

# Implement a PGP scheme with SM2

赵嵘晖 202100460100

## 1 实验环境

编辑器: Visual Studio Code

操作系统: Windows 11

编译语言: Python 3.10

CPU: 12th Gen Intel(R) Core(TM) i5-12500H 2.50 GHz

## 2 SM2 密钥协商

### 2.1 原理

系统参数:

- 1、域  $F_q$  的描述
- 2、椭圆曲线的两个定义元  $a, b \in F_q$
- 3、 $E(F_q)$  的基点  $G(G_x, G_y) \neq O$ , 其中  $G_x, G_y \in F_q$
- 4、 $G$  的阶  $n$ ,  $[n]G = O$
- 5、其他可选项, 如  $n$  的余因子  $h: = \frac{\#E(F_q)}{n}$
- 6、用户 A 的密钥对: 私钥  $d_A$ , 公钥  $P_A = [d_A]G = (x_A, y_A)$
- 7、用户 B 的密钥对: 私钥  $d_B$ , 公钥  $P_B = [d_B]G = (x_B, y_B)$
- 8、用户 A 的可辨别标识  $ID_A$ , 长度  $\text{Elen}_A$  比特
- 9、用户 B 的可辨别标识  $ID_B$ , 长度  $\text{Elen}_B$  比特
- 10、 $Z_A = H_{256}(\text{Elen}_A || ID_A || a || b || G_x || G_y || x_A || y_A)$
- 11、 $Z_B = H_{256}(\text{Elen}_B || ID_B || a || b || G_x || G_y || x_B || y_B)$
- 12、 $w = \lceil \frac{\lceil \log_2 n \rceil}{2} \rceil - 1$

主要函数:

- 1、密钥派生函数 KDF
- 2、伪随机数生成器 PRG
- 3、杂凑函数 SM3

流程图如下所示。

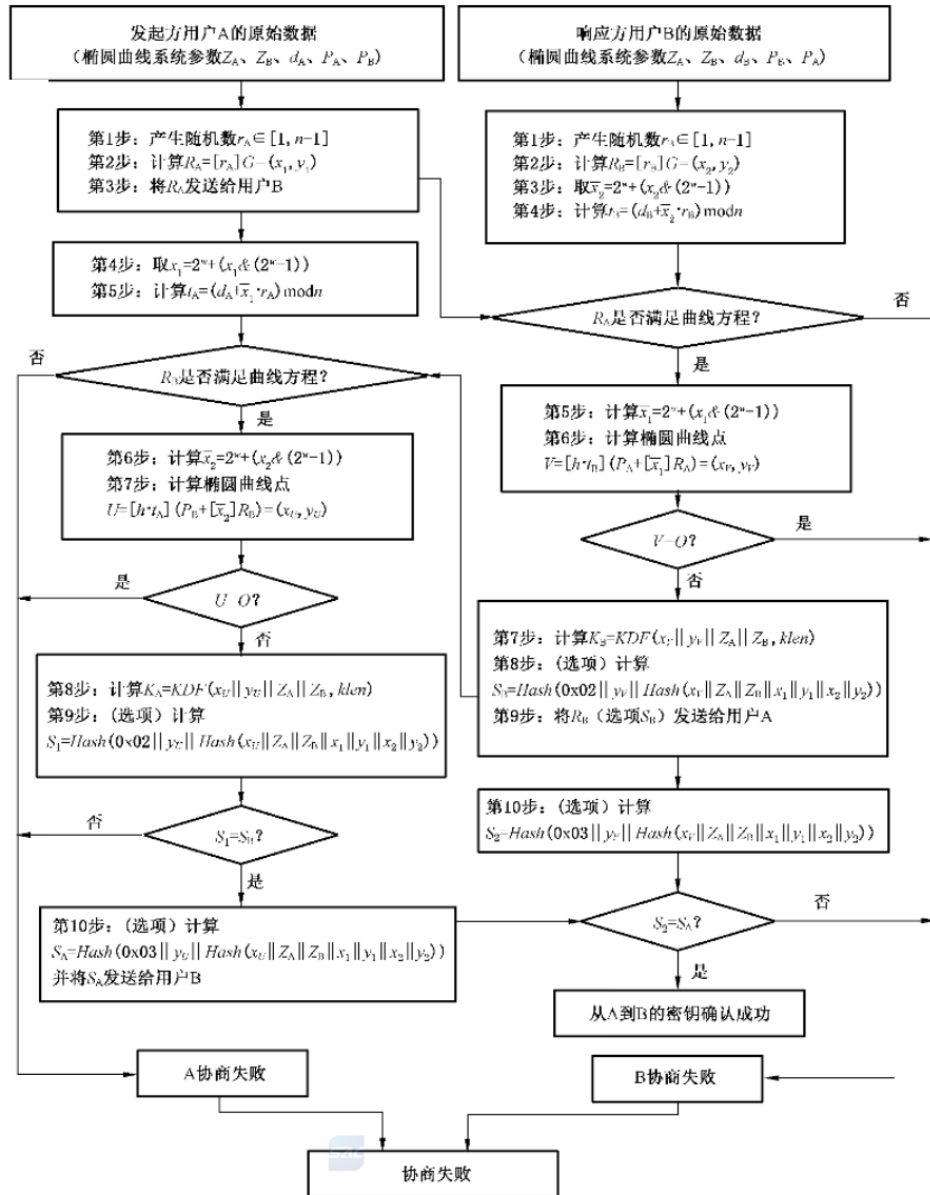


图 1: SM2 密钥协商算法

## 2.2 实现

SM2 密钥协商有两个用户 A 和 B。

其中 A 先随机选取  $r_A$ ,  $r_A \in [1, n-1]$ 。A 计算椭圆曲线上的点  $R_A = [r_A]G = (x_1, y_1)$ 。A 将  $R_A$  发给 B。

B 先随机选取  $r_B$ ,  $r_B \in [1, n-1]$ 。B 计算椭圆曲线上的点  $R_B = [r_B]G = (x_2, y_2)$ 。B 计算  $x_{B2} = 2^w + x_2 \& (2^w - 1)$ ,  $t_B = d_B + x_{B2} \cdot r_B \bmod n$ 。

B 判断  $R_A$  是否在椭圆曲线上, 如果不在那么协商失败。如果在那么 B 计算  $x_{A1} = 2^w + x_1 \& (2^w - 1)$ 。计算椭圆曲线上的点  $V = [h \cdot t_B](P_A + [x_{A1}]R_A) = (x_V, y_V)$ 。B 计算  $K_B = KDF(x_V || y_V || Z_A || Z_B, klen)$ , 其中  $klen$  是协商密钥的比特长度。B 再计算  $S_B = Hash(0X02 || y_V || Hash(x_V || Z_A || Z_B || x_1 || y_1 || x_2 || y_2))$ 。B 将  $R_B$  和  $S_B$  发送给 A。

A 收到后先计算  $x_{A1} = 2^w + x_1 \& (2^w - 1)$ ,  $t_A = d_A + x_{A1} \cdot r_A \bmod n$  并判断  $R_B$  是否在椭圆曲线上, 如果不在那么协商失败。

如果在那么 A 计算  $x_{B2} = 2^w + x_2 \& (2^w - 1)$ 。计算椭圆曲线上的点  $U = [h \cdot t_A](P_B + [x_{B2}]R_B) = (x_U, y_U)$ 。A 再计算  $S_1 = Hash(0X02 || y_U || Hash(x_U || Z_A || Z_B || x_1 || y_1 || x_2 || y_2))$ 。A 判断  $S_1$  是否与  $S_B$  相等, 如果不相等那么协商失败。

如果相等, A 计算  $K_A = KDF(x_U || y_U || Z_A || Z_B, klen)$ , 其中  $klen$  是协商密钥的比特长度。A 再计算  $S_A = Hash(0X03 || y_U || Hash(x_U || Z_A || Z_B || x_1 || y_1 || x_2 || y_2))$ 。A 将  $S_A$  发送给 B。

B 计算  $S_2 = Hash(0X03 || y_V || Hash(x_V || Z_A || Z_B || x_1 || y_1 || x_2 || y_2))$ 。B 判断  $S_2$  是否与  $S_A$  相等, 如果不相等那么协商失败。如果相等那么协商成功。

此时  $K_A = K_B$ 。

## 3 PGP 加密

### 3.1 原理

第一步、sm2 密钥协商得到对称密钥。

第二步、sm2 加密对称密钥。

第三步、对称加密算法加密明文, 密钥是协商后的对称密钥。

流程图如下。

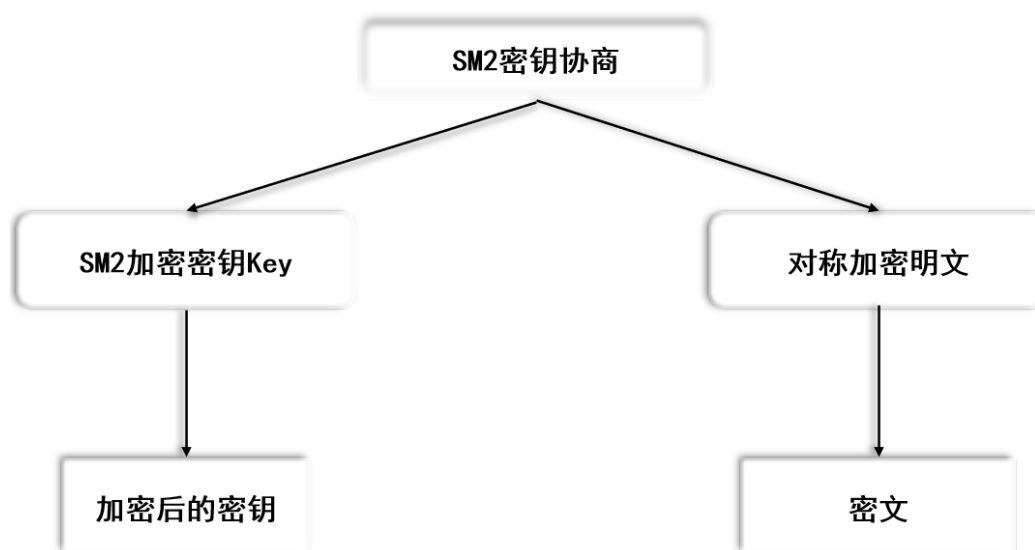


图 2: PGP 加密算法

### 3.2 实现方式

在实现 PGP 加密过程中，首先利用 sm2 实现密钥协商，得到对称密钥 Key。其次，利用 sm2 加密函数对 Key 进行加密，加密公钥是  $P_k$ 。最后，对明文数据使用 AES 对称加密，密钥是 Key。

## 4 PGP 解密

### 4.1 原理

第一步、sm2 解密对称密钥。

第二步、对称加密算法解密密文，密钥是 sm2 解密后的对称密钥。

流程图如下。



图 3: PGP 解密算法

## 4.2 实现方式

在实现 PGP 解密过程中，首先利用私钥  $S_k$ ，使用 sm2 解密算法，得到对称密钥 Key。其次，对密文数据使用 AES 算法，密钥是 Key。最后，得到解密后的数据。

## 5 实验结果

### 5.1 实验结果正确性

SM2 密钥协商正确性验证。

```
d:\vscodefile\pythoncode - V: X + v
A协商的密钥: e8f4fc4578964e9e91799b8269676013
B协商的密钥: e8f4fc4578964e9e91799b8269676013
请按任意键继续. . . |
```

The image shows a terminal window with a dark background. It displays the output of a SM2 key agreement verification process. The first two lines show that both party A and party B have negotiated the same key: 'e8f4fc4578964e9e91799b8269676013'. The third line is a prompt for the user to press any key to continue, followed by a cursor and some dots.

图 4: SM2 密钥协商算法验证

PGP 方案加密函数与解密函数正确性验证。

```

明文是: THANKS
密文是: gAAAAABkxRCyYexZIGT7mPrOWUrFFQPJAmMqQ-JMbPCeubkZ-nXqhalmaY709m0w9TLNjI6I4q3hd5ysU519P942G7L6TZzJQ==
协商的密钥是: 99d54e422a438fadf9d4e14e97ff465b
加密后的密钥是: cff769e9efbf64dc8d793b799a21d6262bcca36cb43da0115cc6b6da50001e940ca7afbf8ae493478933c6be8fe6d1e2f0884b
564640a1e2dfe976f38389e4b 0a5a52005755525053510c510c050356575c540c5d0106545a0f03050154025b 11d2ebe636101776e99f2e706d36c
84c735372f530283b500f6574ead9afa84d
解密后的数据是: THANKS

```

图 5: PGP 加解密算法验证

PGP 加密函数与解密函数时间效率测量。

```

明文是: THANKS
密文是: gAAAAABkxRCyYexZIGT7mPrOWUrFFQPJAmMqQ-JMbPCeubkZ-nXqhalmaY709m0w9TLNjI6I4q3hd5ysU519P942G7L6TZzJQ==
协商的密钥是: 99d54e422a438fadf9d4e14e97ff465b
加密后的密钥是: cff769e9efbf64dc8d793b799a21d6262bcca36cb43da0115cc6b6da50001e940ca7afbf8ae493478933c6be8fe6d1e2f0884b
564640a1e2dfe976f38389e4b 0a5a52005755525053510c510c050356575c540c5d0106545a0f03050154025b 11d2ebe636101776e99f2e706d36c
84c735372f530283b500f6574ead9afa84d
解密后的数据是: THANKS
测量1000次运行时间为: 14.950654029846191
测量1次运行时间为: 0.014950654029846192
请按任意键继续. . .

```

图 6: PGP 加解密算法效率

如上图所示，一次加密与解密耗时为：0.014950654029846192s。

## 6 代码

如下是具体的代码。

### 6.1 sm2 密钥协商

```

1      # 密钥协商
2      def key_change(sa, sb, pa, pb, ida, idb):
3          ida = ida.encode().hex()
4          ENTLa = '{:04X}'.format(len(ida) * 4)
5          za = ENTLa + ida + '{:064X}'.format(a)+'{:064X}'.format(b)+'{:064X}'.format(Gx)+'
{:064X}'.format(Gy)+'{:064X}'.format(pa[0])+'{:064X}'.format(pa[1])
6          za = hex(int(sm3(za),16))[2:]
7          idb = idb.encode().hex()
8          ENTLb = '{:04X}'.format(len(idb) * 4)
9          zb = ENTLb + idb + '{:064X}'.format(a)+'{:064X}'.format(b)+'{:064X}'.format(Gx)+'
{:064X}'.format(Gy)+'{:064X}'.format(pb[0])+'{:064X}'.format(pb[1])
10         zb = hex(int(sm3(zb),16))[2:]
11         w = math.ceil(math.ceil(math.log2(n))/2) - 1
12         while True:
13             ra = random.randint(1, n)

```

```

14         Ra = Mul_Add(Gx, Gy, ra)
15         # 判断Ra是否满足椭圆曲线方程
16         if pow(Ra[1], 2, p) == (pow(Ra[0], 3, p) + a * Ra[0] + b) % p:
17             break
18     x1 = pow(2, w) + (Ra[0] & (pow(2, w) - 1))
19     ta = (sa + x1 * ra) % n
20     while True:
21         rb = random.randint(1, n)
22         Rb = Mul_Add(Gx, Gy, rb)
23         # 判断Ra是否满足椭圆曲线方程
24         if pow(Rb[1], 2, p) == (pow(Rb[0], 3, p) + a * Rb[0] + b) % p:
25             break
26     x2 = pow(2, w) + (Rb[0] & (pow(2, w) - 1))
27     tb = (sb + x2 * rb) % n
28     #密钥长度为256bit
29     klen = 256
30
31     temp1 = Mul_Add(Ra[0], Ra[1], x1)
32     A1 = Add(pa[0], pa[1], temp1[0], temp1[1])
33     V = Mul_Add(A1[0], A1[1], tb)
34     if (len(hex(p)[2:]) * 4 == 192):
35         vx, vy = '{:0192b}'.format(V[0]), '{:0192b}'.format(V[1])
36     else:
37         vx, vy = '{:0256b}'.format(V[0]), '{:0256b}'.format(V[1])
38     kb = KDF(hex(int(vx, 2))[2:] + hex(int(vy, 2))[2:] + za + zb, klen)
39     l = str(0x02)
40     ll = str(0x03)
41     SB = sm3(l + vy + sm3(vx + za + zb + hex(Ra[0])[2:] + hex(Ra[1])[2:] + hex(Rb[0])[2:] +
hex(Rb[1])[2:]))
42
43     temp2 = Mul_Add(Rb[0], Rb[1], x2)
44     B1 = Add(pb[0], pb[1], temp2[0], temp2[1])
45     U = Mul_Add(B1[0], B1[1], ta)
46     if (len(hex(p)[2:]) * 4 == 192):
47         ux, uy = '{:0192b}'.format(U[0]), '{:0192b}'.format(U[1])
48     else:
49         ux, uy = '{:0256b}'.format(U[0]), '{:0256b}'.format(U[1])
50     S1 = sm3(l + uy + sm3(ux + za + zb + hex(Ra[0])[2:] + hex(Ra[1])[2:] + hex(Rb[0])[2:] +
hex(Rb[1])[2:]))
51     if S1 != SB:
52         return False
53     ka = KDF(hex(int(ux, 2))[2:] + hex(int(uy, 2))[2:] + za + zb, klen)
54     SA = sm3(ll + uy + sm3(ux + za + zb + hex(Ra[0])[2:] + hex(Ra[1])[2:] + hex(Rb[0])[2:] +
hex(Rb[1])[2:]))
55     S2 = sm3(ll + vy + sm3(vx + za + zb + hex(Ra[0])[2:] + hex(Ra[1])[2:] + hex(Rb[0])[2:] +

```

```

hex(Rb[1])[2:]))
56         if S2 != SA:
57             return False
58         ka = hex(int(ka, 2)) [2:]
59         ka = binascii.a2b_hex(ka).decode()
60         return ka
61
62

```

## 6.2 PGP 加密

```

1     # 加密
2     def PGP_SM2_ENCRYPT(M, PUBLIC_KEY, KEY):
3         # 加密对称密钥
4         Chip_KEY = encrypt(KEY, PUBLIC_KEY)
5         # 创建加密器
6         KEY = KEY.encode()
7         cipher = Fernet(KEY)
8         # 加密数据
9         encrypted_data = cipher.encrypt(M.encode())
10        return Chip_KEY, encrypted_data.decode()
11

```

## 6.3 PGP 解密

```

1     # 解密
2     def PGP_SM2_DECRYPT(C1, C2, C3, CHIP_TEXT, PRIVATE_KEY):
3         # 解密对称密钥
4         KEY_DECRYPT = decrypt(C1, C2, C3, PRIVATE_KEY)
5         # 创建加密器
6         KEY_DECRYPT = KEY_DECRYPT.encode()
7         cipher = Fernet(KEY_DECRYPT)
8         # 解密数据
9         decrypted_data = cipher.decrypt(CHIP_TEXT.encode())
10        return decrypted_data.decode()
11

```



## 参考文献

- [1] [https://blog.csdn.net/qq\\_30866297/article/details/51194236?ops\\_request\\_misc](https://blog.csdn.net/qq_30866297/article/details/51194236?ops_request_misc)
- [2] <http://c.gb688.cn/bzgk/gb/showGb?type=online&hcno>
- [3] [https://blog.csdn.net/weixin\\_44885334/article/details/121994537?ops\\_request\\_misc](https://blog.csdn.net/weixin_44885334/article/details/121994537?ops_request_misc)