

《网络空间安全创新创业实践》

实验报告

（2022~2023学年第2学期）

学院： 网络空间安全学院

班级： 21级密码1班

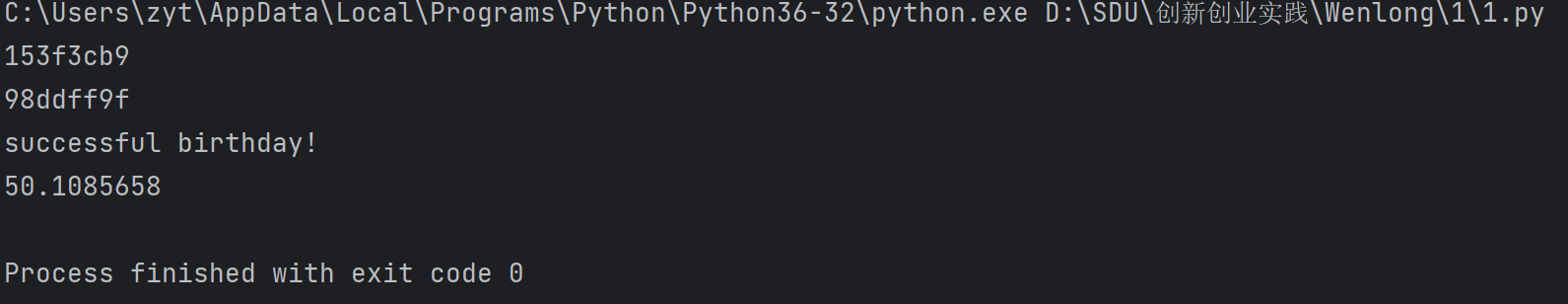
黄仲禹 202100460026

张艺腾 202100460001

张昊宇 202100460084

## Project1 implement the naïve birthday attack of reduced SM3

**运行结果:**

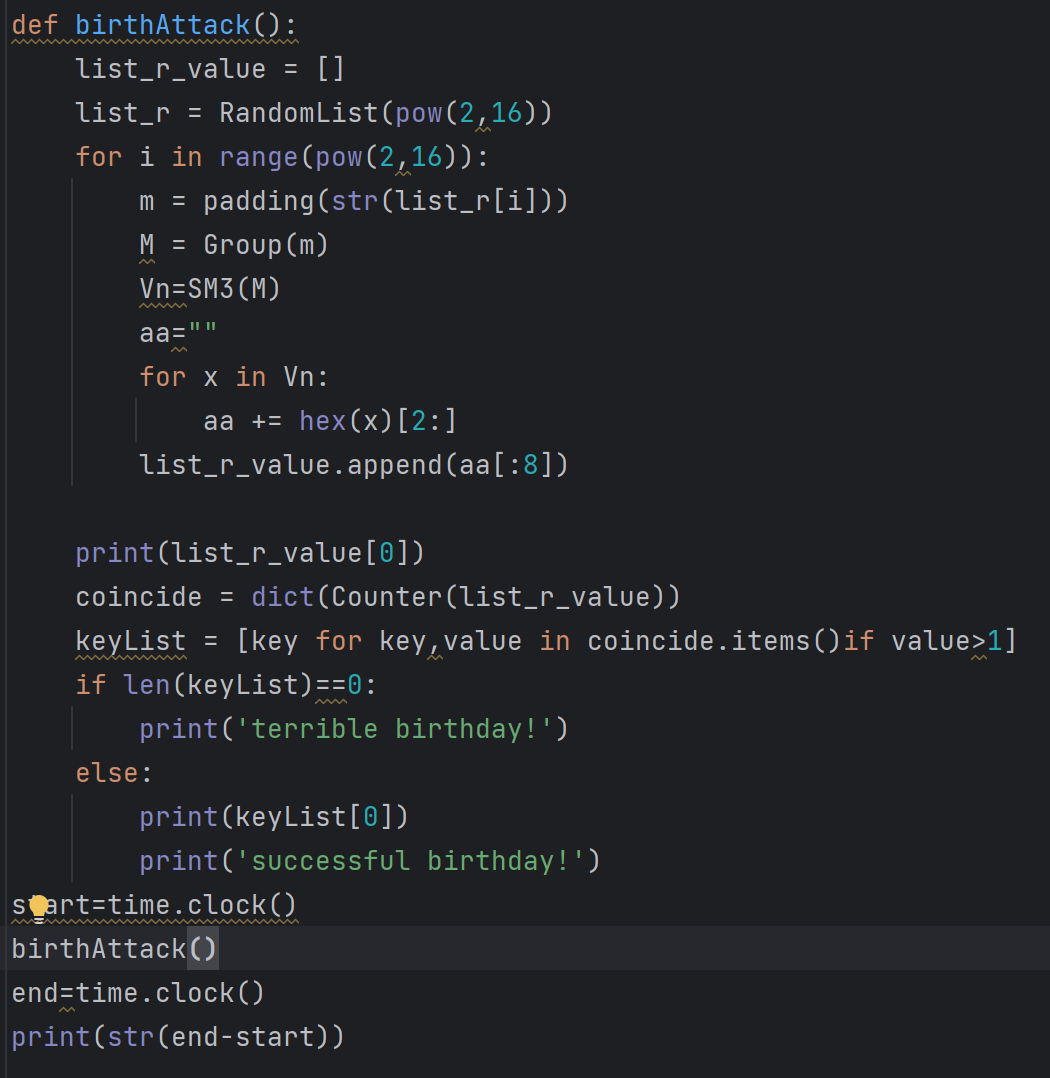


**运行时间: 50.1085658s**

**CPU型号:** **12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

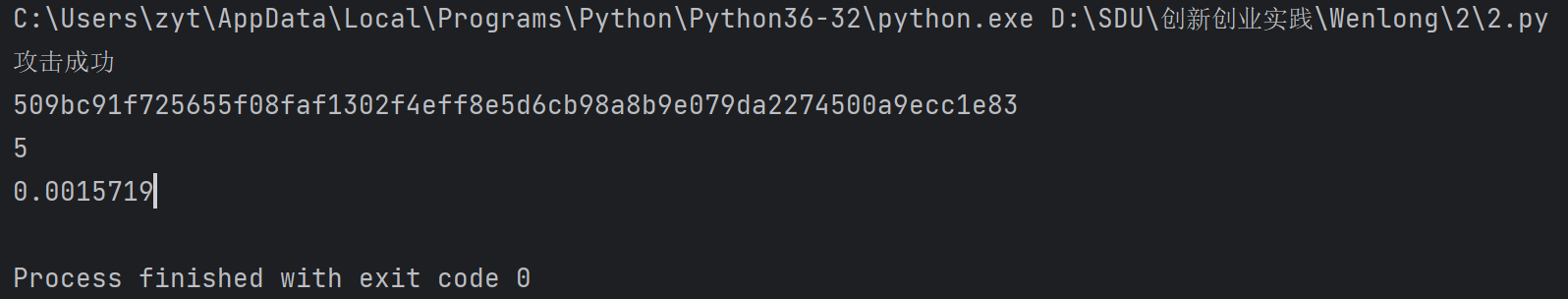
生日攻击是利用概率论中的生日问题，找到冲突的Hash值，伪造报文，使身份验证算法失效。如果输出是256位，我们随机地选择输入，并计算哈希值，在检验第2^256+1个输入之前便很可能找到碰撞。仅仅通过检验可能输出数量的平方根次数，便大体能找到碰撞。

实验中我们利用字典遍历搜索与目标hash值相同的值来构造碰撞。

.

# Project2 implement the Rho method of reduced SM3

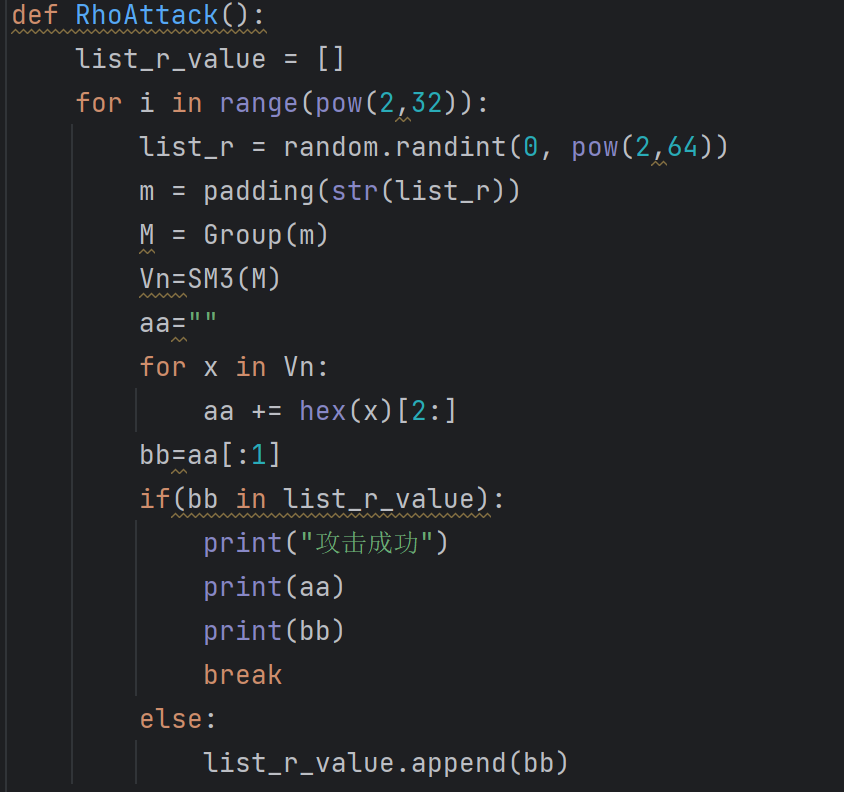
**代码结果:**



**运行时间:0.0015719s**

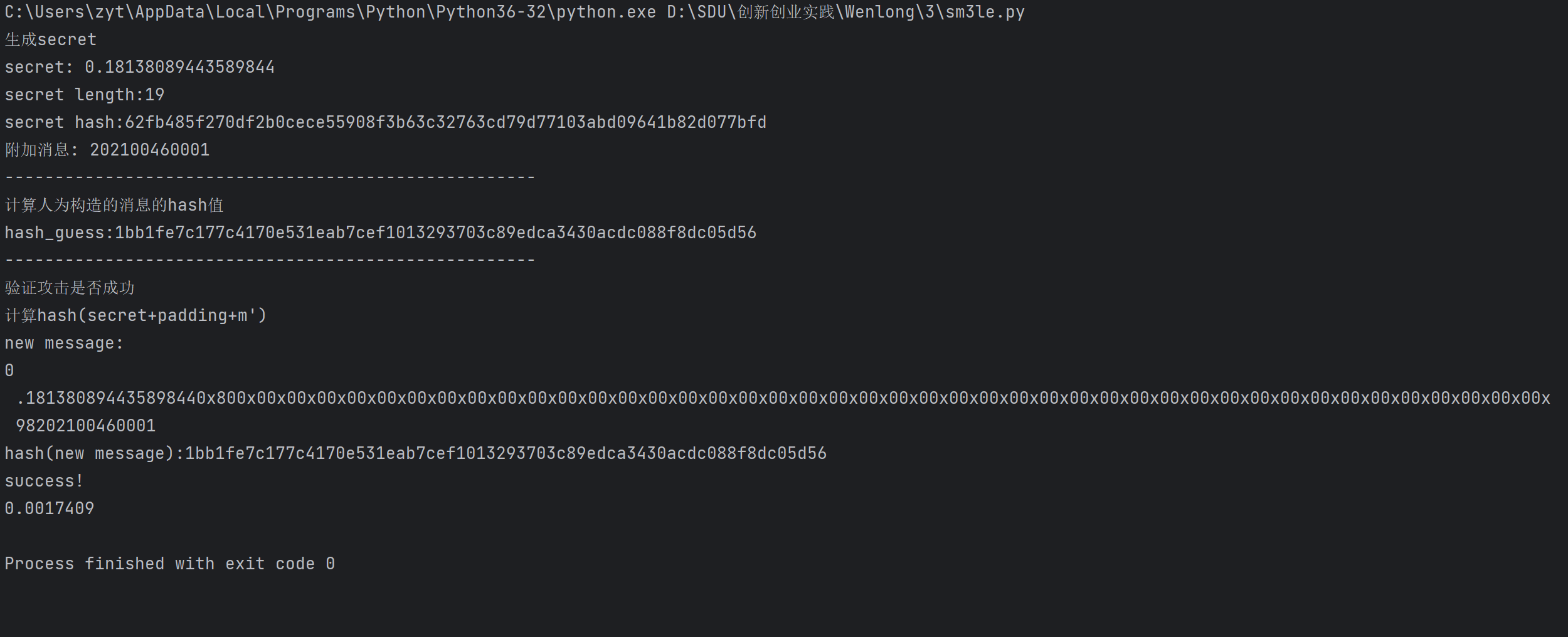
**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

Rho attack将每次hash的结果都放入一个数组中，之后的每次hash都遍历数组，如果结果在数组中能够找到，说明攻击成功，如果没在数组中，则将此次hash结果放入数组并继续循环直到能在数组中找到结果.



**Project3 implement length extension attack for SM3, SHA256, etc.**

**实验结果:**



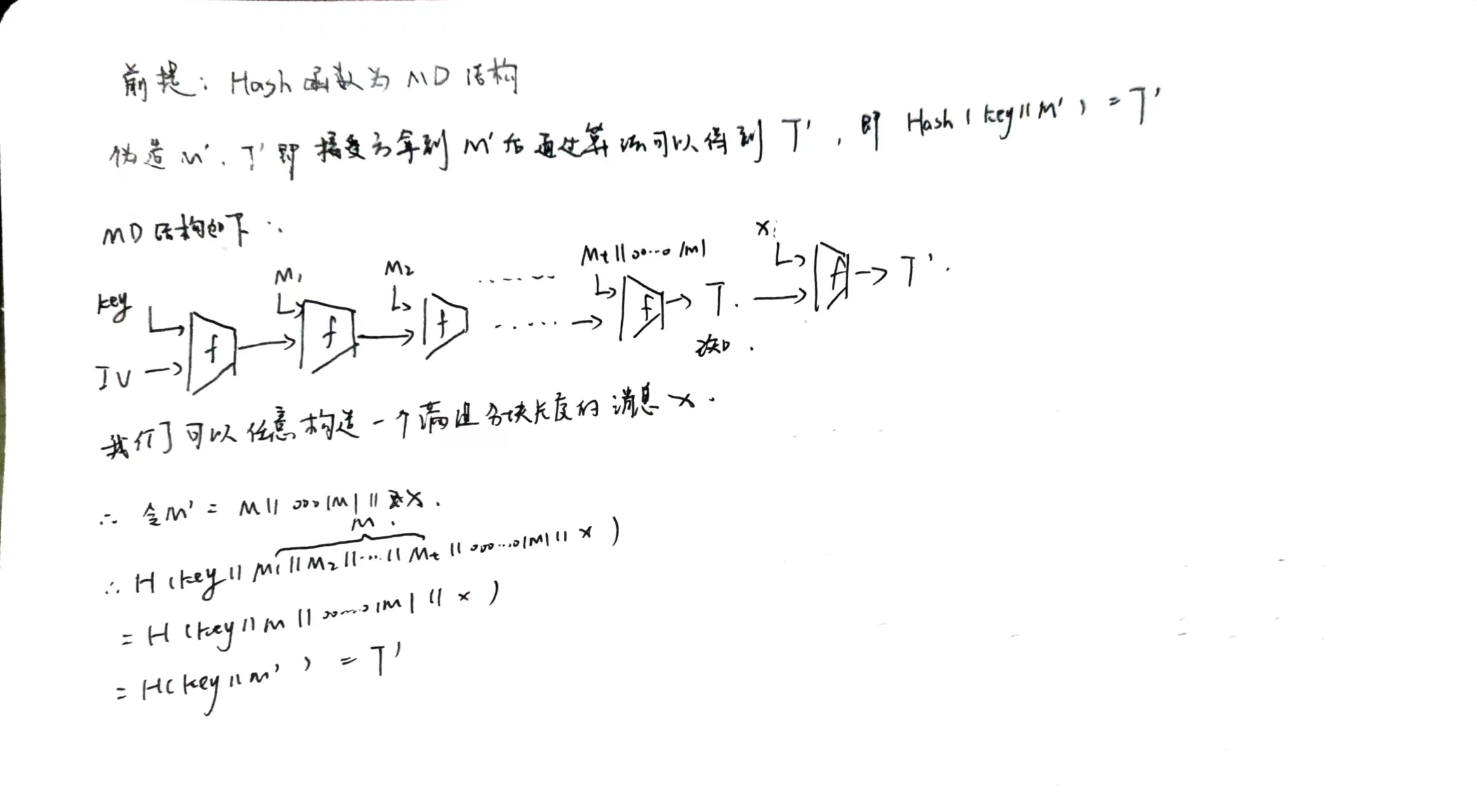
**运行时间:0.0017409s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

长度扩展攻击（length extension attack），是指针对某些允许包含额外信息的加密散列函数的攻击手段。对于满足以下条件的散列函数，都可以作为攻击对象：

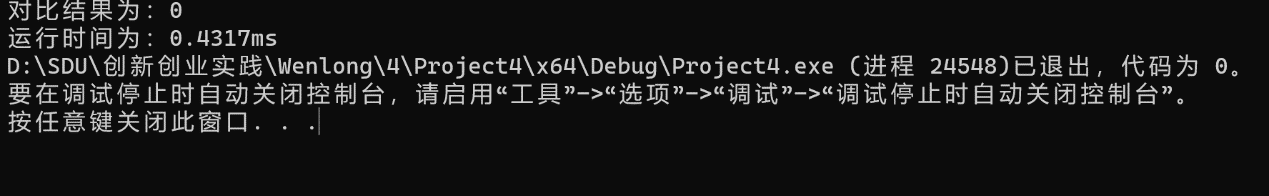
1、加密前将待加密的明文按一定规则填充到固定长度（例如512或1024比特）的倍数；

2、 按照该固定长度，将明文分块加密，并用前一个块的加密结果，作为下一块加密的初始向量（IV）。



**Project4 do your best to optimize SM3 implementation (software)**

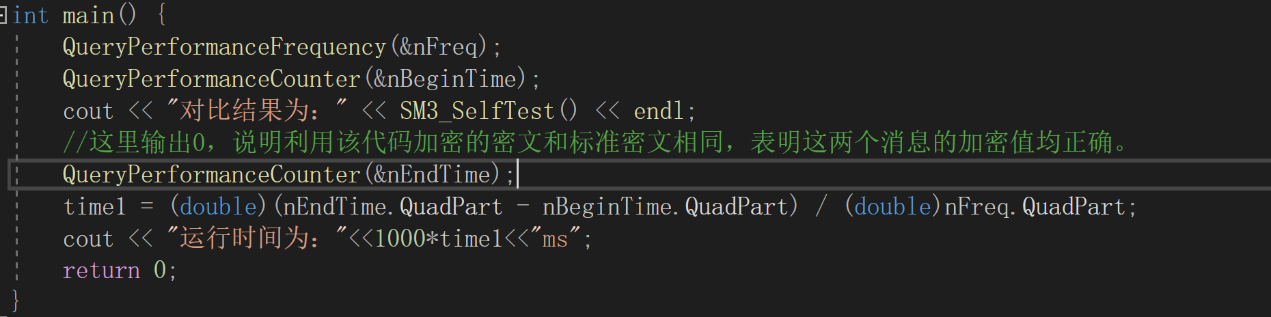
**运行结果：**

****

**运行时间：0.4317ms**

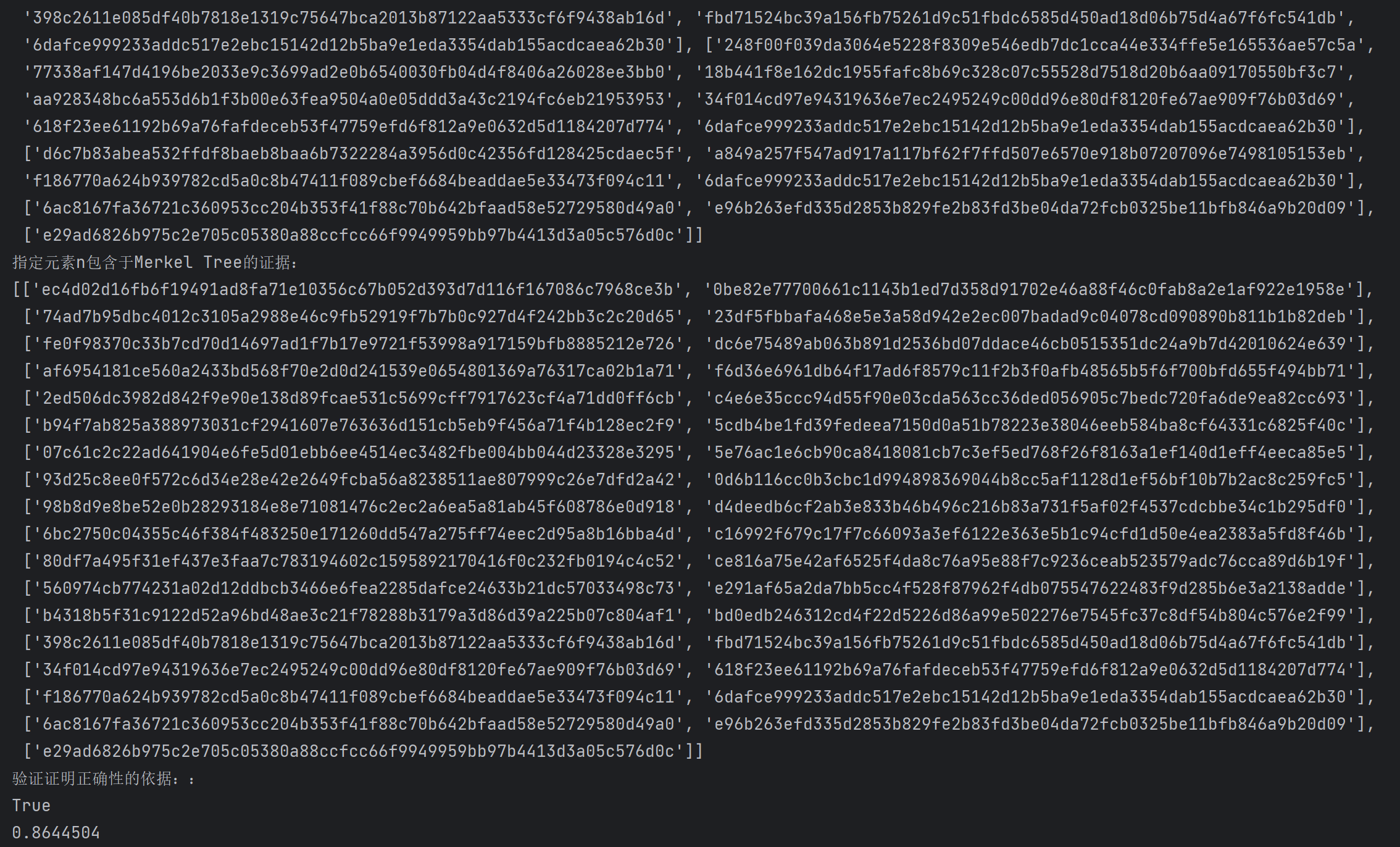
**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

结果说明：输出结果为0则代表该代码加密的密文和标准密文相同，表明这两个消息的加密值均正确。



**Project5: Impl Merkle Tree following RFC6962**

**运行结果：**



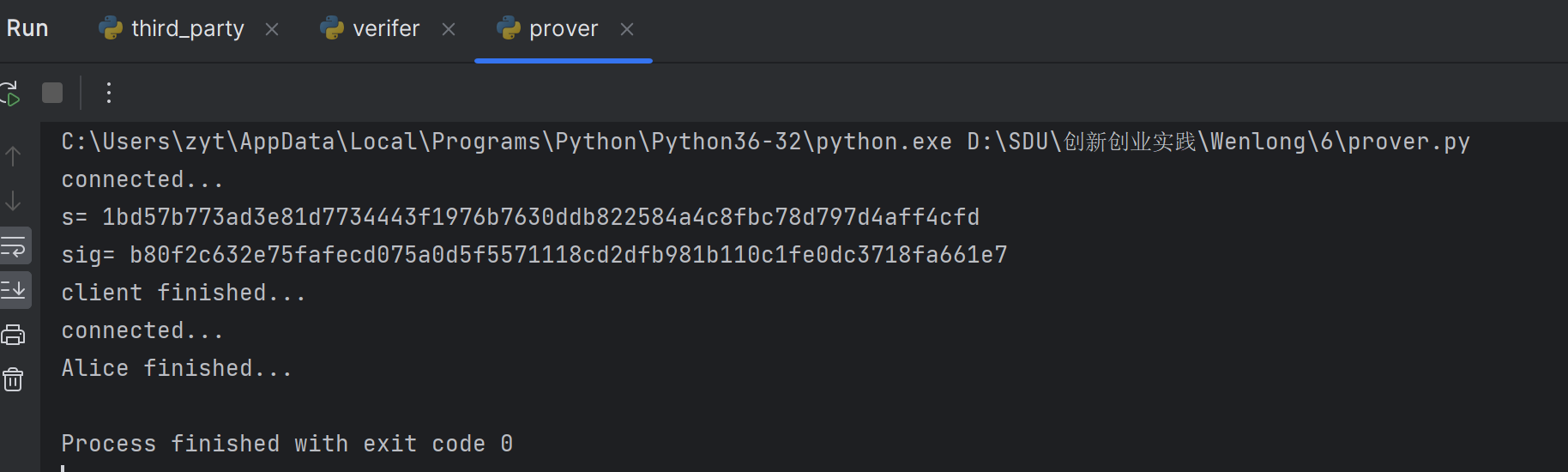
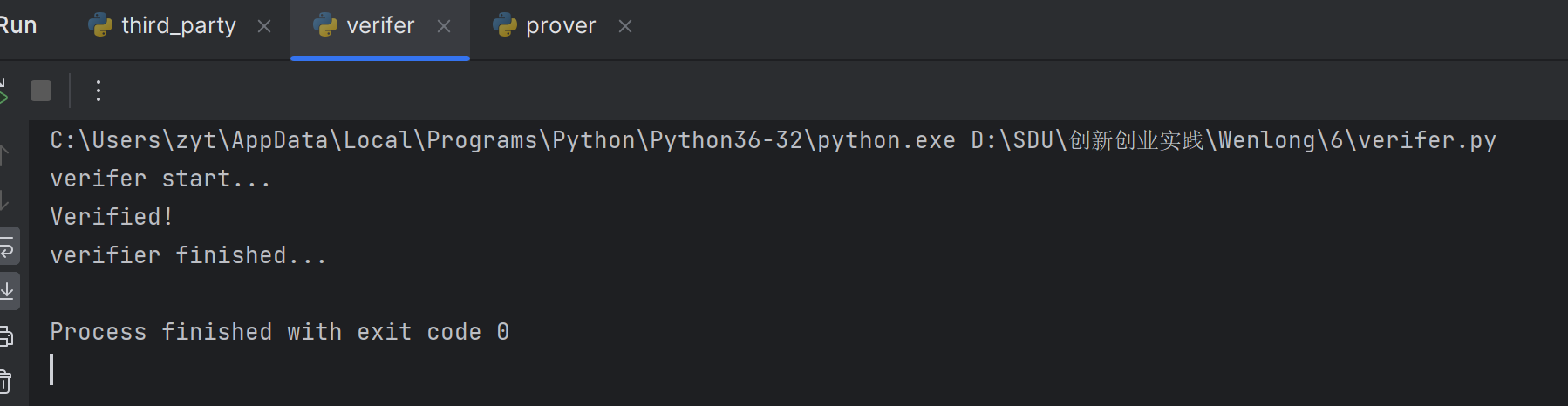
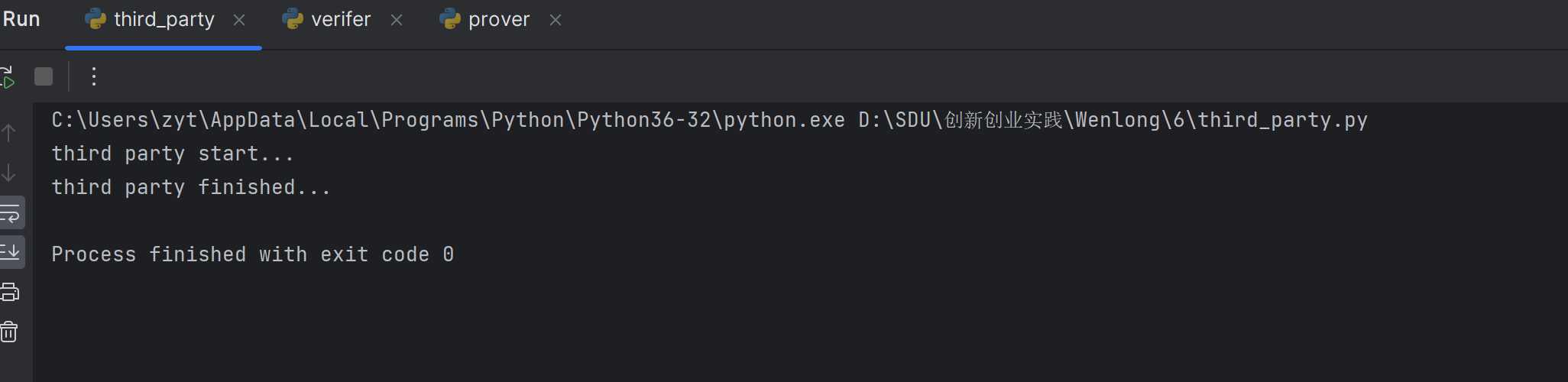
**运行时间：0.8644504s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

思路：首先初始化一个二维列表用于存放我们的Merkel tree，计算树的深度和叶子节点的个数，接着计算数据哈希值并写入叶子节点，每两个子节点计算相加后的哈希值并写入父节点列表。 而对于同一层的节点可以重复调用这个过程，生成下一层（父节点层）Merkle树的节点，每层向上生成父节点的时候，需要讨论对于节点数为奇数的层的最后一个节点，直接写入下一层（父节点层）；节点数为偶数则正好配对完全，进行递归步骤(3)和(4)的过程，循环步骤(1)计算的树的深度，完成Merkle树的生成过程；进行实验测试：输入测试数据，调用Tree\_Generate()函数将整个Merkle树printf出来，相同深度的node位于同一个列表中。

**Proje****ct6: impl this protocol with actual network communication**

**运行结果：**

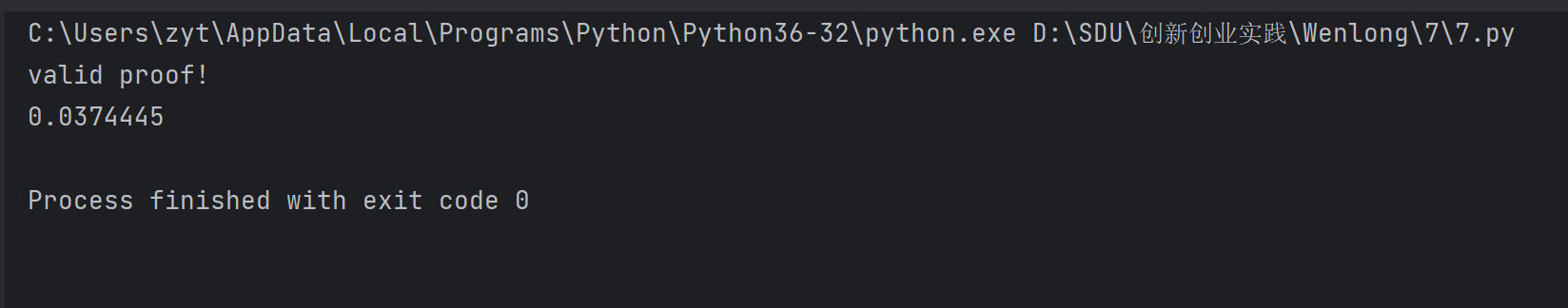


**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

运行说明：依次运行third\_party,verifier,prover。

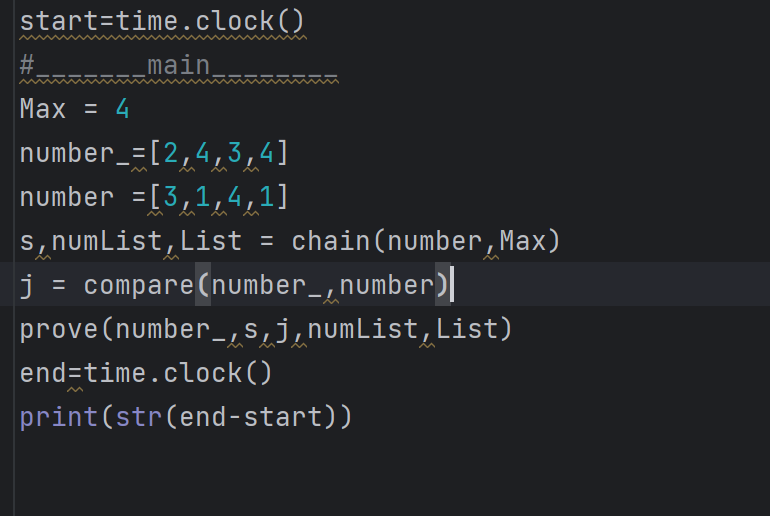
**Project7: Try to Implement this scheme**

**运行结果：**



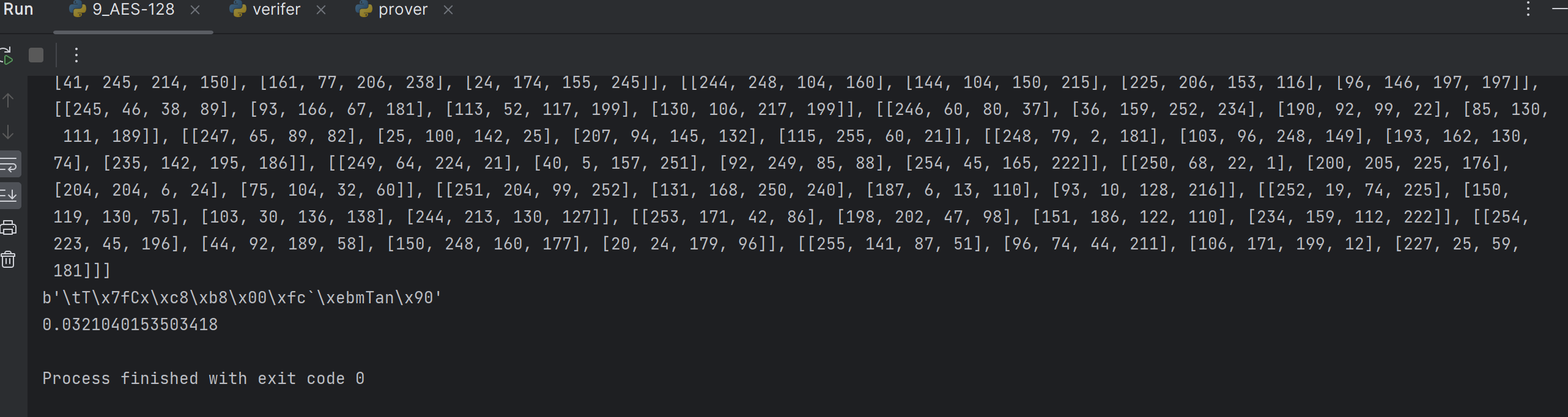
**运行时间：0.0374445s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**



**Project9: AES / SM4 software implementation**

**运行结果：**

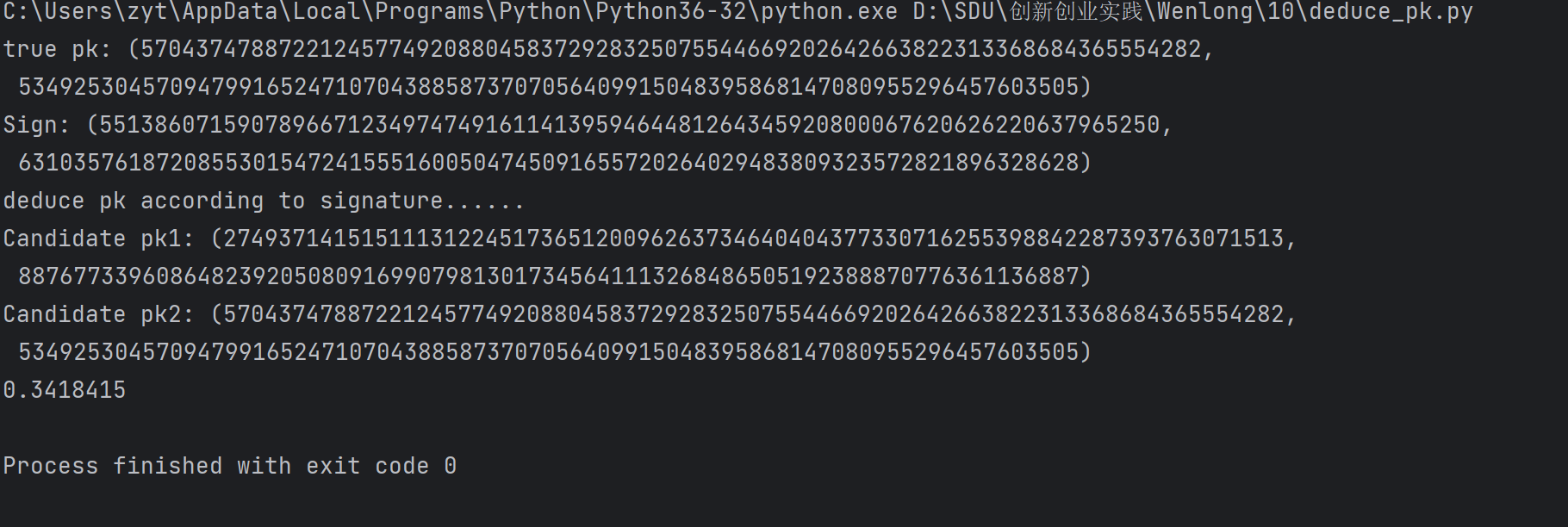
****

**运行时间：0.0321040153503418s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

结果展示了明文加密后的结果。

**Project10: report on the application of this deduce technique in Ethereum with ECDSA**

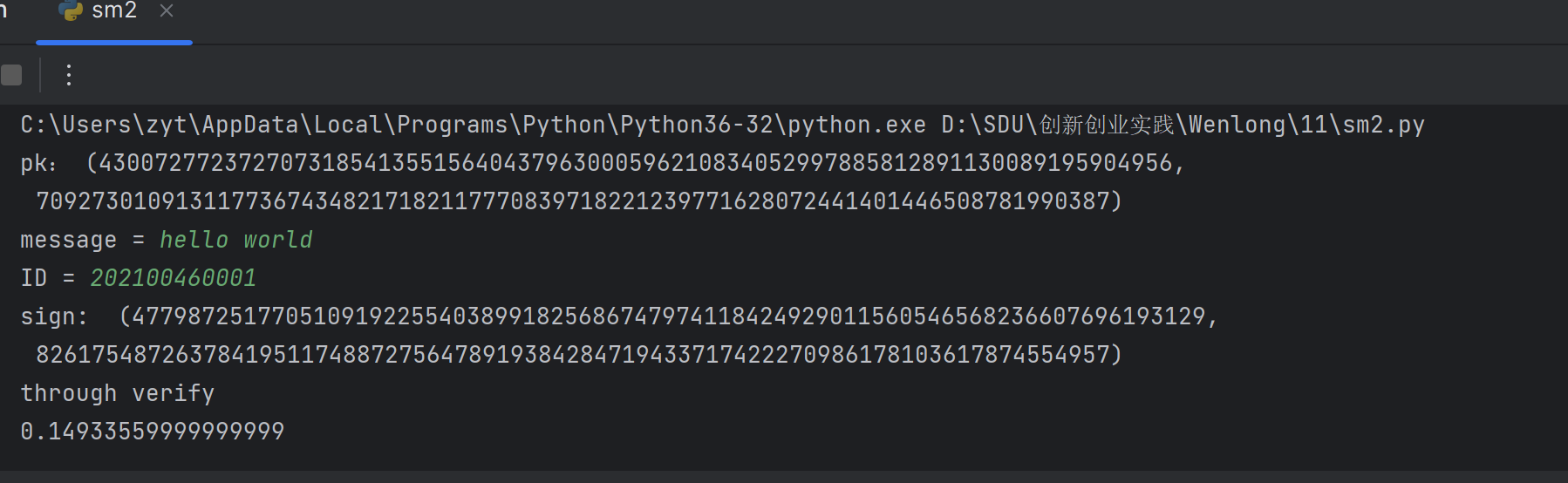
****

**运行时间：0.3418415s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

**Project11: impl sm2 with RFC6979**

**运行结果：**



**运行时间：0.14933559999999999s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

实现SM2数字签名的理论依据

密钥生成算法

Alice选择随机数dA做为私钥，其中0

Alice计算公钥PA=dA⋅G

输出密钥对 (sk=dA, pk=PA) 签名算法

设Alice发签名消息M给Bob，IDA是Alice的标识符，ENTLA是IDA的长度，dA是A的私钥，基点G= (xG, yG)，A的公钥PA=dA ⋅ G= (xA, yA).。

ZA=H (ENTLA ‖IDA ‖a‖b‖ xG ‖ yG ‖ xA ‖ yA)， H是SM3算法

①设置M\*=ZA ‖M并计算 e = H(M\*)

②产生随机数k∈[1, n-1]

③计算椭圆曲线点G1=k ⋅ G= (x1, y1)

④计算r=(e+x1) mod n，若r=0或r+k=n则返回②

⑤计算s=(1+ dA)−1·(k −r ·dA)mod n，若s=0则返回②

⑥以(r, s)作为对消息M的签名

验证算法

接收到的消息为M′，签名为(r′, s′)和发送者Alice的公钥PA，Bob执行如下步骤验证合法性：

检验r′∈[1, n-1]是否成立，若不成立则验证不通过

检验s′∈[1, n-1]是否成立，若不成立则验证不通过

设置M\*=ZA‖M′

计算e′= H(M\* )

计算t= (r′ + s′) mod n，若t=0，则验证不通过

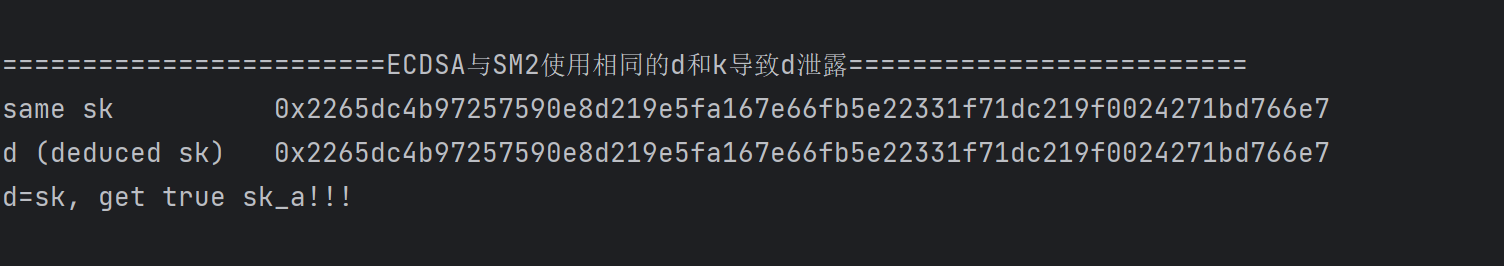
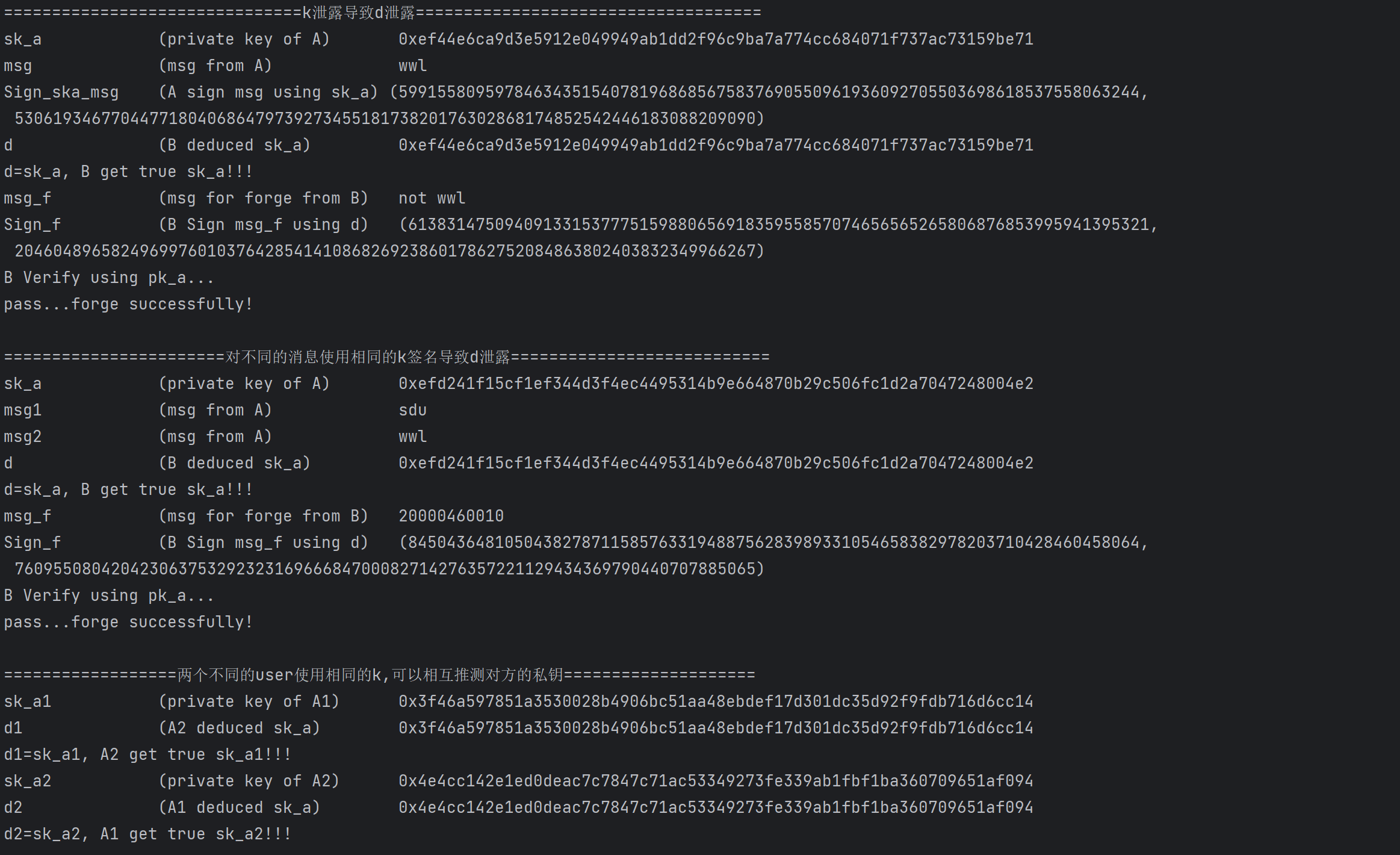
计算椭圆曲线点 (x1′，y1′)= s′ · G + t · PA

计算v=(e′+ x1′) mod n，检验v=r′是否成立，若成立则验证通过；否则验证不通过

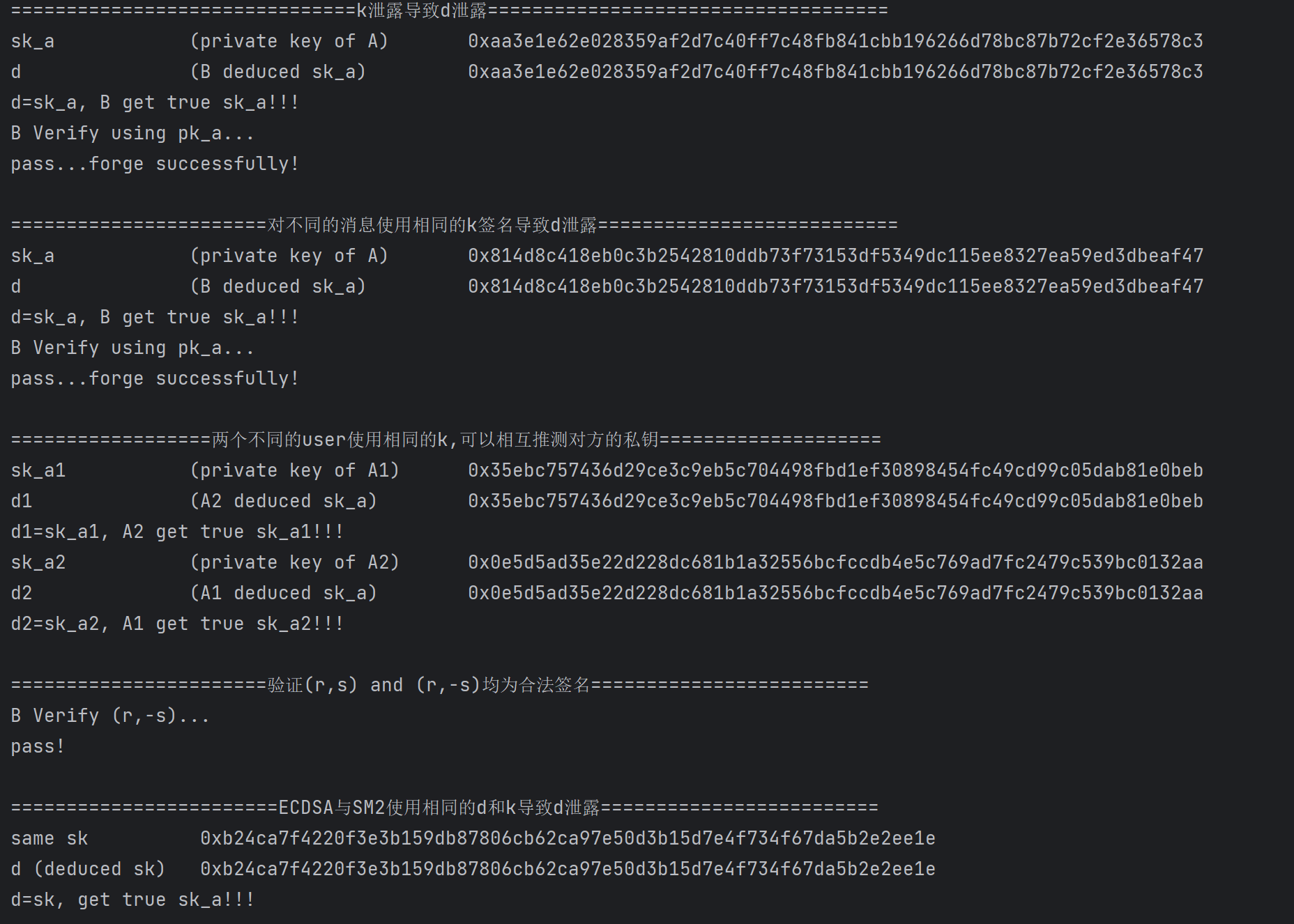
**Project12: verify the above pitfalls with proof-of-concept code**

**运行结果：**

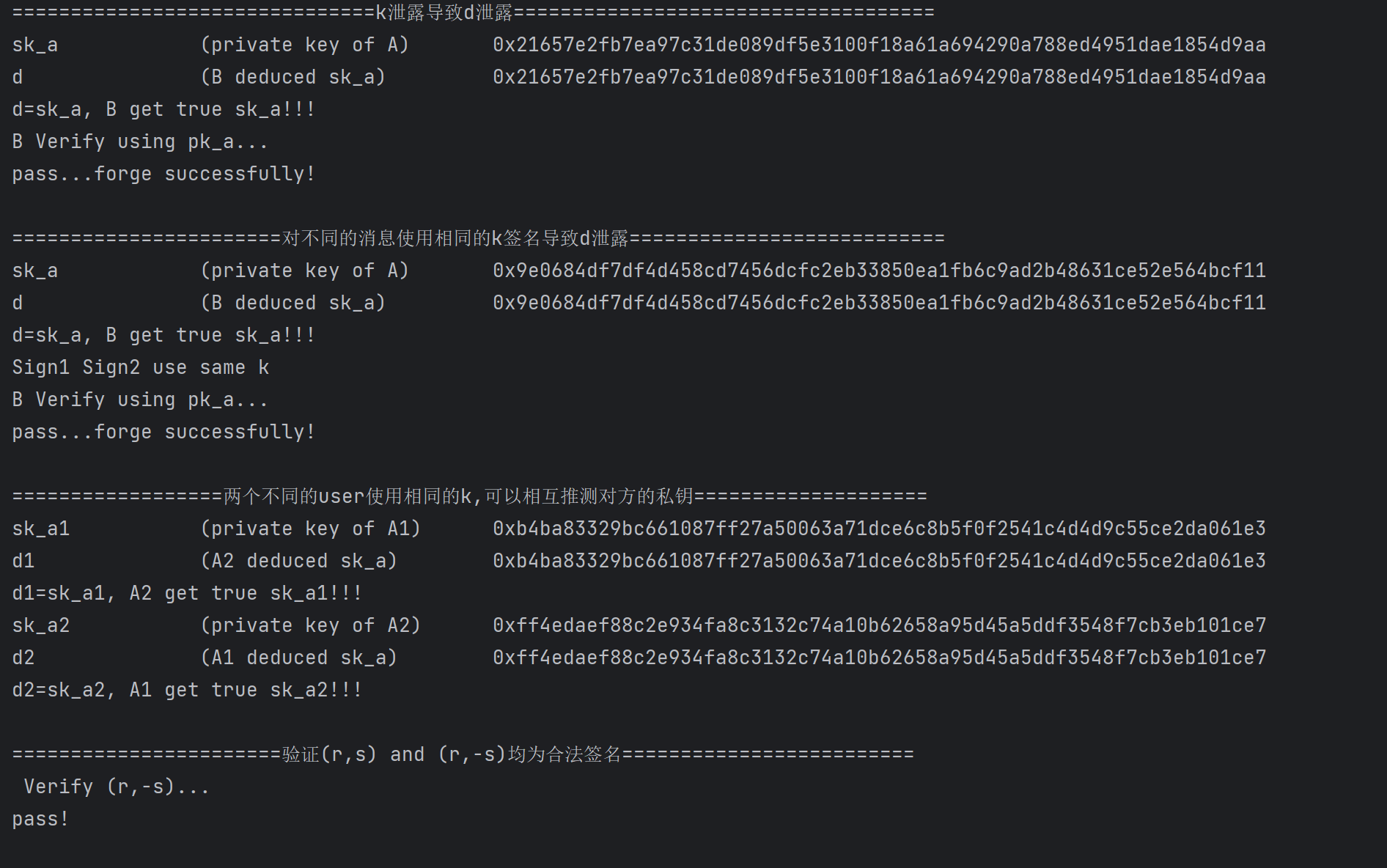
**sm2\_pitfalls：**



**Schnorr\_pitfalls ：**

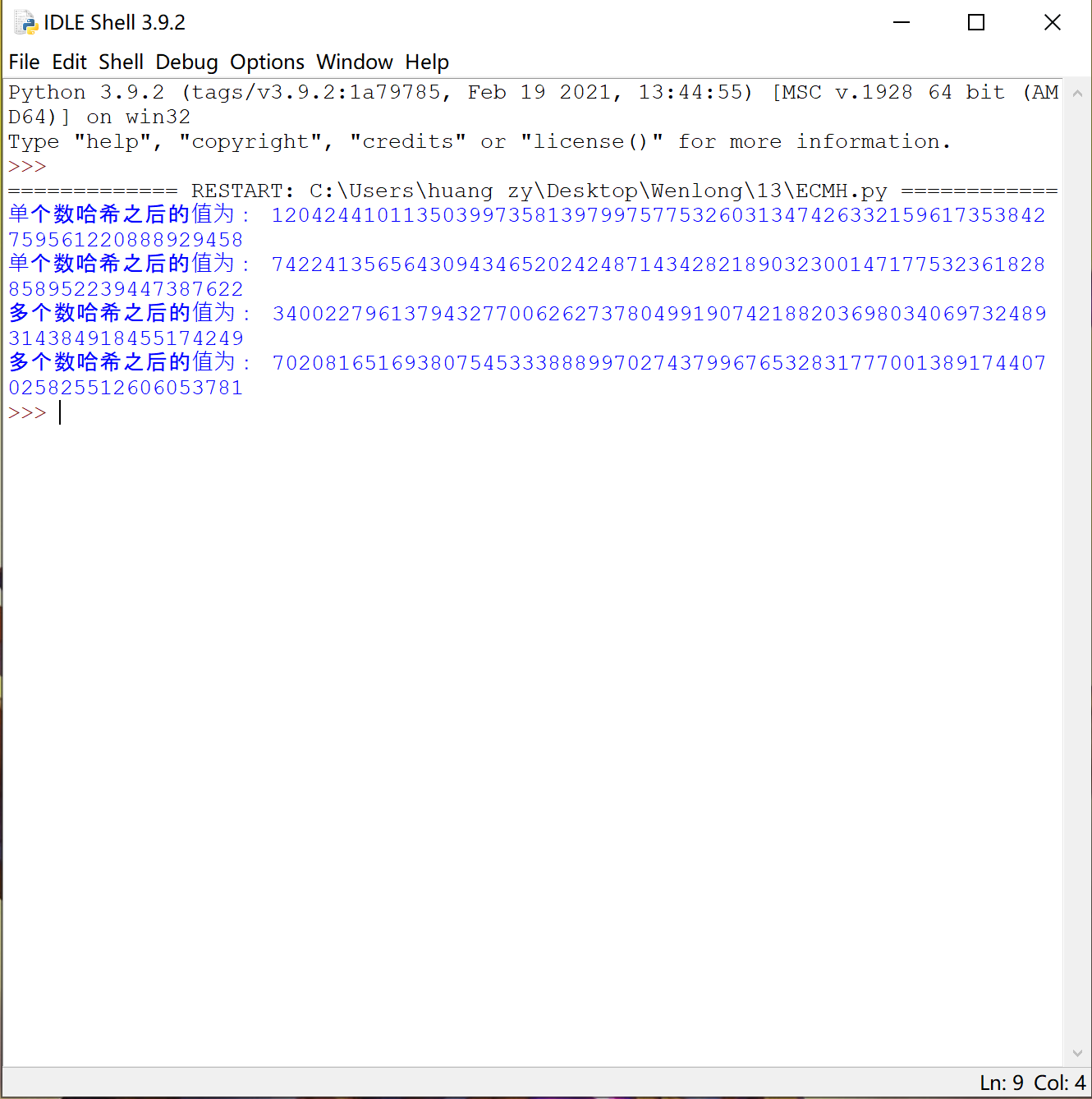


**ECDSA\_pitfalls：**



**Project13: Implement the above ECMH scheme**

**运行结果：**

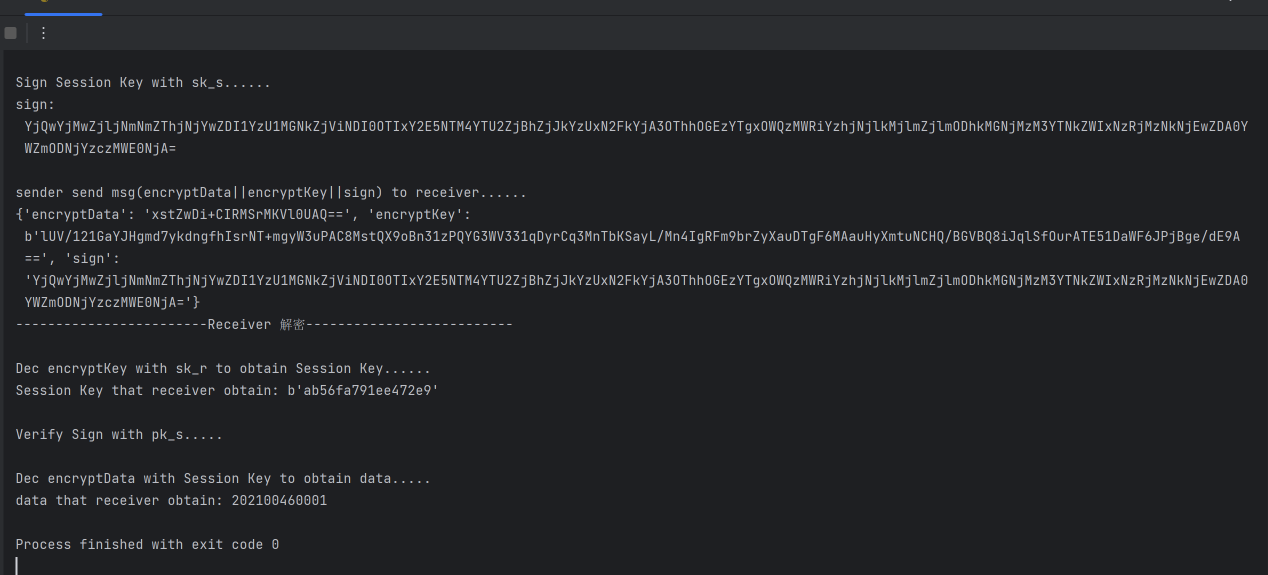
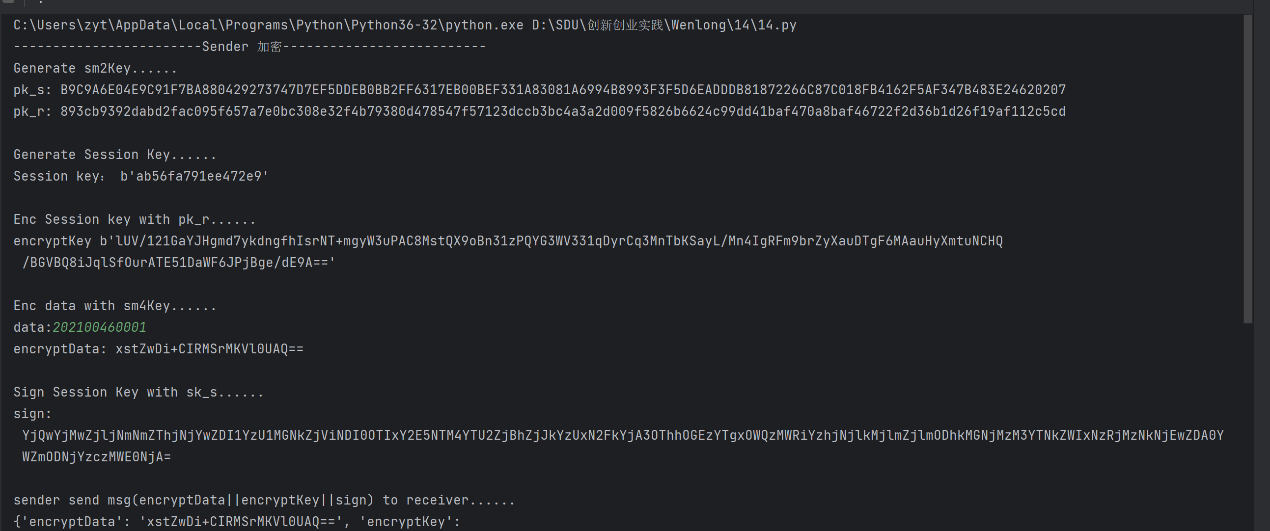


**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

该项目为ECMH方案的实现，将集合中的元素映射为椭圆曲线上的点，然后利用椭圆曲线上的加法求解哈希值。

**Project14:** **Implement a PGP scheme with SM2**

**运行结果：**

****

**运行时间：0.0415196s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

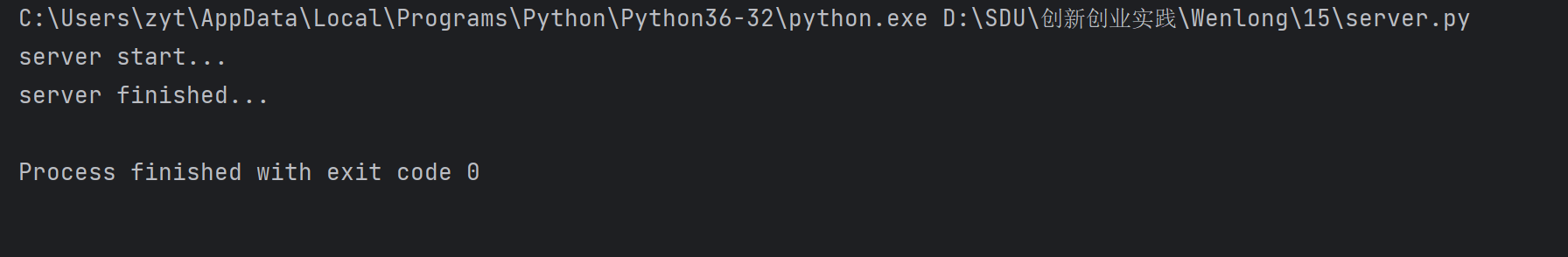
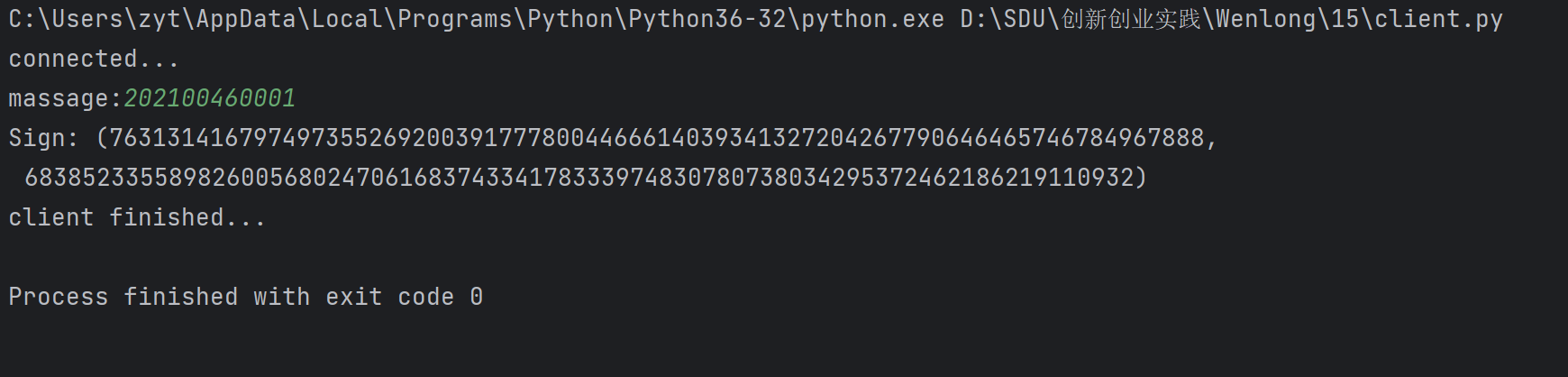
本次实验旨在实现一个简易PGP，调用GMSSL库中封装好的SM2/SM4加解密函数。

加密时使用对称加密算法SM4加密消息，非对称加密算法SM2加密会话密钥；

解密时先使用SM2解密求得会话密钥，再通过SM4和会话密钥求解原消息。

**Project15: implement sm2 2P sign with real network communication**

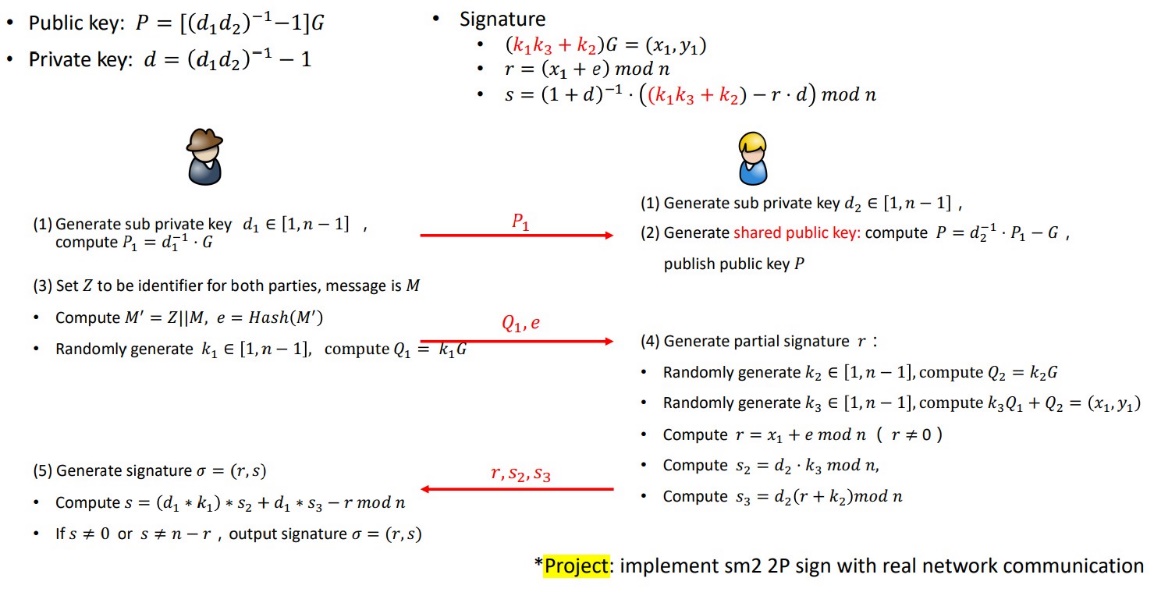
**运行结果：**

****

**运行时间：0.12121320000000001s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

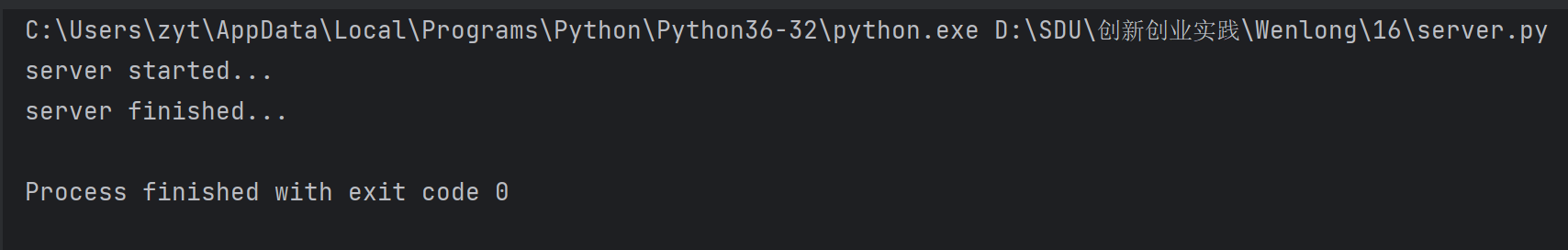
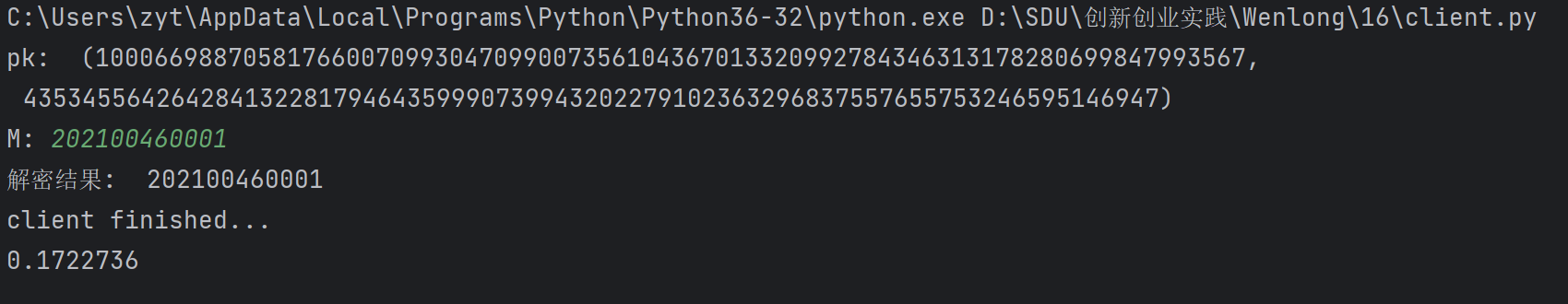
两方协同签名方案是指客户端与可信方服务器共同完成签名与验证的方案。 具体操作方式如图所示



通过这种协同验签方式，客户端保留单方私钥d1，服务器生成单方私钥d2，二者再生成协商私钥和协商公钥，这样一来单方不再能够生成合法的签名，双方协同生成签名，增加了签名的安全性，可信性。

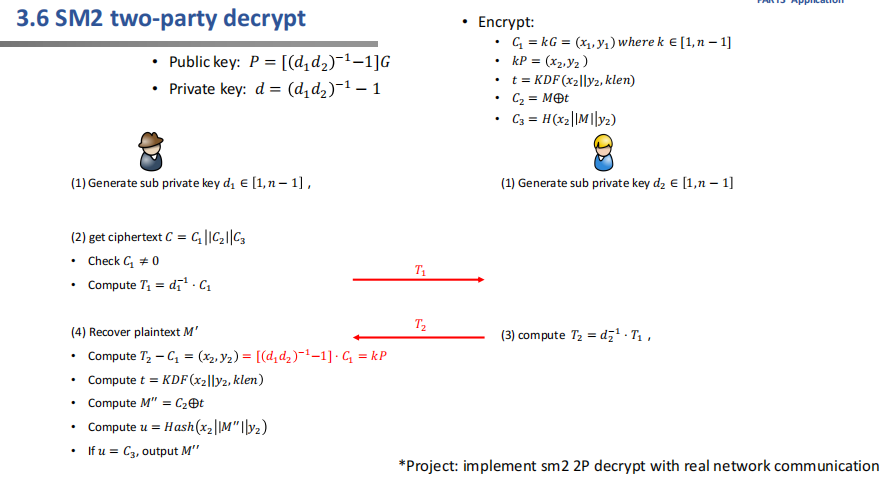
**Project16:** **implement sm2 2P decrypt with real network communication**

**运行结果：**

****

**运行时间：0.1722736s**

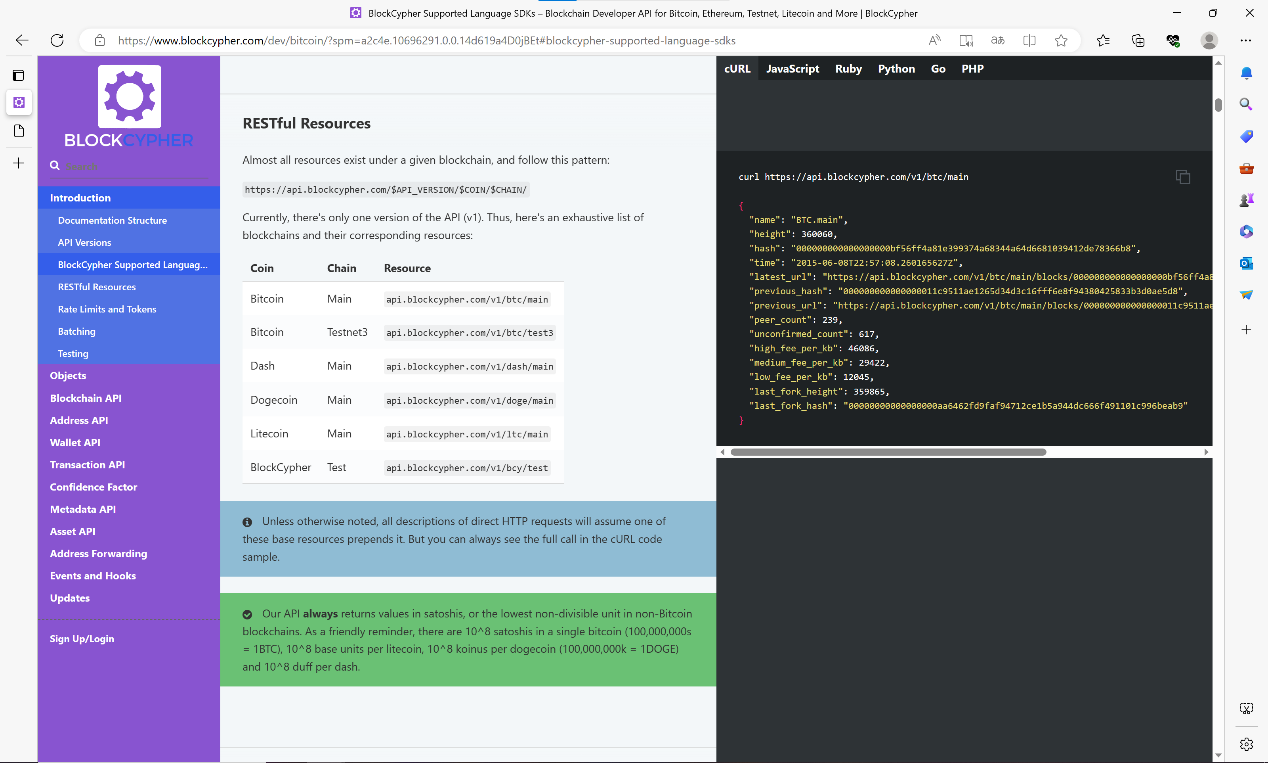
**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

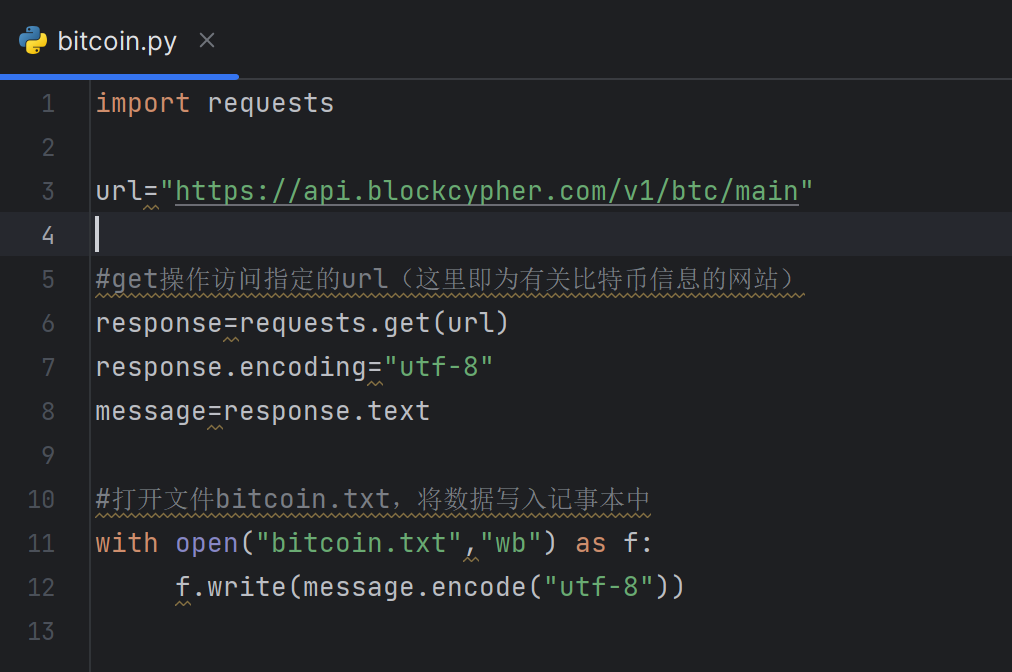


运行指导：这两个project都是先运行server再运行client即可

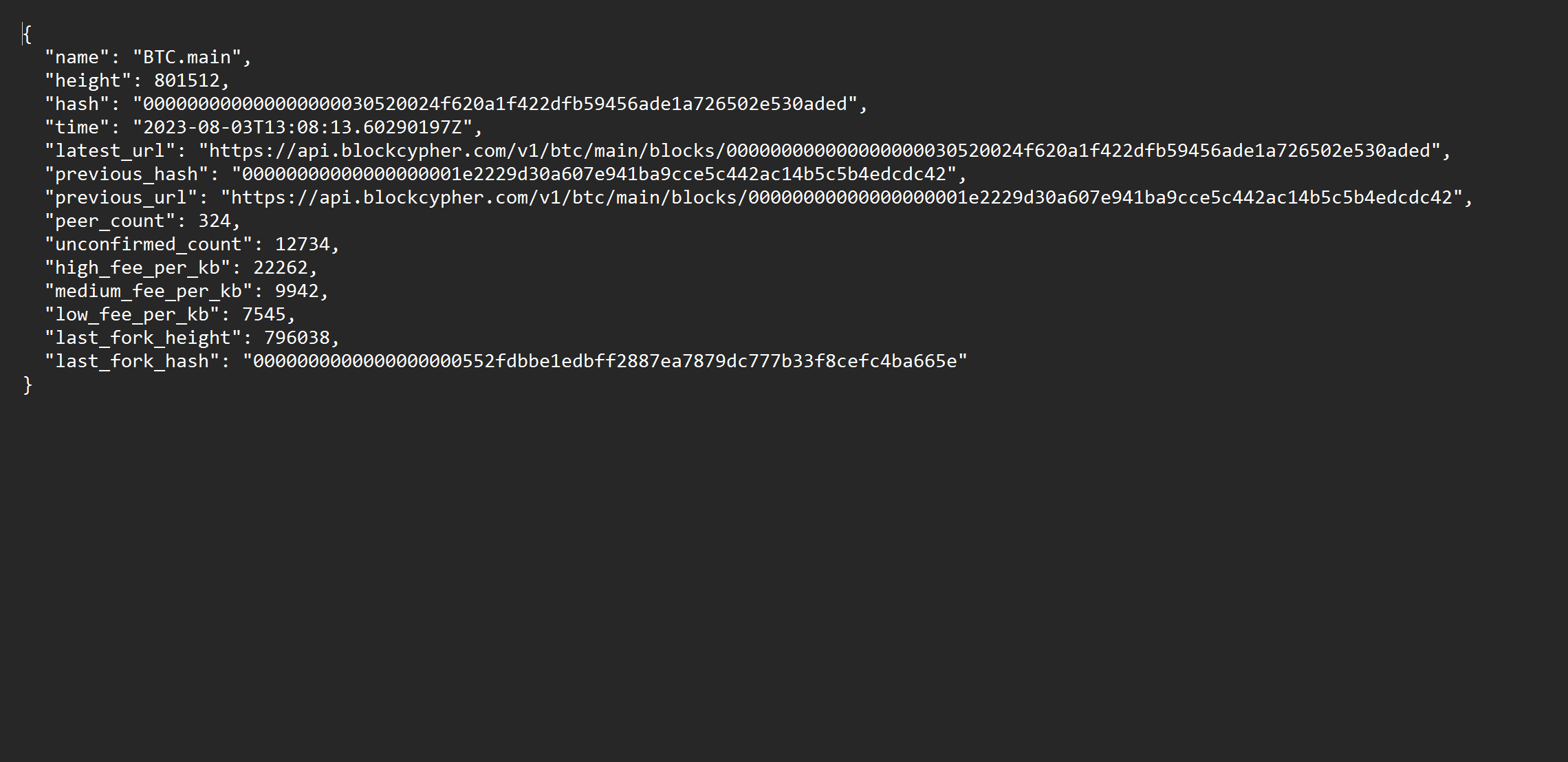
**Project18: send a tx on Bitcoin testnet, and parse the tx data down to every bit, better write script yourself**

**运行结果：**

****

****

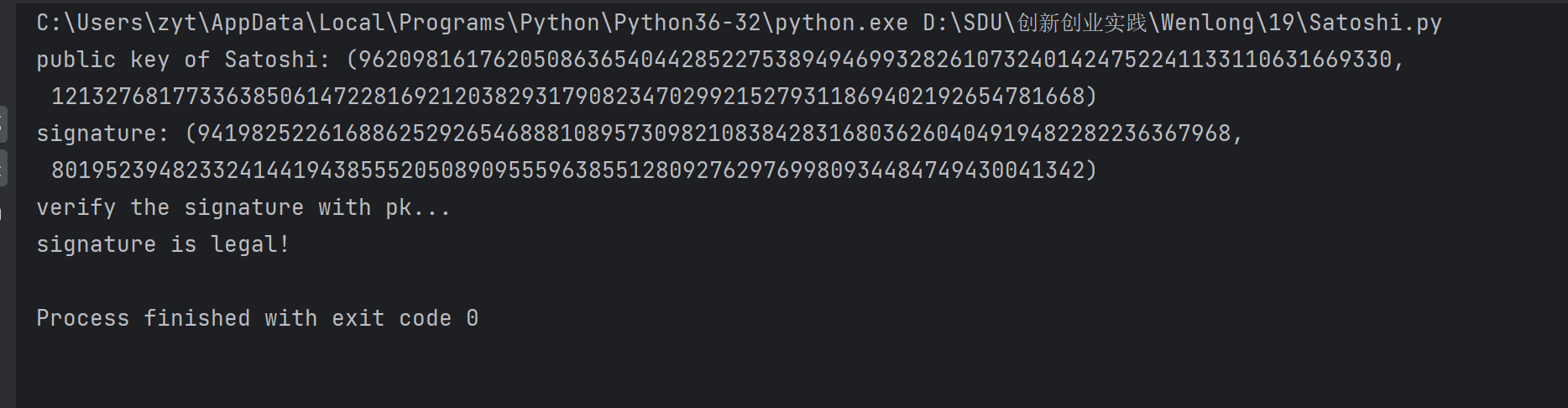
**数据：**

****

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

**Project19: forge a signature to pretend that you are Satoshi**

**运行结果：**

****

**运行时间：0.2774919s**

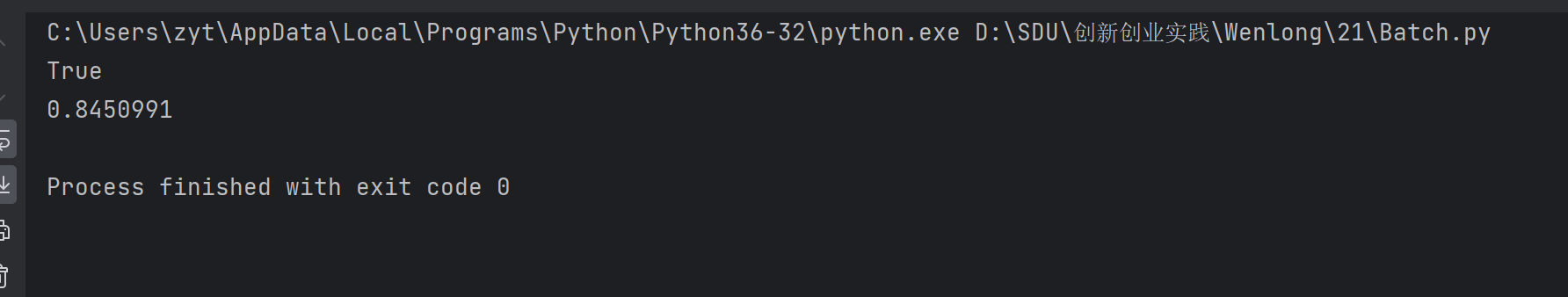
**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

**Project20: ECMH PoC**

**与project13类似**

**Project21: Schnorr Bacth**

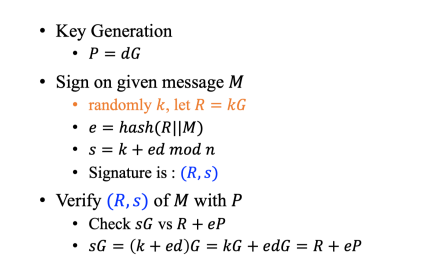
**运行结果：**

****

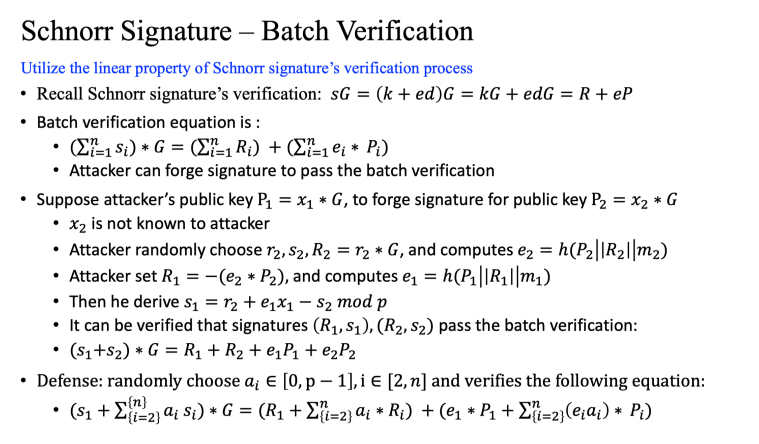
**运行时间：0.8450991s**

**CPU型号: 12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz**

**签名流程：**



**验签过程：**



**Project22: research report on MPT**

**详情请见GitHub仓库中的报告**