld: SM2_ENC.c:(.text+0x40c): undefined reference to
`para_h'
/usr/bin/ld: SM2_ENC.c:(.text+0x429): undefined reference to
`para_p'
/usr/bin/ld: SM2_ENC.c:(.text+0x431): undefined reference to
`para_b'
/usr/bin/ld: SM2_ENC.c:(.text+0x439): undefined reference to
`para_a'
/usr/bin/ld: SM2_ENC.c:(.text+0x44e): undefined reference to
`G'
/usr/bin/ld: SM2_ENC.c:(.text+0x456): undefined reference to
`para_Gy'
/usr/bin/ld: SM2_ENC.c:(.text+0x45e): undefined reference to
`para_Gx'
/usr/bin/ld: SM2_ENC.c:(.text+0x47e): undefined reference to
`G'
/usr/bin/ld: SM2_ENC.c:(.text+0x486): undefined reference to
`para_n'
/usr/bin/ld: /tmp/cczBODut.o: in function
`SM2_KeyGeneration':
SM2_ENC.c:(.text+0x4fa): undefined reference to `G'
/usr/bin/ld: /tmp/cczBODut.o: in function `SM2_Encrypt':
SM2_ENC.c:(.text+0x6a3): undefined reference to `G'
/usr/bin/ld: SM2_ENC.c:(.text+0x714): undefined reference to
`para_h'
/usr/bin/ld: /tmp/cczBODut.o: in function `SM2_Decrypt':
SM2_ENC.c:(.text+0xace): undefined reference to `para_h'
/usr/bin/ld: /tmp/cczBODut.o: in function `SM2_ENC_SelfTest':
SM2_ENC.c:(.text+0xfd6): undefined reference to `mip'
/usr/bin/ld: SM2_ENC.c:(.text+0xfde): undefined reference to
`mip'
collect2: error: ld returned 1 exit status
```

- 解决过程与原理解释:与kimi的对话
- 解决方法:
  - 在SM2_ENC.c中加入变量的定义

```
// 定义全局变量
big para_p, para_a, para_b, para_n, para_Gx, para_Gy, para_h;
epoint *G;
miracl *mip;
```

- 在SM2_ENC.h中给原先变量加上extern守卫

```
extern big para_p, para_a, para_b, para_n, para_Gx, para_Gy,
para_h;
extern epoint *G;
extern miracl *mip;
```

- 签名验证的SM2_sv.h和SM2_sv.c也是同样的处理方法