

SM2 Implement Report

姜舜天

2023 年 7 月 14 日

1 实现方式

基于 Python ecdsa 库实现的 SM2 签名算法

1.SM2 参数设置:

sm2 使用素数域 256 位椭圆曲线 $y^2 = x^3 + ax + b$, 参数如下

Listing 1: 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 59BDC EE3 6B692153 D0A9877C C62A4740 02
    DF32E5 2139F0A0
12
13 # 定义椭圆曲线参数
14 p = SM2_P
15 a = SM2_A
16 b = SM2_B
17 n = SM2_N
18 Gx = SM2_Gx
19 Gy = SM2_Gy
20 curve_sm2 = ellipticcurve.CurveFp(p, a, b)
21
22 # 定义生成器点G
23 G = ellipticcurve.Point(curve_sm2, Gx, Gy, n)

```

2.SM2 密钥生成算法:

Alice 随机选取 d 作为私钥计算公钥 $P = d \cdot G$, 其中 $G = (x, y)$ 是 SM2 使用的椭圆曲线上的生成元, 输出密钥对 (d, P)

Listing 2: SM2 KeyGen

```

1 def SM2_Key_Generate():
2     d = random.randint(2^160, SM2_N - 1)
3     public_key = d * G
4     secret_key = d
5     pair = [secret_key, public_key]
6     return pair

```

3.SM2 签名算法:

假定消息为 M , 使用 SM3 计算 $e = H(M)$, 生成随机数 k , 计算 $c = k \cdot G$, 计算 $r = (e + x) \bmod n$, 其中 n 为 G 的阶数, x 为 G 的第一个坐标, 如果 $r = 0$ 或 $r + k = n$ 则重新生成 k , 计算 $s = (1 + d)^{-1} \cdot (k - r \cdot d) \bmod n$, 若 $s = 0$, 则重新生成 k , 输出签名 (r, s)

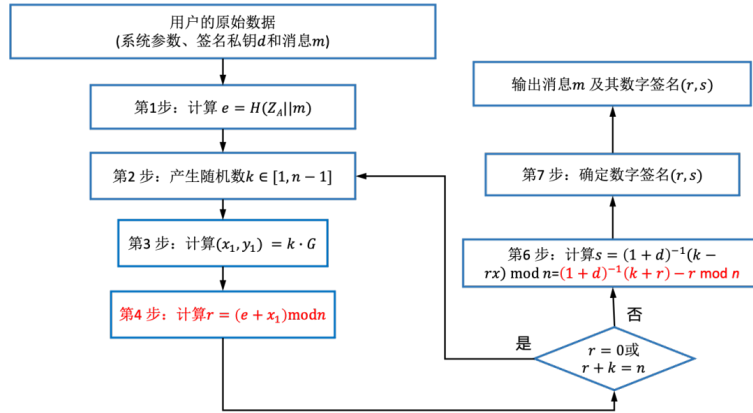


图 1: SM2 签名算法

Listing 3: SM2 Sign

```

1 def sm2_sig(message, sk):
2     message = message.encode()
3     message = list(message)
4     print(message)
5     e = sm3.sm3_hash(message) # 获取消息 hash 值
6     e = int(e, 16)
7     print('消息散列值为', e)
8     k = random.randint(1, SM2_N - 1)
9     c = k * G
10    r = pow(e + c.x(), 1, SM2_N)
11    mid = mod_inverse((1 + sk), SM2_N)
12    s = pow(mid * (k - r * sk), 1, SM2_N)
13
14    while r == 0 or (r + k) == SM2_Nors == 0:
15        k = random.randint(1, SM2_N - 1)
16        c = k * G
17        r = pow(e + c.x(), 1, SM2_N)
18        mid = mod_inverse((1 + sk), SM2_N)
19        s = pow(mid * (k - r * sk), 1, SM2_N)
  
```

20
21
22

```
sig = [r, s]
return sig
```

4.SM2 签名验证算法：假定消息为 M ，收到的签名为 (r', s') ，使用 SM3 计算 $e' =$

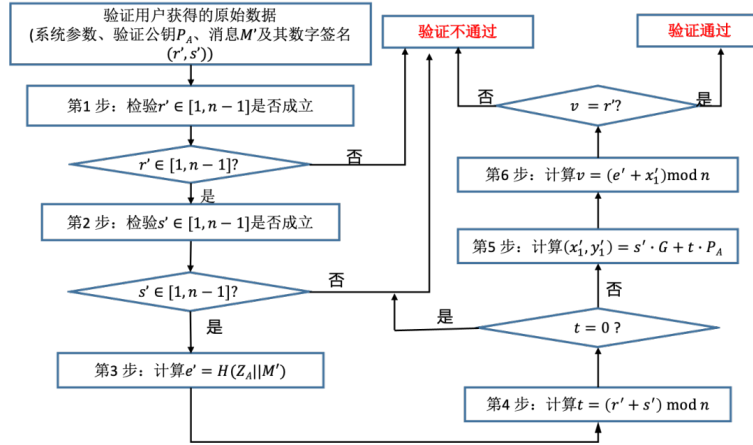


图 2: SM2 签名验证算法

$H(M)$ ，计算 $t = (r' + s') \bmod n$ ，计算 $(x', y') = s' \cdot G + t' \cdot P$ ，计算 $R = (e + x') \bmod n$ ，检验 $R == r$ ，若为 1 则验证通过，否则不通过

Listing 4: SM2 Verify

1
2
3
4
5
6
7
8
9
10
11

```
def sm2_ver(message, r, s, pk):
    message = message.encode()
    message = list(message)
    e = sm3.sm3_hash(message) # 获取消息 hash 值

    e = int(e, 16)
    t = pow((r + s), 1, SM2_N)
    mid = s * G + t * pk
    R = pow(e + mid.x(), 1, SM2_N)
    if R == r:
        return 1
```

```
12     else:
13         return 0
```

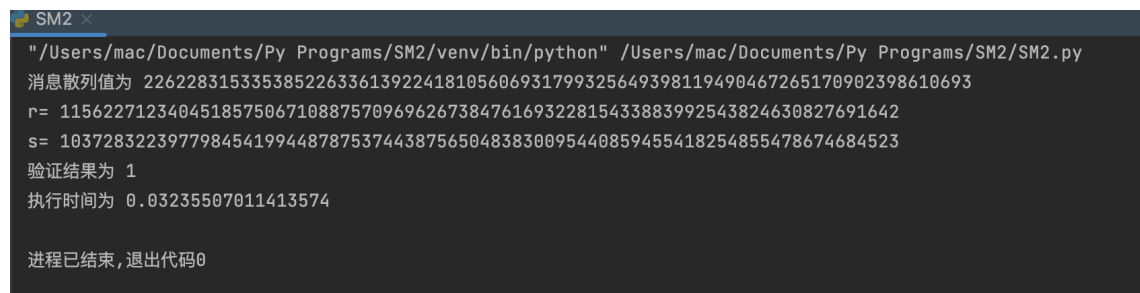
2 实现效果

运行环境: Apple M1 16GB Python 3.9

测试代码:

Listing 5: test bench

```
1 message = 'sm2'
2 t0 = time.time()
3 sk, pk = SM2_Key_Generate()
4 sig = sm2_sig(message, sk)
5 out = sm2_ver(message, sig[0], sig[1], pk)
6 t1 = time.time()
7 print('r=', sig[0])
8 #print('r=', str(hex(sig[0])))
9 print('s=', sig[1])
10 print('验证结果为', out)
11 print('执行时间为', t1-t0)
```



```
SM2
"/Users/mac/Documents/Py Programs/SM2/venv/bin/python" /Users/mac/Documents/Py Programs/SM2/SM2.py
消息散列值为 22622831533538522633613922418105606931799325649398119490467265170902398610693
r= 115622712340451857506710887570969626738476169322815433883992543824630827691642
s= 103728322397798454199448787537443875650483830095440859455418254855478674684523
验证结果为 1
执行时间为 0.03235507011413574

进程已结束,退出代码0
```

图 3: 执行结果

参考文献

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