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

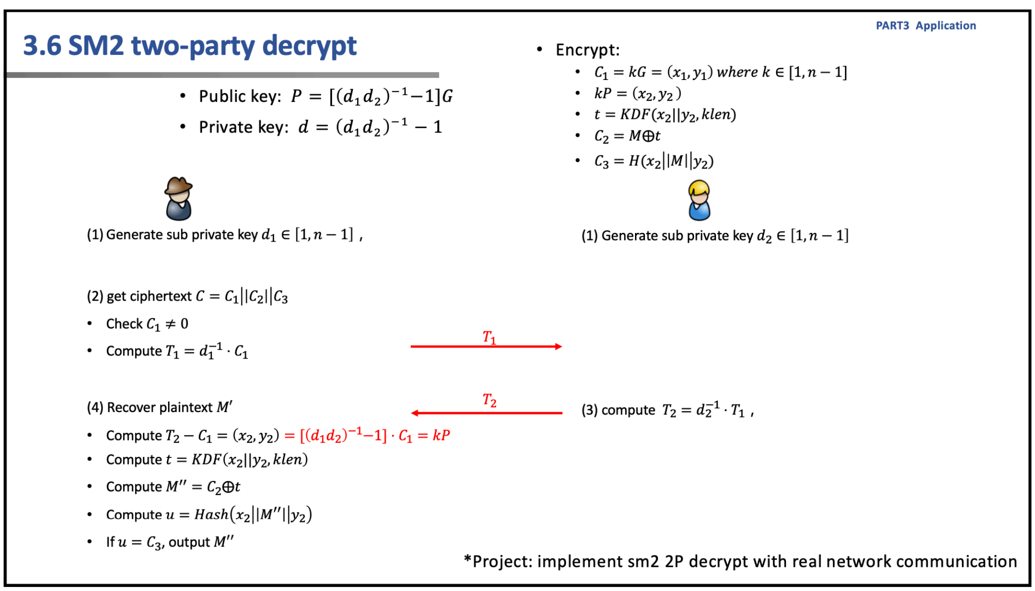

图 1: SM2 加密算法

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)

```
def s_1_a():
    d_1 = random.randint(1, SM2_N)
    P_1 = mod_inverse(d_1, SM2_N) * G
    return P_1, d_1
def s_2_a(C,d_1):
     if C[0]:
        T_1 = mod_inverse(d_1,SM2_N)*C[0]
        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)

```
def s_1_b(P_1):
    d_2 = random.randint(1, SM2_N)
    P = mod_inverse(d_2, SM2_N) * P_1 + (-1) * G
    return P, d_2

def s_3_b(T_1,d_2):
    T_2 = mod_inverse(d_2,SM2_N)*T_1
    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)

```python
def s_4_a(T_2,C):
    kdf = T_2 + (-1)*C[0]
    KDF = str(hex(kdf.x()))[2:]+str(hex(kdf.y()))[2:]

    t = sm3.sm3_kdf(KDF.encode(),256)
    M_ = C[1]^int(t,16)
    data = str(hex(M_))[2:].encode()
    data_1 = binascii.a2b_hex(data)


    HASH = str(hex(kdf.x()))[2:] + str(data) + str(hex(kdf.y()))
        [2:]
    u = int(sm3.sm3_hash(list(HASH.encode())),16)
    print(u)


    if(u == C[2]):
        print('yes')
        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

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

```

```
    t = sm3.sm3_kdf(KDF.encode(), 256)
    M = binascii.b2a_hex(message)


    HASH = str(hex(kP.x()))[2:] + str(M) + str(hex(kP.y()))[2:]

    M = int(M, 16)
    C_2 = M ^ int(t, 16)
    C_3 = int(sm3.sm3_hash(list(HASH.encode())),16)
    print(C_3)

    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

```
P_1, d_1 = s_1_a()
P, d_2 = s_1_b(P_1)
C = encrypt(b'Sunshine Rainbow Pony')
print('密文为:\n',C)
T_1 = s_2_a(C,d_1)
T_2 = s_3_b(T_1,d_2)
s_4_a(T_2,C)
```

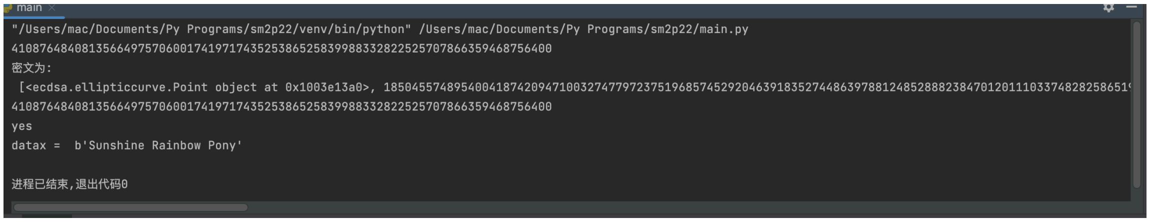

图 2: 执行结果

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

```
p= FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF

a= FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC

b= 28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93

n= FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123

Gx= 32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1715A4589 334C74C7

Gy= BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02 DF32E5 2139F0A0
# 定义椭圆曲线参数
p = SM2_P
a = SM2_A
b = SM2_B
n = SM2_N
Gx = SM2_Gx
Gy = SM2_Gy
curve_sm2 = ellipticcurve.CurveFp(p, a, b)

# 定义生成器点G
G = ellipticcurve.Point(curve_sm2, Gx, Gy, n)
```