****

密 码 学 与 应 用

实验报告(现代密码及协议)

**学生姓名 杨凯楠**

**学 号** 8208201004

**专业班级 信息安全2002班**

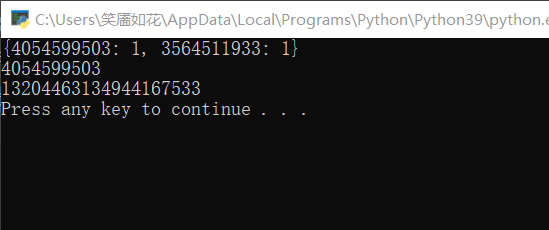
**指导教师** 段桂华

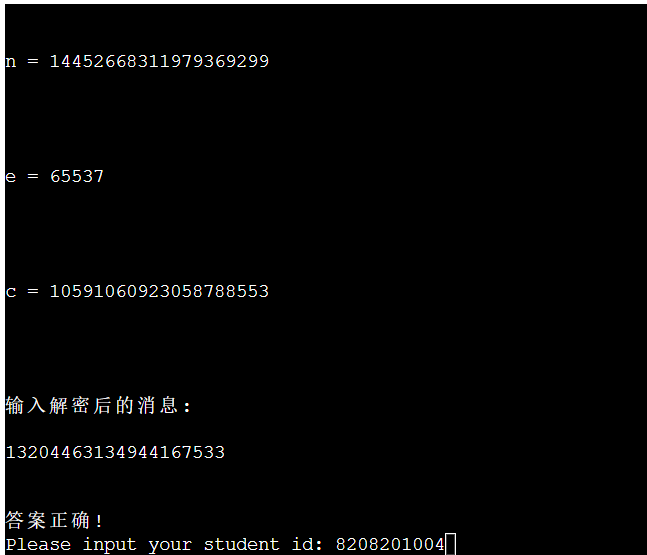
**学 院** 计算机学院

**完成时间** 2021年12月5日

1. **因式分解攻击**

**分析：**这次实验的因式分解攻击的质数并不是太大，用python标准库中的libnum.factorize(n)即可将其在很短的时间内分解，然后利用RSA算法原理就能很轻松将其破解。如果公钥n更大一点，则可能需要动用yafu等工具。





**代码：**

1. **import** gmpy2
2. **import** libnum
4. n = 14452668311979369299
6. e = 65537
8. c = 10591060923058788553
9. **print**(libnum.factorize(n))
10. q=int(input())
12. p=n//q
13. phi=(p-1)\*(q-1)
14. d=gmpy2.invert(e,phi)
15. m=pow(c,d,n)
16. **print**(int(m))
17. **选择密文攻击**

**分析：**

攻击思路：

令m2=2

c2=2^e

计算X = (C × 2^e)mod n

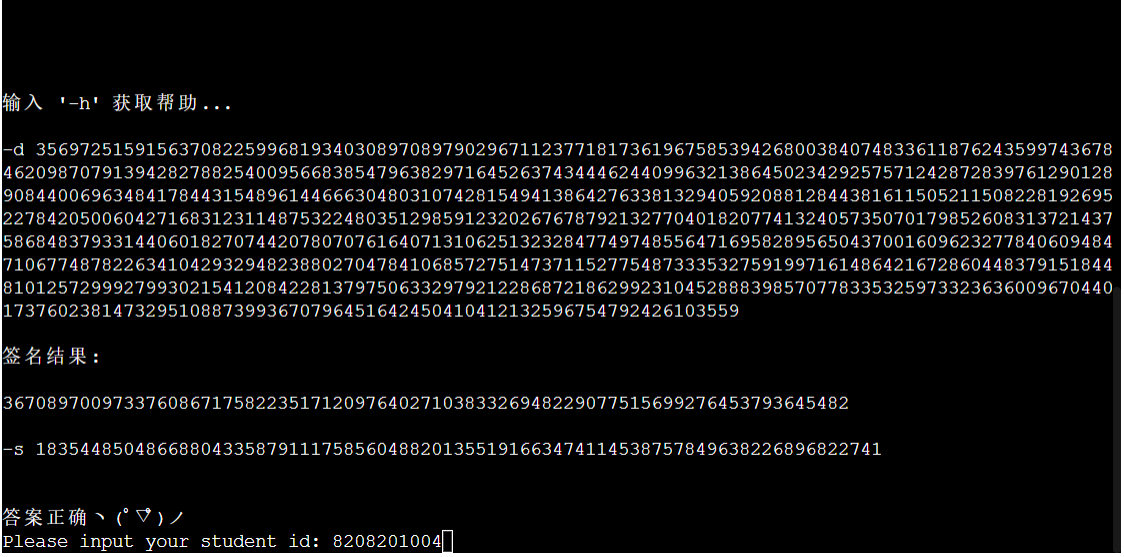
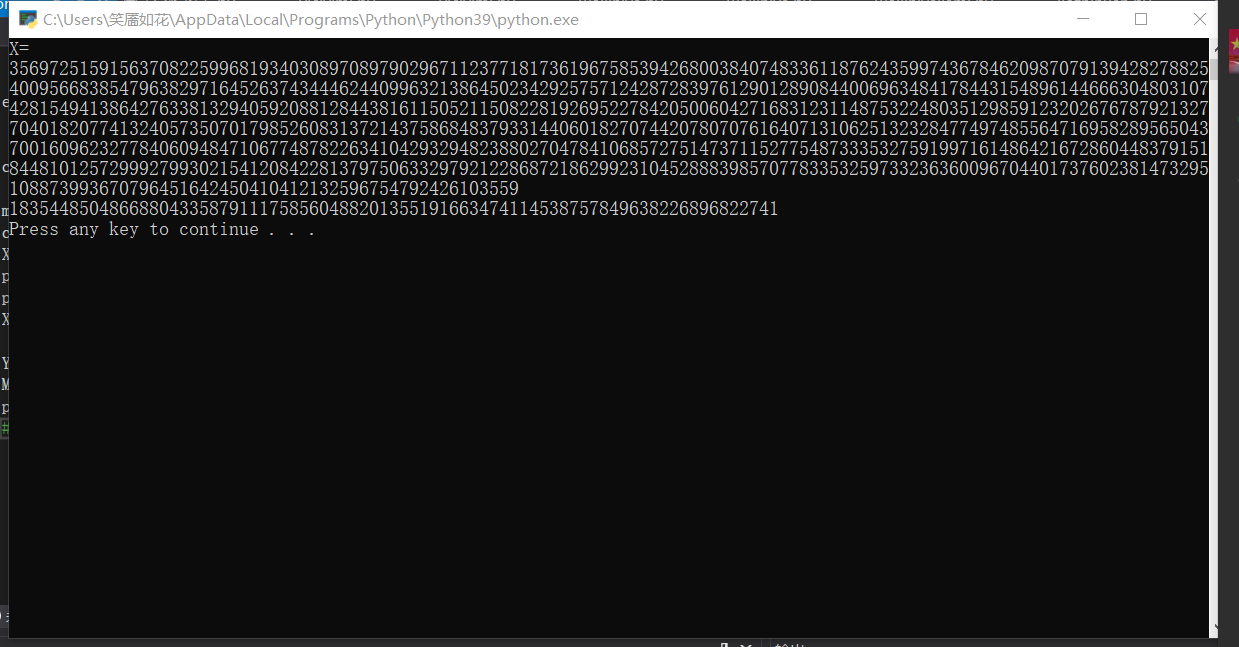
将X作为选择明文提交，并收到Y = X^d mod n。

因此：X = (C mod n) × (2^e mod n) = (M^e mod n) × (2 ^e mod n) = (2M)^e mod n

因此：Y = (2M) mod n

然后计算M=(Y\*gmpy2.invert(2,n))%n，得到M

**实现：**



**代码：**由于代码中数字位数占比甚至比实际代码量还高，故而不贴全部代码了，只贴去掉数字的部分

1. **import** libnum
2. **import** gmpy2
3. n =
5. e = 65537
7. c =
9. m2=2
10. c2=
11. X=(c\*(2\*\*e))%n
12. **print**("X=")
13. **print**(X)
14. X=
16. Y=36708970097337608671758223517120976402710383326948229077515699276453793645482
17. M=(Y\*gmpy2.invert(2,n))%n
18. **print**(M)
19. #M=18354485048668804335879111758560488201355191663474114538757849638226896822741
20. **共模攻击**

**分析：**生成秘钥的过程中使用了相同的模数n，此时用不同的秘钥e加密同一信息m即：

c1 = m^e1 % n

c2 = m^e2 % n

若两个秘钥e互素根据扩展的欧几里得算法则存在s1，s2有：

e1 \* s1 + e2 \* s2 = gcd(e1, e2) = 1

经过数学计算：

(c1^s1 \* c2^s2) %n

= (m^e1 % n)^s1 \* (m^e2 %n)^s2 % n

= m^(e1 \* s1 + e2 \* s2) % n

= m % n

= m

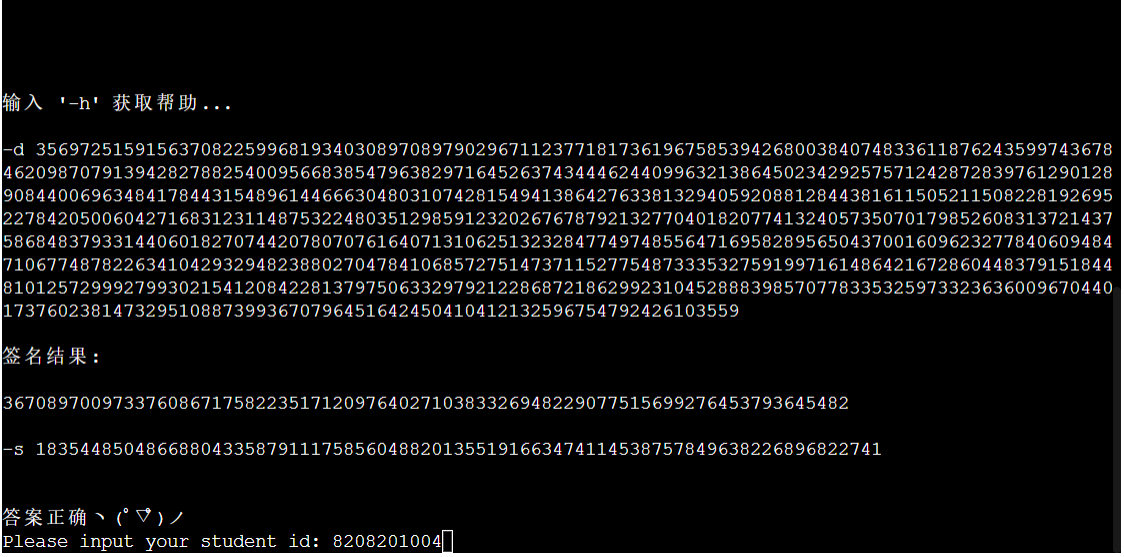
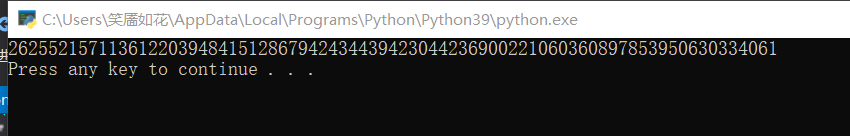
也就是在不知道私钥的情况下，得到了明文m

在此攻击过程中，若s2或s1有一个为负数，则需要将其变换为

s1=-s1

c1=gmpy2.invert(c1,n)

**实现：**



**代码：**

1. n =
2. e1 = 42131
3. c1 =
4. e2 = 37561
5. c2 =
6. **import** libnum
7. **import** gmpy2
8. s=gmpy2.gcdext(e1,e2)#gcdext(a, b) 返回一个 3 元素元组 ( g , s , t ) 使得g == gcd( a , b ) 和g == a \* s + b \* t
9. s1=s[1]
10. s2=s[2]
11. **if** s1<0:
12. s1=-s1
13. c1=gmpy2.invert(c1,n)
14. **if** s2<0:
15. s2=-s2
16. c2=gmpy2.invert(c2,n)
17. m=int(pow(c1,s1,n)\*pow(c2,s2,n)%n)
18. **print**(m)
20. m=58753995245939403348632161371730171834688257191720851947508380454128181010988
21. **针对RSA的低加密指数广播攻击**

**分析**：本道题解题方法在题干里已经说过了，简明的说来，就是

当e=3时

ci =m^3 mod Ni

可以设c且c满足0<c<N\_1\*N\_2\*N\_3

则c=c1 mod N1 ,c=c2 mod N2 ,c=c3 mod N3

c=m^3 mod N1\*N2\*N3

由于m<Ni对于所有的i都成立，所以有 m^3<N\_1\*N\_2\*N\_3

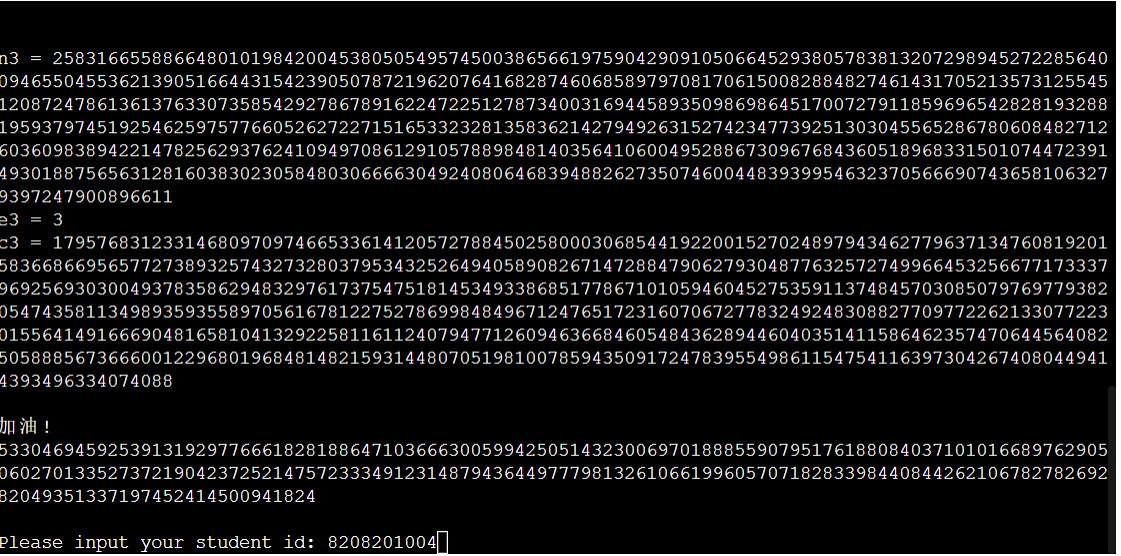
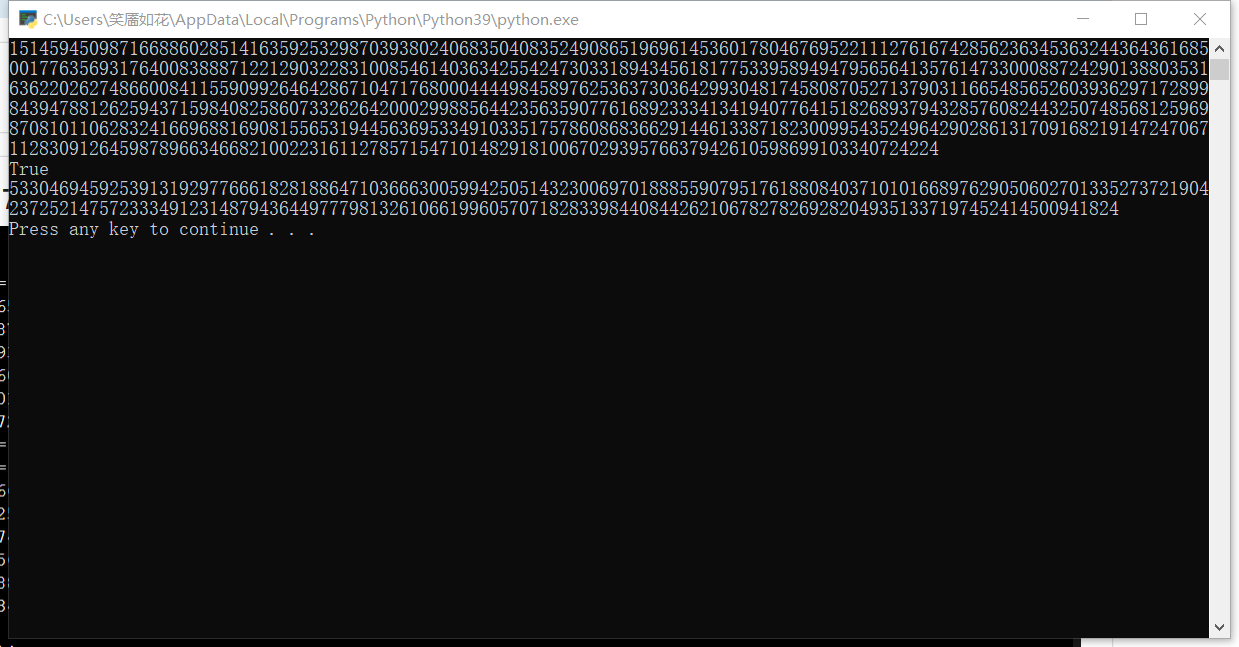
从而，就可以得c=m^3，对c做立方根运算即可得到消息m。

本题的一大难点是对一个大数进行开立方根的运算普通的python开根方法c\*\*（1/3）和pow（c,(1/3)）均会带来精度损失得不到正确的结果。因此我查阅了各种方法，终于发现了gmpy2库中的函数iroot（m，a）表示对m开a次方，返回只有两个，第一个为结果，第二个为bool类型，True表示能被开方成为整数。

**代码**：

1. **from** gmpy2 **import**\*
3. n1 =
4. e1 = 3
5. c1 =
7. n2 =
8. e2 = 3
9. c2 =
11. n3 =
12. e3 = 3
13. c3 =
14. N=n1\*n2\*n3
15. N1=N//n1
16. N2=N//n2
17. N3=N//n3
18. niN1=invert(N1,n1)
19. niN2=invert(N2,n2)
20. niN3=invert(N3,n3)
21. m3=(c1\*N1\*niN1+c2\*N2\*niN2+c3\*N3\*niN3)%N
22. **print**(int(m3))
23. status = iroot(m3, 3)[1] #status为1表示改数能被开立方成整数
24. **print**(status)
25. m =  iroot(m3, 3)[0] #开立方后的结果
26. **print**(int(m))

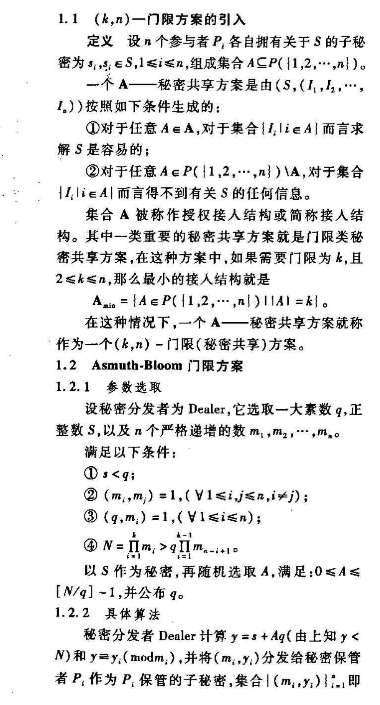
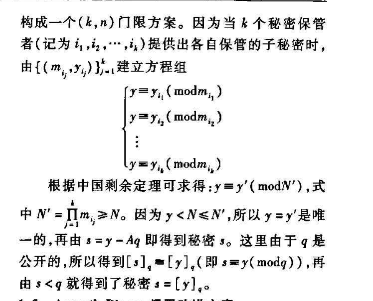
**结果：**



1. **bloom门限方案协议**

门限方案作为一个大题目，实验要求比前面的实验高不少，但是由于这学期的特殊性，本专业学生没有时间写一个完整的，具有多种方案的，有GUI的门限方案协议，故而只将其中最核心的，最根本的算法实现后提交。

**分析**：



**代码实现**：

1. **import** gmpy2
3. p = 752979826723504289885517849949884860012101561465169741735331775988665620861721215325933699211966288616590491399174416214424633096224996347432419645885614679664443557666233192253943645677162226586707259783212070571582625166138445389975802487442785676999894664276779822080400611576514387245013421805401

6. x1 = 54221978911948699451080919207804870371197109074259509922022488136996718858502924130029942605665157731909071026317540997654476419941048421259319732649894355086917543753319521800522672953415423954210234393574733138326549142425963278835183213270005039033665220995012047634212734961880592507135746380463260174818
7. m1 = 152907914649023715406015997278717584439726711589576713324827955419928922242042827610398882782946082470043773891850346728835994511479302856796032357673475374787417825643956495833920216689910590476761395592809983986398786226475806755960495406121994243465081767531670587775068735762154790201776599042843130603693

10. x2 = 154193930153055172150768078993280823005404869057828334502211120769077651390498877324992904514987138296690612604505278511028216409355770777373197454000031201252948371386608665167801724866025354511527074890367382052642960154486043744423603153689669308674010179416173073286129376243765931435178227789514359022981
11. m2 = 311083617279867968458837797409994220728749200222056947534226031475888988609136933735179101424289832249251205537137950510713332097751165362045555277314499319598384509174982178476642581906202374587748418097509684660939361644989304514358502280861624749482566278363668877073819192523956174250049481732836412437403

14. x3 = 86003714435720515858424634002533405759388810979261582206721319192448905829067922914703940397619725021361270058996885018199199893726930531664091457274465578809438706111803515820907024710466276752768497363920069344321569434427469252646730516648065820086227872888985201335652137316440212569709116659496043385570
15. m3 = 493692399665099468632126395810415335223491395165149077481682432603103541020703876768432706092958212173041513513386753405213957383466435430777301569107867146546530762481511273856570895393626416274884577280396794489719686058705033634398497655519674529177364714236662113403368054318288511717707726922373804210937

18. Mlist=[m1,m2,m3]
19. Xlist=[x1,x2,x3]
20. M=1
21. **for** i **in** Mlist:
22. M\*=i
24. A=[]
25. W=[]
26. T=[]
28. **for** i **in** range(3):
29. A.append(Xlist[i]%Mlist[i])
30. W.append(M//Mlist[i])
31. T.append(gmpy2.invert(W[i],Mlist[i]))
32. x=0
33. **for** i **in** range(3):
34. x+=A[i]\*T[i]\*W[i]
35. x=x%M
36. s=x%p
37. **print**(s)
38. **print**(bin(s))
39. #print(len(bin(s))-2)

**实现**：

