## Project-19， forge a signature to pretend that you are Satoshi

### 1，实验内容

通过中本聪的公钥，在没有密钥情况下伪造中本聪签名并通过验证

首先可以查到中本聪的公钥和签名：

```python

P\_Satoshi=(26877259512020005462763638353364532382639391845761963173968516804546337027093,48566944205781153898153509065115980357578581414964392335433501542694784316391)

sig\_Satoshi=(41159732757593917641705955129814776632782548295209210156195240041086117167123, 57859546964026281203981084782644312411948733933855404654835874846733002636486)

```

伪造签名的原理如下：

只需要中本聪的公钥便可伪造其签名（在验签过程中不重新计算e，而是直接使用给出的e进行验证）

验签过程如下：

from base import \*

import random

if \_\_name\_\_ == "\_\_main\_\_":

prikey, pubkey = Key\_Generation()

print("生成的 Satoshi 的公钥为:", pubkey)

u = random.randint(1, N - 1)

v = random.randint(1, N - 1)

R = add\_point(mul\_point(u, G), mul\_point(v, pubkey), P)

r = R[0] % N

s = (r \* calc\_inverse(v, N)) % N

sign = (r, s)

print("生成的签名为:", sign[0])

e = (r \* u \* calc\_inverse(v, N)) % N

r\_ = (mul\_point(calc\_inverse(s, N), add\_point(mul\_point(e, G), mul\_point(r, pubkey), P)))[0] % N

print("伪造的签名问:", r\_)

if r\_ == r:

print("签名伪造成功！")

else:

print("伪造失败！")```

额外传入了参数e，直接用e进行验证，而不是传入消息m，用消息m重新计算e再验证。

### 2，运行指导

直接运行代码即可，能够输出伪造的签名，以及该伪造签名是否通过验证。

### 3，运行结果

运行程序结果如下：