

Implement SM2 2P sign with real network communication

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 实验原理

流程图如下所示。

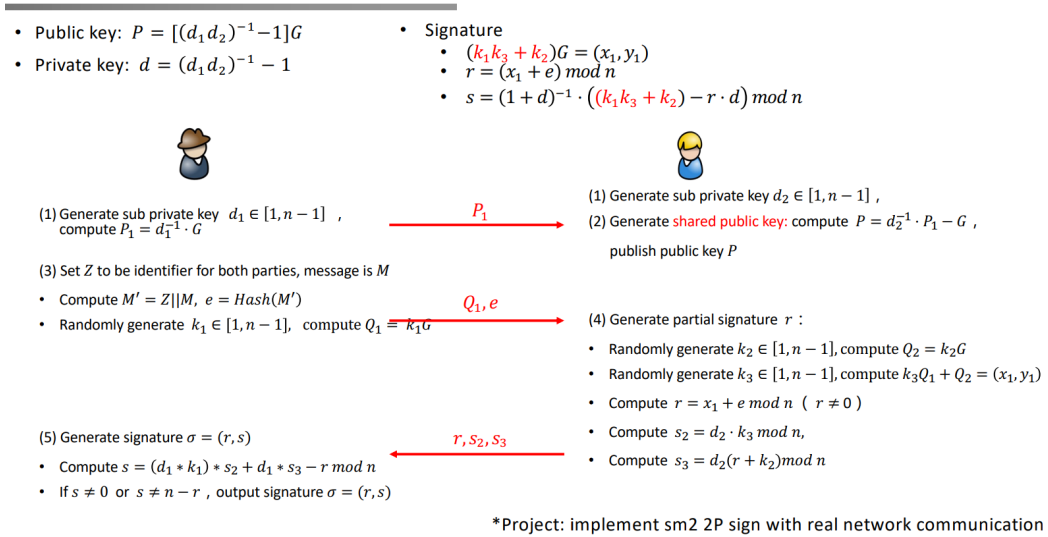


图 1: SM2_2P_SIGN 流程图

3 实现方法

实现签名算法，使用 python 的 TCP 通信。通信双方为 A 和 B。其中，A 向 B 发送 P_1 , Q_1 和 e ，B 向 A 发送 r, s_2, s_3 。

在代码 SM2_2P_SIGN_A.py 中，实现 A 的功能。第一步，A 随机生成 d_1 ，然后计算 P_1 ，将 P_1 发给 B。第二步，A 在收到 B 的确认后计算 Q_1 和 e ，并将其发给 B。第三步，A 在收到 B 发送的 r, s_2, s_3 后依次回复确认，计算出 s 。

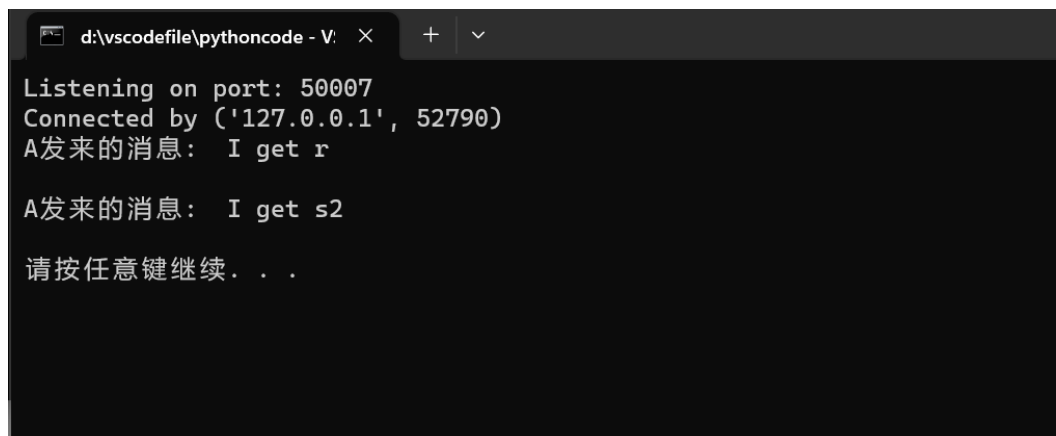
在代码 SM2_2P_SIGN_B.py 中，实现 B 的功能。第一步，B 随机生成 d_2 。第二步，收到 A 发送的 P_1 后回复确认，然后计算 P 。第三步，收到 A 发送的 Q_1 和 e ，依次回复确认，计算出 r, s_2, s_3 ，并将其发送给 A。

注意，由于通信质量较差，在最初的实现中，数据是连续发送的，如 A 将 Q_1 和 e 直接发给 B。但是这样容易出现数据丢失的问题，或者数据合并到一起发送，容易出错。所以改为了数据发送一个，对方收到后就发送一个回复，以确保数据传输准确。但是这样降低了效率。

4 实验结果

实验结果如下。

B 的结果展示。



```
d:\vscodefile\pythoncode - V  ×  +  ▾  
Listening on port: 50007  
Connected by ('127.0.0.1', 52790)  
A发来的消息: I get r  
  
A发来的消息: I get s2  
  
请按任意键继续...
```

图 2: B 通信实验结果

A 的结果展示。

```
d:\vscodefile\pythoncode - V: X + v
B发来的消息: I have get P1[0]
B发来的消息: I have get P1[1]
B发来的消息: I have get e
B发来的消息: I have get Q1[0]
B发来的消息: I have get Q1[1]
消息: I am Homelander
r: 3b381e78491f9e50cf85fb4478b4eae85d024b699d886f4c23af7a117024d0f7
S: 82ec693b867cc8a521c9883510ed91eb32e0451835424aa5d7f3877e7d39d505
耗时: 0.00594329833984375
请按任意键继续. . .
```

图 3: A 通信实验结果

如上图所示，一次签名耗时为：0.00594329833984375s。

5 代码

如下是核心的代码。

5.1 SM2_2P_SIGN_B

```
1 import gmpy2
2 import random
3 import socket
4 from threading import Thread
5
6 HOST = ''
7 PORT = 50007
8
9 def SIGN_2P_B(conn, addr):
10     while True:
11         # 接收p1
12         p1 = conn.recv(1024).decode()
13         conn.sendall("I_have_get_P1[0]".encode())
14         p2 = conn.recv(1024).decode()
15         conn.sendall("I_have_get_P1[1]".encode())
16         p1 = int(p1)
17         p2 = int(p2)
18         P1 = (p1, p2)
```

```

19
20     # Second Step
21     d2 = random.randint(1, n-1)
22     Pub = Mul__Add(P1[0], P1[1], gmpy2.invert(d2, n))
23     P = Add(Pub[0], Pub[1], Gx, p - Gy)
24
25     # 接收e和Q1
26     e = conn.recv(1024).decode()
27     conn.sendall("I_have_get_e".encode())
28     q1 = conn.recv(1024).decode()
29     conn.sendall("I_have_get_Q1[0]".encode())
30     q2 = conn.recv(1024).decode()
31     conn.sendall("I_have_get_Q1[1]".encode())
32
33     q1 = int(q1)
34     q2 = int(q2)
35     Q1 = (q1, q2)
36
37     # Forth Step
38     k2 = random.randint(1, n - 1)
39     Q2 = Mul__Add(Gx, Gy, k2)
40     k3 = random.randint(1, n - 1)
41     Q3_TEMP = Mul__Add(Q1[0], Q1[1], k3)
42     Q3 = Add(Q3_TEMP[0], Q3_TEMP[1], Q2[0], Q2[1])
43     r = (Q3[0] + int(e, 16)) % n
44     s2 = (d2 * k3) % n
45     s3 = (d2 * (r + k2)) % n
46
47     # 给A发送r, s2, s3
48     conn.sendall(str(r).encode())
49     data_1 = conn.recv(1024).decode()
50     print("A发来的消息:", data_1, '\n')
51     conn.sendall(str(s2).encode())
52     data_2 = conn.recv(1024).decode()
53     print("A发来的消息:", data_2, '\n')
54     conn.sendall(str(s3).encode())
55
56     break
57
58 conn.close()
59
60 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
61 sock.bind((HOST, PORT))
62 sock.listen(10)
63 print('Listening on port:', PORT)

```

```

64     conn, addr = sock.accept()
65     print('Connected by', addr)
66     mthread = Thread(target = SIGN_2P_B, args = (conn, addr))
67     mthread.start()
68     sock.close()
69

```

5.2 SM2_2P_SIGN_A

```

1     # 签名算法
2     def SIGN_2P_A(Message):
3         # First Step
4         d1 = random.randint(1, n - 1)
5         p1 = Mul_Add(Gx, Gy, gmpy2.invert(d1, n))
6
7         # 发送P1
8         SOCKET.sendall(str(p1[0]).encode())
9         data1 = SOCKET.recv(1024).decode()
10        print("B发来的消息:", data1, '\n')
11        SOCKET.sendall(str(p1[1]).encode())
12        data2 = SOCKET.recv(1024).decode()
13        print("B发来的消息:", data2, '\n')
14
15        # Third Step
16        Z = "I can do whatever I want to do"
17        M = Z + Message
18        e = sm3(M)
19        k1 = random.randint(1, n - 1)
20        Q1 = Mul_Add(Gx, Gy, k1)
21
22        # 将e和Q1发给B
23        SOCKET.sendall(e.encode())
24        data3 = SOCKET.recv(1024).decode()
25        print("B发来的消息:", data3, '\n')
26        SOCKET.sendall(str(Q1[0]).encode())
27        data4 = SOCKET.recv(1024).decode()
28        print("B发来的消息:", data4, '\n')
29        SOCKET.sendall(str(Q1[1]).encode())
30        data5 = SOCKET.recv(1024).decode()
31        print("B发来的消息:", data5, '\n')
32
33        # 接收B发来的r, s2, s3
34        r = SOCKET.recv(1024).decode()
35        SOCKET.sendall("I get r".encode())

```

```

36         s2 = SOCKET.recv(1024).decode()
37         SOCKET.sendall("I_get_s2_".encode())
38         s3 = SOCKET.recv(1024).decode()
39
40         r = int(r, 16)
41         s2 = int(s2, 16)
42         s3 = int(s3, 16)
43
44         # Fifth Step
45         s = ((d1 * k1) * s2 + d1 * s3 - r) % n
46         if s == 0 and s == n - r:
47             print("\n签名失败!\n")
48             return False
49         SOCKET.close()
50         return hex(r) [2:], hex(s) [2:]
51
52     # 示例
53     MESSAGE = "I_am_Homeland"
54     time_start = time.time()
55     SIGNATURE_R, SIGNATURE_S = SIGN_2P_A(MESSAGE)
56     time_end = time.time()
57     print("消息: ", MESSAGE, "\n")
58     print("r:", SIGNATURE_R, "\n")
59     print("S:", SIGNATURE_S, "\n")
60     print("耗时:", time_end - time_start, '\n')
61

```