

SM2 2P Dec Report

姜舜天

2023 年 7 月 16 日

1 实现方式

基于 Python ecdsa 库实现的在实际网络中的 SM2 两方加解密算法

总述

SM2 两方加密算法基于 SM2 加密算法，但在于公私钥的生成方式上有所区别

SM2 签名算法中私钥 d 为选取产生，公钥 $P = d \cdot G$

SM2 2P 签名算法中 $d = (d_1 d_2)^{-1} - 1, P = [(d_1 d_2)^{-1} G]$

具体流程如下

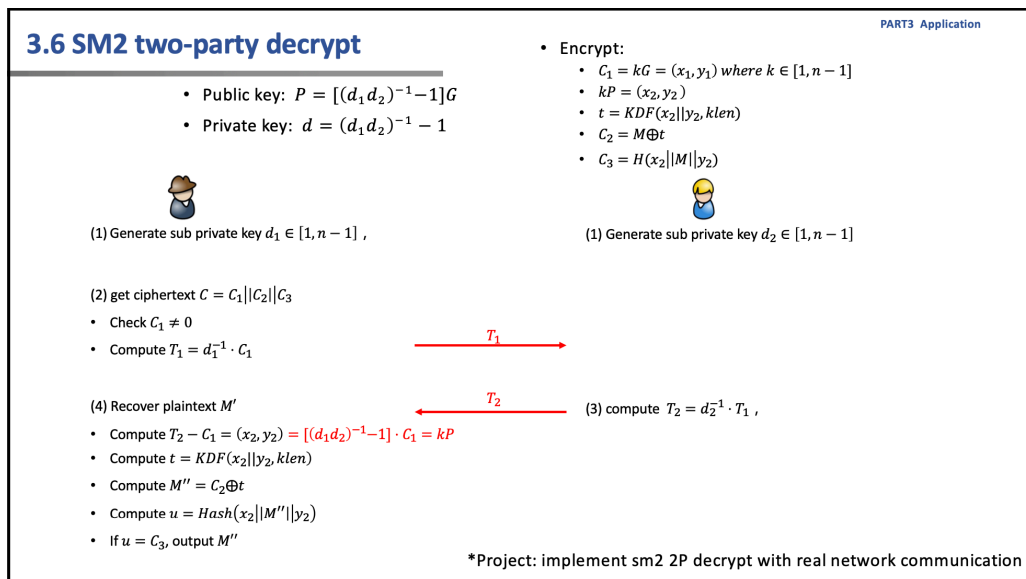


图 1: SM2 加密算法

下面将按照上图一步一步实现 SM2 2P 加密解密算法

1.A (1)&(2)

Alice 随机选取 $d_1 \in [1, n - 1]$, 同时拿到密文 $C = C_1 || C_2 || C_3$, 先检查 $C_1 \neq 0$. 然后计算 $T_1 = d_1^{-1} \cdot G$, 并将 T_1 发送给 Bob

Listing 1: A (1)&(2)

```
1 def s_1_a():
2     d_1 = random.randint(1, SM2_N)
3     P_1 = mod_inverse(d_1, SM2_N) * G
4     return P_1, d_1
5 def s_2_a(C, d_1):
6     if C[0]:
7         T_1 = mod_inverse(d_1, SM2_N) * C[0]
8     return T_1
```

2.B (1)&(3)

Bob 随机选取 $d_2 \in [1, n - 1]$, 计算 $T_2 = d_2^{-1} \cdot T_1$

Listing 2: B (1)&(3)

```
1 def s_1_b(P_1):
2     d_2 = random.randint(1, SM2_N)
3     P = mod_inverse(d_2, SM2_N) * P_1 + (-1) * G
4     return P, d_2
5
6 def s_3_b(T_1, d_2):
7     T_2 = mod_inverse(d_2, SM2_N) * T_1
8     return T_2
```

3.A (4)

进行明文的恢复, 计算 $(x_2, y_2) = T_2 - C_1 = [(d_1 d_2)^{-1} - 1 \cdot C_1] = kP$, 计算 $t = KDF(x_2 || y_2, klen)$ (这里的 klen 采用 256), 计算 $M'' = C_2 \oplus t$, 计算 $u = Hash(x_2 || M'' || y_2)$, 如果 $u = C_3$, 输出 M''

Listing 3: A (3)

```

1 def s_4_a(T_2,C):
2     kdf = T_2 + (-1)*C[0]
3     KDF = str(hex(kdf.x()))[2:] + str(hex(kdf.y()))[2:]
4
5     t = sm3.sm3_kdf(KDF.encode(),256)
6     M_ = C[1]^int(t,16)
7     data = str(hex(M_))[2:].encode()
8     data_1 = binascii.a2b_hex(data)
9
10
11     HASH = str(hex(kdf.x()))[2:] + str(data) + str(hex(kdf.y()))
        [2:]
12     u = int(sm3.sm3_hash(list(HASH.encode())) ,16)
13     print(u)
14
15
16     if(u == C[2]):
17         print( 'yes ' )
18         print( 'datax_□=□' ,data_1)

```

5. 加密算法

随机选取 $k \in [1, n-1]$, 计算 $C_1 = k \cdot G = (x_1, y_1)$, $k \cdot P = (x_2, y_2)$, 计算 $t = KDF(x_2||y_2, klen)$, $C_2 = M \oplus t$, $C_3 = H(x_2||M||y_2)$

Listing 4: SM2 加密算法

```

1 def encrypt(message):
2     k = random.randint(1, SM2_N)
3     C_1 = k * G
4     kP = k * P
5     KDF = str(hex(kP.x()))[2:] + str(hex(kP.y()))[2:]
6

```

```

7     t = sm3.sm3_kdf(KDF.encode(), 256)
8     M = binascii.b2a_hex(message)
9
10
11     HASH = str(hex(kP.x()))[2:] + str(M) + str(hex(kP.y()))[2:]
12
13     M = int(M, 16)
14     C_2 = M ^ int(t, 16)
15     C_3 = int(sm3.sm3_hash(list(HASH.encode())), 16)
16     print(C_3)
17
18     return [C_1, C_2, C_3]

```

2 实现效果

运行环境: MacBook air 2020 16GB Apple M1 Python 3.9

测试代码:

Listing 5: test bench

```

1 P_1, d_1 = s_1_a()
2 P, d_2 = s_1_b(P_1)
3 C = encrypt(b'Sunshine_Rainbow_Pony')
4 print('密文为:\n', C)
5 T_1 = s_2_a(C, d_1)
6 T_2 = s_3_b(T_1, d_2)
7 s_4_a(T_2, C)

```

```
main
/Users/mac/Documents/Py Programs/sm2p22/venv/bin/python" /Users/mac/Documents/Py Programs/sm2p22/main.py
41087648408135664975706001741971743525386525839988332822525707866359468756400
秘文为:
[<ecdsa.ellipticcurve.Point object at 0x1003e13a0>, 1850455748954004187420947100327477972375196857452920463918352744863978812485288823847012011103374828258651'
41087648408135664975706001741971743525386525839988332822525707866359468756400
yes
datax = b'Sunshine Rainbow Pony'

进程已结束,退出代码0
```

图 2: 执行结果

参考文献

- [1] <https://blog.csdn.net/Digquant/article/details/124429472>

A 附录标题

Listing 6: SM2 参数设置

```
1 p= FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF
   FFFFFFFF
2
3 a= FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF
   FFFFFFFC
4
5 b= 28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41
   4D940E93
6
7 n= FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409
   39D54123
8
9 Gx= 32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1715A4589
   334C74C7
10
11 Gy= BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02
   DF32E5 2139F0A0
12 # 定义椭圆曲线参数
13 p = SM2_P
14 a = SM2_A
15 b = SM2_B
16 n = SM2_N
17 Gx = SM2_Gx
18 Gy = SM2_Gy
19 curve_sm2 = ellipticcurve.CurveFp(p, a, b)
20
21 # 定义生成器点G
22 G = ellipticcurve.Point(curve_sm2, Gx, Gy, n)
```