**海南大学网络空间安全学院**

**实 验 报 告**

|  |  |
| --- | --- |
| **评定成绩** | **指导教师** |
|  | 王文丽 |

|  |  |
| --- | --- |
| **实验课程** | 安全协议分析与设计 |
| **实验名称** | RSA密码系统的实现及其应用 |
| **学 号** | 20213006839 |
| **姓 名** | 甄五四 |
| **班 级** | 信息安全(密码学方向)理科实验班 |
| **完成日期** | 2024年6月5日 |

1. **实验目的**

1. 熟练掌握RSA密码算法及其签名算法。

2. 掌握交换律的概念。

3. 应用RSA密码算法。

4. 熟练掌握RSA批签名方案。

1. **实验任务**
2. 任选C或Python语言实现RSA密码算法，其中参数p,q的长度可调整，例如512bits,1024bits。

2. 验证具有相同模数的RSA密码算法是否满足交换律。

3. 利用RSA密码算法实现两方智力扑克游戏。

4. 编写RSA签名算法以及批签名方案(两个消息的批签名)。

1）实例化：令p,q均为512bits的参数，m1=4,随机选择一对公私钥(e，d)，计算其签名。

2）令m1=4,m2=7,随机选择两对公私钥(e1,d1),(e2,d2),计算其批签名，并从批签名中划分出单个消息m1,m2的签名。

1. **实验环境**

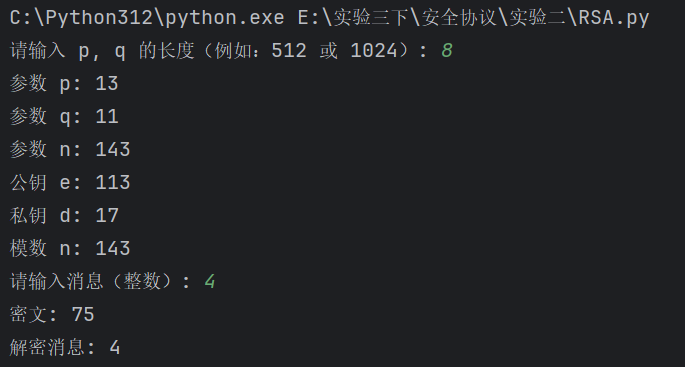
**Windows 10,Pycharm 2023,python 3.12.2**

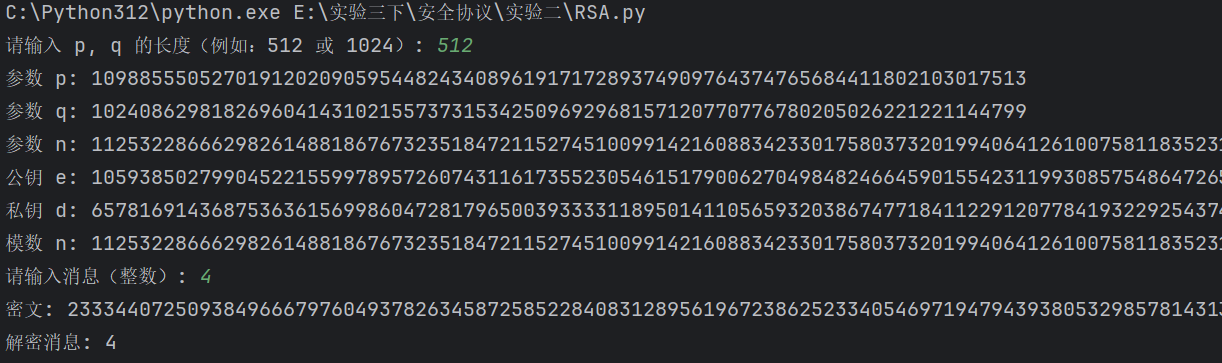
1. **实验内容及步骤(题目、相应程序代码和运行效果截图)**
2. Python语言实现RSA密码算法，其中参数p,q的长度可调整，例如512bits,1024bits。

完整代码RSA.py

import random  
def is\_prime(n, k=5): # Miller-Rabin素性检测  
 if n <= 1:  
 return False  
 if n <= 3:  
 return True  
 if n % 2 == 0:  
 return False  
 r, s = 0, n - 1  
 while s % 2 == 0:  
 r += 1  
 s //= 2  
 for \_ in range(k):  
 a = random.randint(2, n - 2)  
 x = pow(a, s, n)  
 if x == 1 or x == n - 1:  
 continue  
 for \_ in range(r - 1):  
 x = pow(x, 2, n)  
 if x == n - 1:  
 break  
 else:  
 return False  
 return True  
  
def generate\_prime(bits): # 生成指定比特长度的素数  
 while True:  
 p = random.getrandbits(bits)  
 p |= (1 << bits - 1) | 1  
 if is\_prime(p):  
 return p  
  
def generate\_p\_q\_n(bits): # 生成 RSA 的 p, q, n  
 p = generate\_prime(bits // 2)  
 q = generate\_prime(bits // 2)  
 while p == q:  
 q = generate\_prime(bits // 2)  
 n = p \* q  
 return p, q, n  
  
def gcd(a, b): # 计算最大公约数  
 while b:  
 a, b = b, a % b  
 return a  
  
def modinv(a, m): # 计算模逆  
 m0, x0, x1 = m, 0, 1  
 if m == 1:  
 return 0  
 while a > 1:  
 q = a // m  
 m, a = a % m, m  
 x0, x1 = x1 - q \* x0, x0  
 if x1 < 0:  
 x1 += m0  
 return x1  
  
def generate\_e\_d(phi): # 生成 RSA 的 e, d  
 e = random.randrange(2, phi)  
 while gcd(e, phi) != 1:  
 e = random.randrange(2, phi)  
 d = modinv(e, phi)  
 return e, d  
def generate\_rsa\_keys(p,q,n,): # 生成 RSA 密钥对  
 phi = (p - 1) \* (q - 1)  
 e, d = generate\_e\_d(phi)  
 return (e, n), (d, n)  
  
def encrypt\_message(public\_key, message): # 使用公钥加密消息  
 e, n = public\_key  
 ciphertext = pow(message, e, n)  
 return ciphertext  
  
def decrypt\_message(private\_key, ciphertext): # 使用私钥解密消息  
 d, n = private\_key  
 plaintext = pow(ciphertext, d, n)  
 return plaintext  
if \_\_name\_\_ == "\_\_main\_\_":  
 # 生成密钥对（p, q 均为 512 位）  
 bits = int(input("请输入 p, q 的长度（例如：512 或 1024）: "))  
 p, q, n = generate\_p\_q\_n(bits)  
 print("参数 p:", p)  
 print("参数 q:", q)  
 print("参数 n:", n)  
 public\_key, private\_key = generate\_rsa\_keys(p, q, n)  
 e, n = public\_key  
 print("公钥 e:", e)  
 d, n = private\_key  
 print("私钥 d:", d)  
 print("模数 n:", n)  
 # 加密和解密消息  
 message = int(input("请输入消息（整数）: "))  
 ciphertext = encrypt\_message(public\_key, message)  
 print("密文:", ciphertext)  
 plaintext = decrypt\_message(private\_key, ciphertext)  
 print("解密消息:", plaintext)

运行结果：



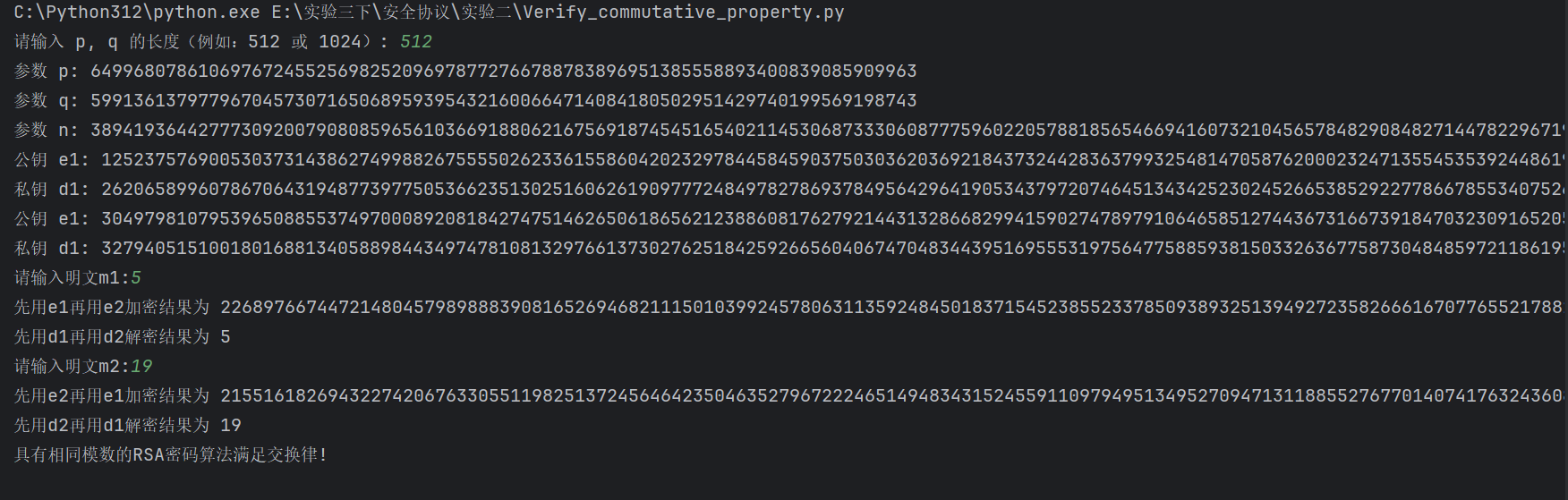


1. 验证具有相同模数的RSA密码算法是否满足交换律

完整代码Verify\_commutative\_property.py：

from RSA import generate\_p\_q\_n,generate\_rsa\_keys,encrypt\_message,decrypt\_message  
bits = int(input("请输入 p, q 的长度（例如：512 或 1024）: "))  
p,q,n =generate\_p\_q\_n(bits)  
print("参数 p:", p)  
print("参数 q:", q)  
print("参数 n:", n)  
public\_key1, private\_key1 = generate\_rsa\_keys(p,q,n)  
e1, n = public\_key1  
print("公钥 e1:", e1)  
d1, n = private\_key1  
print("私钥 d1:", d1)  
public\_key2, private\_key2 = generate\_rsa\_keys(p,q,n)  
e2, n = public\_key2  
print("公钥 e1:", e2)  
d2, n = private\_key2  
print("私钥 d1:", d2)  
m1=int(input("请输入明文m1:"))  
e2e1m1=encrypt\_message(public\_key2,encrypt\_message(public\_key1, m1))  
d2d1c1=decrypt\_message(private\_key2,decrypt\_message(private\_key1, e2e1m1))  
print("先用e1再用e2加密结果为",e2e1m1)  
print("先用d1再用d2解密结果为",d2d1c1)  
m2=int(input("请输入明文m2:"))  
e1e2m2=encrypt\_message(public\_key1,encrypt\_message(public\_key2, m2))  
d1d2c2=decrypt\_message(private\_key1,decrypt\_message(private\_key2, e1e2m2))  
print("先用e2再用e1加密结果为",e1e2m2)  
print("先用d2再用d1解密结果为",d1d2c2)  
if m1==d2d1c1 and m2==d1d2c2:  
 print("具有相同模数的RSA密码算法满足交换律!")

运行结果：

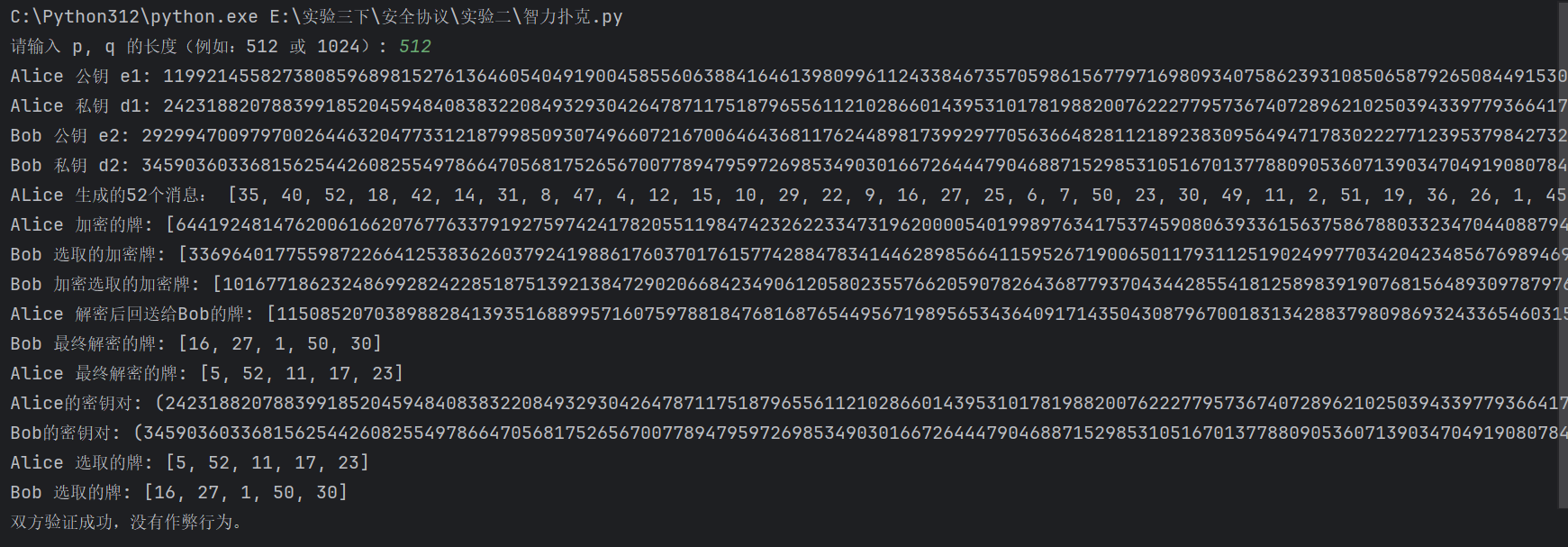


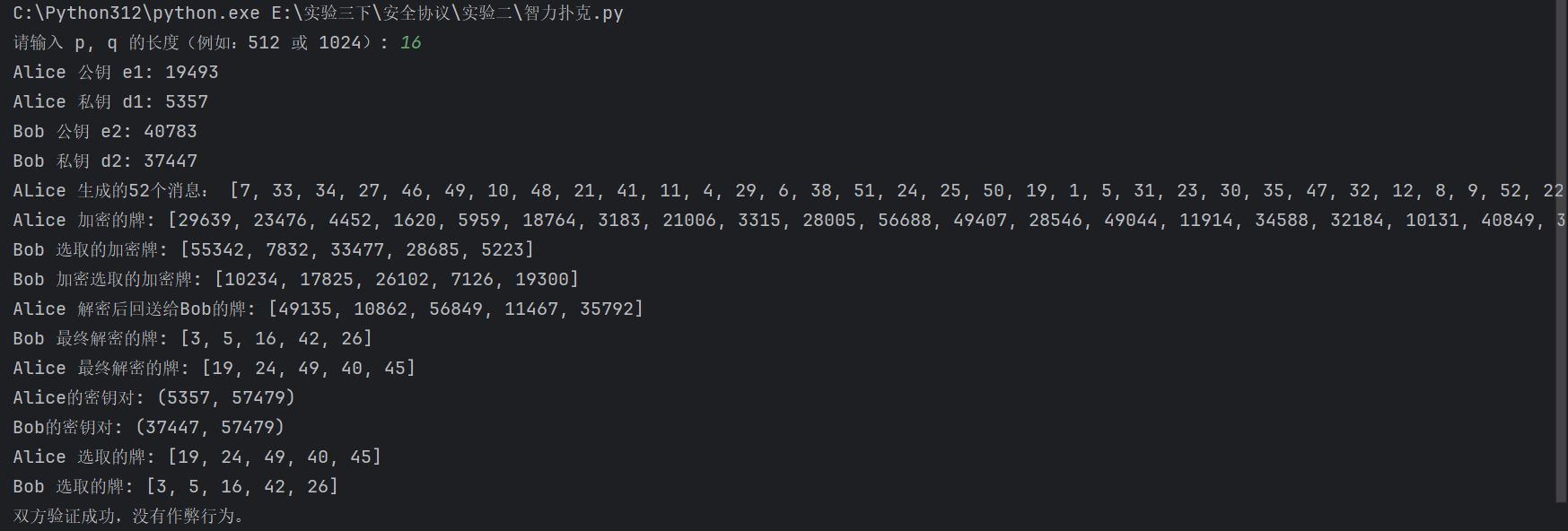
1. 利用RSA密码算法实现两方智力扑克游戏

完整代码：

from RSA import generate\_p\_q\_n, generate\_rsa\_keys, encrypt\_message, decrypt\_message  
import random  
# 生成Alice和Bob的公私钥对，使用相同的模数n  
bits = int(input("请输入 p, q 的长度（例如：512 或 1024）: "))  
p, q, n = generate\_p\_q\_n(bits)  
# Alice的公私钥  
public\_key\_alice, private\_key\_alice = generate\_rsa\_keys(p, q, n)  
e1, n = public\_key\_alice  
d1, n = private\_key\_alice  
print("Alice 公钥 e1:", e1)  
print("Alice 私钥 d1:", d1)  
# Bob的公私钥  
public\_key\_bob, private\_key\_bob = generate\_rsa\_keys(p, q, n)  
e2, n = public\_key\_bob  
d2, n = private\_key\_bob  
print("Bob 公钥 e2:", e2)  
print("Bob 私钥 d2:", d2)  
# 生成52张扑克牌的编号消息  
cards = list(range(1, 53))  
random.shuffle(cards) # 任意顺序发送给Bob  
print("ALice 生成的52个消息：", cards)  
# 1. Alice生成52个消息并加密  
encrypted\_cards\_alice = [encrypt\_message(public\_key\_alice, card) for card in cards]  
print("Alice 加密的牌:", encrypted\_cards\_alice)  
# 2. Bob随机选取5张牌并用Bob的公钥加密发送给Alice  
selected\_indices\_bob = random.sample(range(52), 5)  
selected\_encrypted\_cards\_bob = [encrypted\_cards\_alice[i] for i in selected\_indices\_bob]  
print("Bob 选取的加密牌:", selected\_encrypted\_cards\_bob)  
encrypted\_selected\_cards\_bob = [encrypt\_message(public\_key\_bob, card) for card in selected\_encrypted\_cards\_bob]  
print("Bob 加密选取的加密牌:", encrypted\_selected\_cards\_bob)  
# 3. Alice用Alice私钥解密消息并回送给Bob  
decrypted\_selected\_cards\_alice = [decrypt\_message(private\_key\_alice, card) for card in encrypted\_selected\_cards\_bob]  
print("Alice 解密后回送给Bob的牌:", decrypted\_selected\_cards\_alice)  
# 4. Bob用Bob私钥解密以确定他的一手牌  
final\_decrypted\_cards\_bob = [decrypt\_message(private\_key\_bob, card) for card in decrypted\_selected\_cards\_alice]  
print("Bob 最终解密的牌:", final\_decrypted\_cards\_bob)  
# Bob选择另外5张牌发送给Alice  
remaining\_indices\_bob = list(set(range(52)) - set(selected\_indices\_bob))  
additional\_selected\_indices\_bob = random.sample(remaining\_indices\_bob, 5)  
additional\_encrypted\_cards\_bob = [encrypted\_cards\_alice[i] for i in additional\_selected\_indices\_bob]  
# 5. Alice解密它们，变成她的一手牌  
final\_decrypted\_cards\_alice = [decrypt\_message(private\_key\_alice, card) for card in additional\_encrypted\_cards\_bob]  
print("Alice 最终解密的牌:", final\_decrypted\_cards\_alice)  
# 游戏结束，Alice和Bob出示他们的牌和密钥对  
print("Alice的密钥对:", private\_key\_alice)  
print("Bob的密钥对:", private\_key\_bob)  
# 验证双方牌是否正确  
print("Alice 选取的牌:", final\_decrypted\_cards\_alice)  
print("Bob 选取的牌:", final\_decrypted\_cards\_bob)  
# 验证牌的合法性和正确性  
# 验证 Bob 选取的5张牌通过 Alice 私钥解密后是否等于 Bob 最终解密的牌  
bob\_verification = all(decrypt\_message(private\_key\_alice, encrypted\_card) == card  
 for encrypted\_card, card in zip(selected\_encrypted\_cards\_bob, final\_decrypted\_cards\_bob))  
# 验证 Alice 用Bob的私钥解密 Bob 发送的牌是否在 Alice 最初的加密牌中  
alice\_verification = all(card in encrypted\_cards\_alice for card in (decrypt\_message(private\_key\_bob,selected\_cards\_bob)for selected\_cards\_bob in encrypted\_selected\_cards\_bob))  
if bob\_verification and alice\_verification:  
 print("双方验证成功，没有作弊行为。")  
else:  
 print("验证失败，存在作弊行为。")

运行结果：





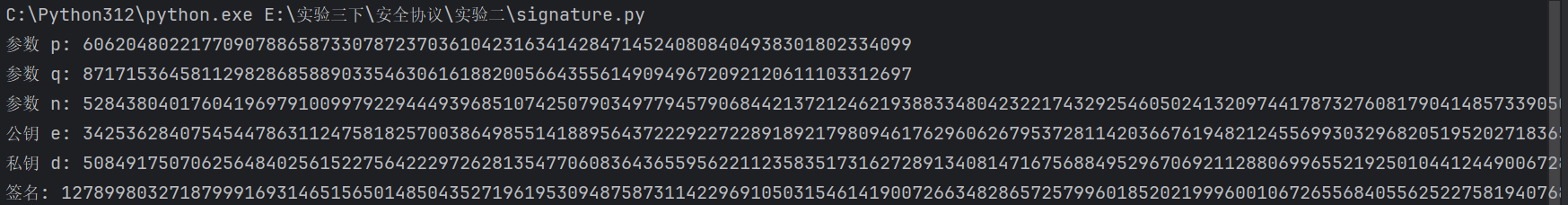
4. 编写RSA签名算法以及批签名方案(两个消息的批签名)。

1）实例化：令p,q均为512bits的参数，m1=4,随机选择一对公私钥(e，d)，计算其签名。

完整代码：

from RSA import generate\_p\_q\_n, generate\_rsa\_keys, encrypt\_message  
# 生成 p、q、n  
bits = 512  
p, q, n = generate\_p\_q\_n(bits)  
print("参数 p:", p)  
print("参数 q:", q)  
print("参数 n:", n)  
# 随机选择一对公私钥  
public\_key, private\_key = generate\_rsa\_keys(p, q, n)  
e, n = public\_key  
d, n = private\_key  
print("公钥 e:", e)  
print("私钥 d:", d)  
# 消息 m1=4  
m1 = 4  
# 计算签名,相当于拿私钥d对m1进行RSA加密  
signature = encrypt\_message(private\_key, m1)  
print("签名:", signature)

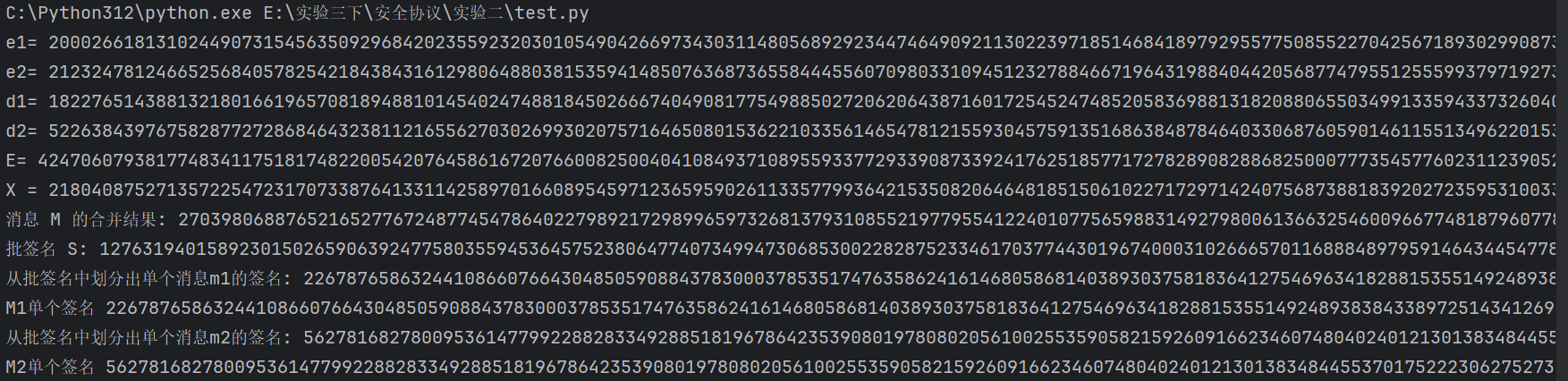
运行结果：



2）令m1=4,m2=7,随机选择两对公私钥(e1,d1),(e2,d2),计算其批签名，并从批签名中划分出单个消息m1,m2的签名。

from RSA import generate\_p\_q\_n, generate\_rsa\_keys,encrypt\_message  
# 扩展欧几里得  
def chinese\_remainder\_theorem(moduli, remainders):  
 if len(moduli) != len(remainders):  
 raise ValueError("模数和余数数量不一致")  
 n = 1  
 for modulus in moduli:  
 n \*= modulus  
 result = 0  
 for modulus, remainder in zip(moduli, remainders):  
 n\_i = n // modulus  
 inverse = pow(n\_i, -1, modulus) # 计算模数的乘法逆元  
 result += remainder \* n\_i \* inverse  
 return result % n  
# 生成512位的 p, q, n  
bits = 512  
p, q, n = generate\_p\_q\_n(bits)  
# 生成两对随机的公私钥对  
public\_key1, private\_key1 = generate\_rsa\_keys(p, q, n)  
e1, n = public\_key1  
d1, n = private\_key1  
public\_key2, private\_key2 = generate\_rsa\_keys(p, q, n)  
e2, n = public\_key2  
d2, n = private\_key2  
# 消息 M1 和 M2  
phi = (p - 1) \* (q - 1)  
M1 = 4  
M2 = 7  
# 计算私钥 E = e1 \* e2  
E = e1 \* e2  
print("e1=",e1)  
print("e2=",e2)  
print("d1=",d1)  
print("d2=",d2)  
print("E=",E)  
# 合并消息 M  
M = (pow(M1, (E\*pow(e1,-1,phi))%n, n) \* pow(M2, (E\*pow(e2,-1,phi))%n, n)) % n  
# 对消息 M 进行签名  
S = pow(M1,pow(e1,-1,phi),n)\*pow(M2,pow(e2,-1,phi),n)  
# 求 X 满足 X = 0 mod e1 和 X = 1 mod e2  
moduli = [e1, e2] # 模数 e1 和 e2  
remainders = [0, 1] # 对应的余数  
X = chinese\_remainder\_theorem(moduli, remainders)  
print("X =", X)  
# 划分出每一条消息的签名  
signature\_M2 = (pow(S, X, n) \* pow(pow(M1, X\*pow(e1,-1,phi), n) \* pow(M2,(X-1)\*pow(e2,-1,phi), n), -1,n)) %n  
signature\_M1 = (S\*pow(pow(M2,(pow(e2,-1,phi)),n),-1,n)) % n  
print("消息 M 的合并结果:", M)  
print("批签名 S:", S)  
print("从批签名中划分出单个消息m1的签名:", signature\_M1)  
print("M1单个签名",pow(M1,pow(e1,-1,phi),n))  
print("从批签名中划分出单个消息m2的签名:", signature\_M2)  
print("M2单个签名",pow(M2,pow(e2,-1,phi),n))

运行结果：



1. **实验总结**

在本次实验中，我深入实践了RSA密码算法的各个方面，具体包括RSA加密解密、签名与验证、以及批签名方案。通过实现RSA算法，我理解了密钥生成、加密和解密的全过程。使用不同位数的参数（如512位和1024位）生成大素数，增强了对算法安全性的理解。之后验证了RSA算法在相同模数下满足交换律。这一验证加深了我们对RSA算法数学基础的理解，对利用RSA密码算法实现两方智力扑克游戏原理也有更深的理解，通过实现基于RSA加密的两方智力扑克游戏，我理解了公钥加密在信息安全传输中的应用。感受到密码学、安全协议的乐趣。通过编写RSA签名算法，我们掌握了数字签名的原理及其在信息认证中的作用。实现两个消息的批签名方案，展示了批签名在提高签名效率方面的应用，并学习了如何从批签名中提取单个消息的签名，对于提取单个签名过程中，是对还是n求逆有了更深的理解，以及除法求逆操作，对于指数需要对求逆，对于除数要对n求逆。总体而言，本次实验不仅巩固了我们对RSA密码算法的理解，还使我们能够应用RSA算法解决实际问题，强化了我对安全协议相关知识的理解与掌握。