

SM2 2P Sign 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 两方签名算法

总述

SM2 两方签名算法基于 SM2 签名算法，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 $d = (d_1 d_2)^{-1} - 1, P = [(d_1 d_2)^{-1} G]$

验证算法与 SM2 算法中一致，具体流程如下

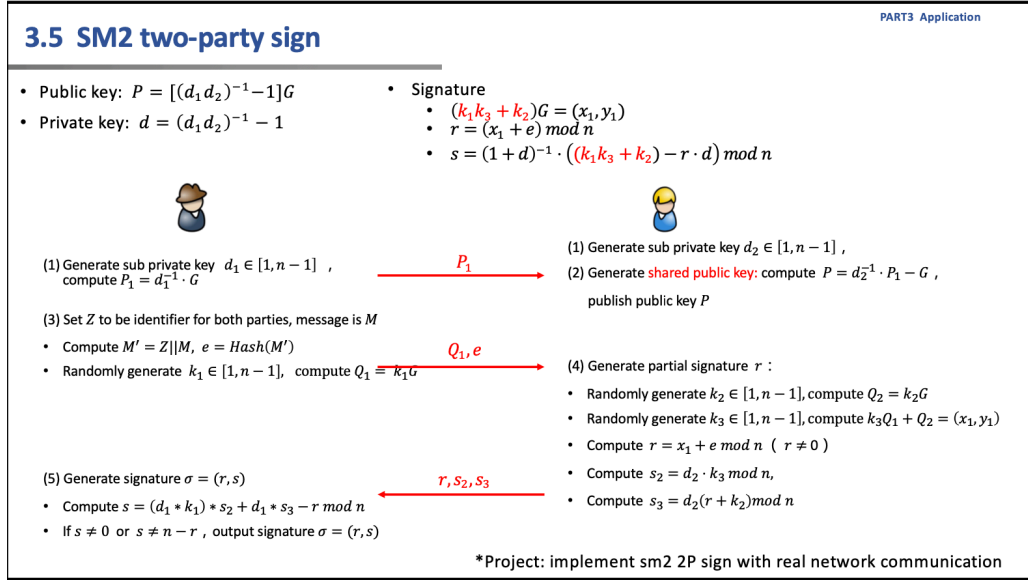


图 1: SM2 签名算法

下面将按照上图一步一步实现 SM2 2P 签名算法

1.A (1)

Alice 随机选取 $d_1 \in [1, n - 1]$, 计算 $P_1 = d_1^{-1} \cdot G$, 并将 P_1 发送给 Bob

Listing 1: A (1)

```
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
```

2.B (1)&(2)

Bob 随机选取 $d_2 \in [1, n - 1]$, 生成公钥 $P = [(d_1 d_2)^{-1} G]$

Listing 2: B (1)&(2)

```
1 def s_12_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
```

3.A (3)

设置 Z 作为两方的一个验证标识符, 假定消息是 M (在这里设置为‘identifier-’), 计算 $M' = Z || M, e = Hash(M')$, 随机选取 $k_1 \in [1, n - 1]$, 计算 $Q_1 = k_1 \cdot G$, 将 Q_1, e 发送给 Bob

Listing 3: A (3)

```
1 def s_2_a(M):
2     Z = 'identifier-'
3     M_ = Z + M
4
5     e = sm3.sm3_hash(list(M_.encode()))
6     print(e)
7
8     e = int(e, 16) # 获取消息 hash 值
9     k_1 = random.randint(1, SM2_N)
```

```

10     Q_1 = k_1 * G
11     return e, Q_1, k_1

```

4.B (4)

随机生成 $k_2 \in [1, n-1], k_3 \in [1, n-1]$, 计算 $Q_2 = k_2 \cdot G, k_3 \cdot Q_1 + Q_2 = (x_1, y_1)$, 接着计算 $r = x_1 + e \bmod n, s_2 = d_2 \cdot k_3 \bmod n, s_3 = d_2(r + k_2) \bmod n$, 将 r, s_2, s_3 发送给 Alice

Listing 4: B (4)

```

1  def s_4_b(e, Q_1, d_2):
2      k_2 = random.randint(1, SM2_N)
3      Q_2 = k_2 * G
4
5      k_3 = random.randint(1, SM2_N)
6      Q_3 = k_3 * Q_1 + Q_2
7
8      r = pow(Q_3.x() + e, 1, SM2_N)
9      s_2 = pow(d_2 * k_3, 1, SM2_N)
10     s_3 = pow(d_2 * (r + k_2), 1, SM2_N)
11
12     return r, s_2, s_3

```

5.A (5)

计算签名 $\sigma = (r, s)$, 其中 $s = (d_1 * k_1) \cdot s_2 + d_1 * s_3 - r \bmod n$, 如果 $s \neq 0$ or $s \neq n - r$, 输出 σ

Listing 5: A (5)

```

1  def s_5_a(r, s_2, s_3, d_1, k_1):
2      s = pow((d_1 * k_1) * s_2 + d_1 * s_3 - r, 1, SM2_N)
3      if s == 0 or s == SM2_N - r:
4          print("Fail to generate")
5      else:
6          return [r, s]

```

2 实现效果

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

测试代码:

Listing 6: test bench of Alice

```
1 s = socket.socket()
2 host = socket.gethostname()
3 port = 9999
4 s.bind((host, port))
5
6 s.listen(5)
7 p1, d1 = s_1_a()
8
9 c, addr = s.accept()
10 print('连接地址为: ', addr)
11 c.send(b'Here_is_pk')
12
13 serialized_point = pickle.dumps(p1)
14 c.send((serialized_point))
15 M = 'Sunshine_Rainbow_Pony_SM2'
16
17 e, Q1, k1 = s_2_a(M)
18
19 serialized_Q1 = pickle.dumps(Q1)
20 c.send(serialized_Q1)
21 print(len(serialized_Q1))
22 print('Here_is_Q1:\n', Q1)
23
24 send_e = str(e)
25 c.send(send_e.encode())
26 print('Here_is_e:\n', e)
```

```

27 print(len(send_e.encode()))
28
29 seq = c.recv(256).decode()
30 len = []
31 len.append(int(seq[0:2]))
32 len.append(int(seq[2:4]))
33 len.append(int(seq[4:6]))
34 r = int(seq[6:6+len[0]])
35 s2 = int(seq[6+len[0]:6+len[0]+len[1]])
36 s3 = int(seq[6+len[0]+len[1]:6+len[0]+len[1]+len[2]])
37
38 print( 'Here_is_r:\n', r)
39 print( 'Here_is_s2:\n', s2)
40 print( 'Here_is_s3:\n', s3)
41 print( 'Now_generate_s')
42
43 s = s_5_a(r, s2, s3, d1, k1)
44 print( 'Here_is_s:\n',s)
45
46 c.close()

```

Listing 7: test bench of Bob

```

1 s = socket.socket()
2 host = socket.gethostname()
3 port = 9999
4 s.connect((host, port))
5 print(s.recv(1024))
6 p1 = s.recv(1024)
7 p1 = pickle.loads(p1)
8
9 print( 'P1_is:', p1)

```

```

10 print( 'Now_generate_P' )
11 P, d2 = s_12_b(p1)
12
13 Q1 = s.recv(366)
14 Q1 = pickle.loads(Q1)
15 print( 'Q1_is_:\n', Q1)
16
17 e = s.recv(100)
18 print(type(e))
19
20 print( 'e_is_:\n', int(e.decode()))
21
22 e = int(e)
23
24 print( 'Now_generate_r' )
25 r, s2, s3 = s_4_b(e, Q1, d2)
26 print( 'Here_is_r:\n', r)
27 print( 'Here_is_s2:\n', s2)
28 print( 'Here_is_s3:\n', s3)
29 print(len(str(r)),len(str(s2)),len(str(s3)))
30
31
32 seq = str(len(str(r)))+str(len(str(s2)))+str(len(str(s3)))+str(r)
    +str(s2)+str(s3)
33
34
35 s.send(seq.encode())
36
37
38 s.close()

```

```
"/Users/mac/Documents/Py Programs/SM2p21/venv/bin/python" /Users/mac/Documents/Py Programs/SM2p21/Alice_server.py
连接地址为: ('127.0.0.1', 61866)
fdb0ea3ee624a0526fb438c783d340903ea46558f489b65984fdc59b24f53717
360
Here is Q1:
(11881470326528953675762943380203039082576640104594876182219716675656900934204, 29546435891548170476353391381451635379)
Here is e:
114747732479366128508056041324618706088689723351884763141087830683808945223447
78
Here is r:
77002581107133083554925261018650264177476596156726314163356792552678029695642
Here is s2:
106126600663964412926197582071935034054317619204621509444652478468103917881031
Here is s3:
19447093159985734539958058659719268435211672951084515847511033801915453304565
Now generate s
Here is s:
[77002581107133083554925261018650264177476596156726314163356792552678029695642, 4153709122617280617191131796025417208]

进程已结束, 退出代码0
```

图 2: A 执行结果

```
"/Users/mac/Documents/Py Programs/SM2p21/venv/bin/python" /Users/mac/Documents/Py Programs/SM2p21/Bob_client.py
b'Here is pk'
P1 is : (82316826642832023265433775662189996489475361146908904114575145248424835001577, 8787996350618547252540697769516)
Now generate P
Q1 is :
(11881470326528953675762943380203039082576640104594876182219716675656900934204, 29546435891548170476353391381451635379)
<class 'bytes'>
e is :
114747732479366128508056041324618706088689723351884763141087830683808945223447
Now generate r
Here is r:
77002581107133083554925261018650264177476596156726314163356792552678029695642
Here is s2:
106126600663964412926197582071935034054317619204621509444652478468103917881031
Here is s3:
19447093159985734539958058659719268435211672951084515847511033801915453304565
77 78 77

进程已结束, 退出代码0
```

图 3: B 执行结果

Listing 8: testbench of sig verification

```

1 P_1 , d_1 = s_1_a()
2
3 P, d_2 = s_12_b(P_1)
4
5 test = pow((mod_inverse(d_1*d_2,SM2_N)-1),1,SM2_N)*G
6
7 print(P)
8 print(test)
9
10 e, Q_1, k_1 = s_2_a('sm22p')
11
12 r, s_2, s_3 = s_4_b(e,Q_1,d_2)
13
14 sig = s_5_a(r,s_2,s_3,d_1,k_1)
15
16 print('签名结果: \n',sig)
17
18 out = sm2_ver('sm22p',sig[0],sig[1],P)
19
20 print('验证结果: \n',out)

```

```

"/Users/mac/Documents/Py Programs/SM2p21/venv/bin/python" /Users/mac/Documents/Py Programs/SM2p21/SM2PSign.py
(25276493491907868469751306382293671116153378888490618864424130267019523916776,507417409667592401907548361742311171752
(25276493491907868469751306382293671116153378888490618864424130267019523916776,507417409667592401907548361742311171752
aaa49dfccd2e340ddde90e9da148b92cd9a8e8ea389f57a778d05453b813f2b8
签名结果:
[88626980221309019278975076582672944383632967580222909945006144599973509505373, 5741236537684928828700900132434585874
aaa49dfccd2e340ddde90e9da148b92cd9a8e8ea389f57a778d05453b813f2b8
88626980221309019278975076582672944383632967580222909945006144599973509505373
88626980221309019278975076582672944383632967580222909945006144599973509505373
验证结果:
1
进程已结束,退出代码0

```

图 4: 签名验证测试结果

参考文献

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

A 附录标题

Listing 9: SM2 参数设置

```
1 p= FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF
   FFFFFFFF
2
3 a= FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF
   FFFFFFFC
4
5 b= 28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41
   4D940E93
6
7 n= FFFFFFFE 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)
```