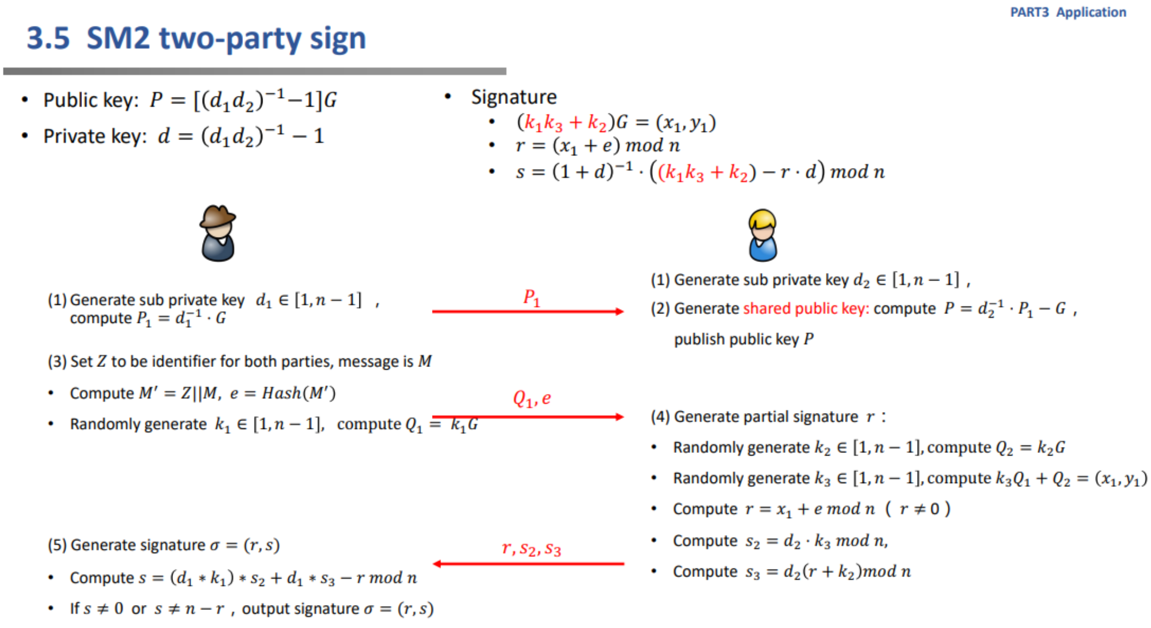
Project: implement sm2 2P sign with real network communication

按照图示实现了两方签名 未给出验证算法

代码直接运行即可



PART3 Application

3.5 SM2 two-party sign

- Public key: $P = [(d_1 d_2)^{-1} - 1]G$
- Private key: $d = (d_1 d_2)^{-1} - 1$

- Signature
  - $(k_1 k_3 + k_2)G = (x_1, y_1)$
  - $r = (x_1 + e)\ mod\ n$
  - $s = (1 + d)^{-1} \cdot ((k_1 k_3 + k_2) - r \cdot d)\ mod\ n$

(1) Generate sub private key $d_1 \in [1, n-1]$, compute $P_1 = d_1^{-1} \cdot G$

$\xrightarrow{P_1}$

(1) Generate sub private key $d_2 \in [1, n-1]$,

(2) Generate shared public key: compute $P = d_2^{-1} \cdot P_1 - G$, publish public key P

(3) Set Z to be identifier for both parties, message is M

- Compute $M' = Z||M,\ e = Hash(M')$
- Randomly generate $k_1 \in [1, n-1]$, compute $Q_1 = k_1 G$

$\xrightarrow{Q_1, e}$

(4) Generate partial signature r :

- Randomly generate $k_2 \in [1, n-1]$, compute $Q_2 = k_2 G$
- Randomly generate $k_3 \in [1, n-1]$, compute $k_3 Q_1 + Q_2 = (x_1, y_1)$
- Compute $r = x_1 + e\ mod\ n$ ($r \neq 0$)
- Compute $s_2 = d_2 \cdot k_3\ mod\ n,$
- Compute $s_3 = d_2(r + k_2) mod\ n$

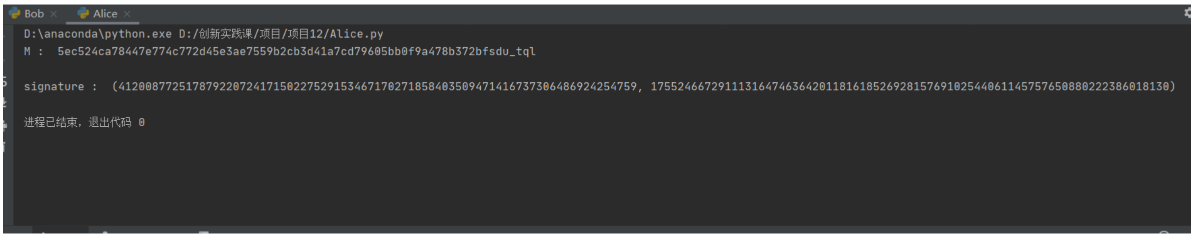
$\xleftarrow{r, s_2, s_3}$

(5) Generate signature $\sigma = (r, s)$

- Compute $s = (d_1 * k_1) * s_2 + d_1 * s_3 - r\ mod\ n$
- If $s \neq 0$ or $s \neq n - r$, output signature $\sigma = (r, s)$

结果如下:

Alice:

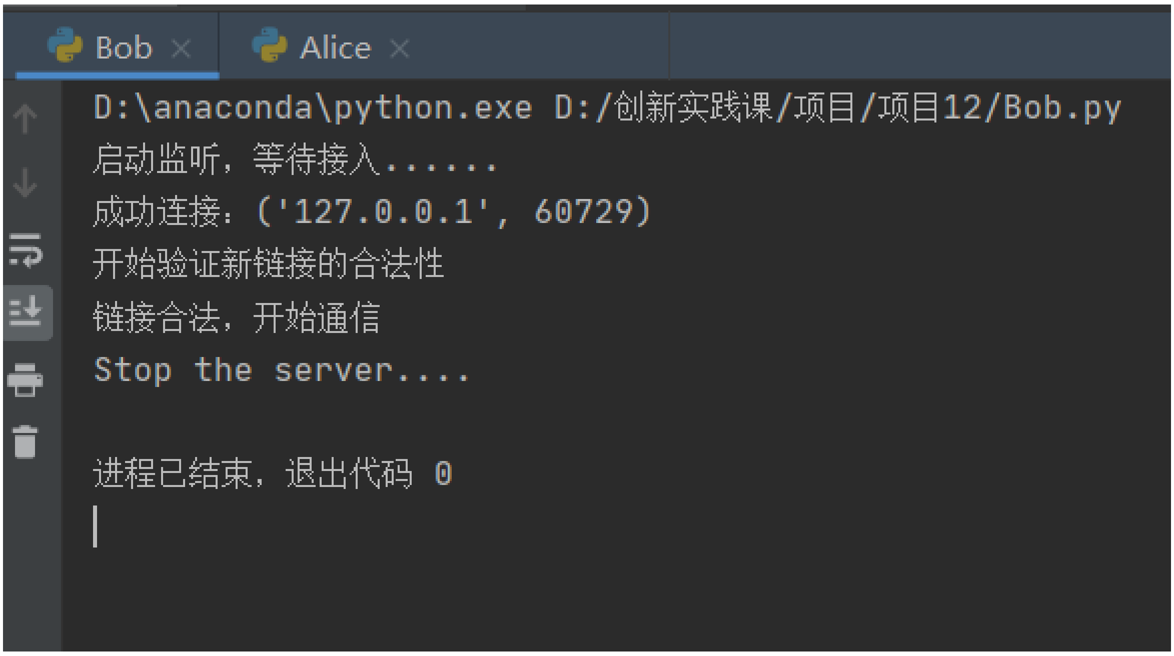


```
D:\anaconda\python.exe D:/创新实践课/项目/项目12/Alice.py
M :  5ec524ca78447e774c772d45e3ae7559b2cb3d41a7cd79605bb0f9a478b372bfsdu_tql

signature :  (41200877251787922072417150227529153467170271858403509471416737306486924254759, 17552466729111316474636420118161852692815769102544061145757650880222386018130)

进程已结束，退出代码 0
```

Bob:



```
D:\anaconda\python.exe D:/创新实践课/项目/项目12/Bob.py
启动监听，等待接入......
成功连接：('127.0.0.1', 60729)
开始验证新链接的合法性
链接合法，开始通信
Stop the server....

进程已结束，退出代码 0
```