CENTRAL SOUTH UNIVERSITY

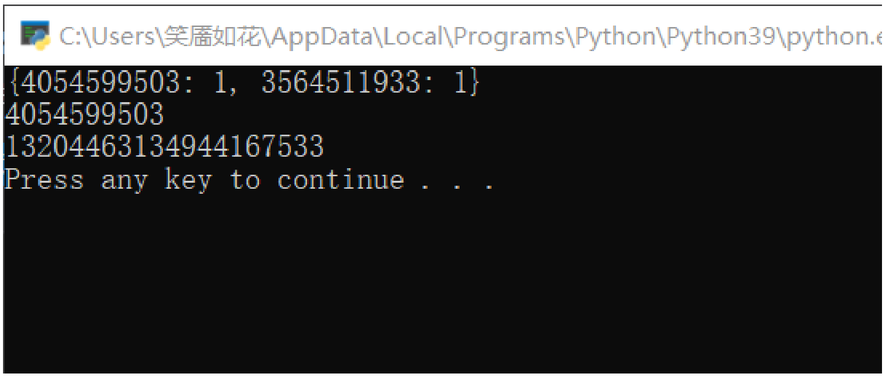
密 码 学 与 应 用

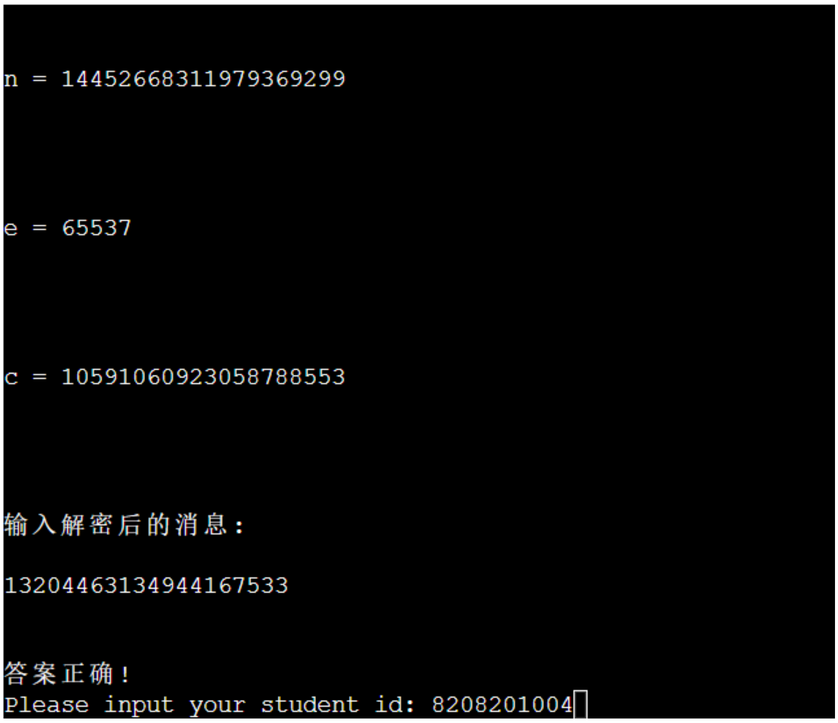
实验报告(现代密码及协议)

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

一、 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：

```python
import gmpy2
import libnum

n = 14452668311979369299

e = 65537

c = 10591060923058788553
```

```
print(libnum.factorize(n))
q=int(input())

p=n//q
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(int(m))
```

二、 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

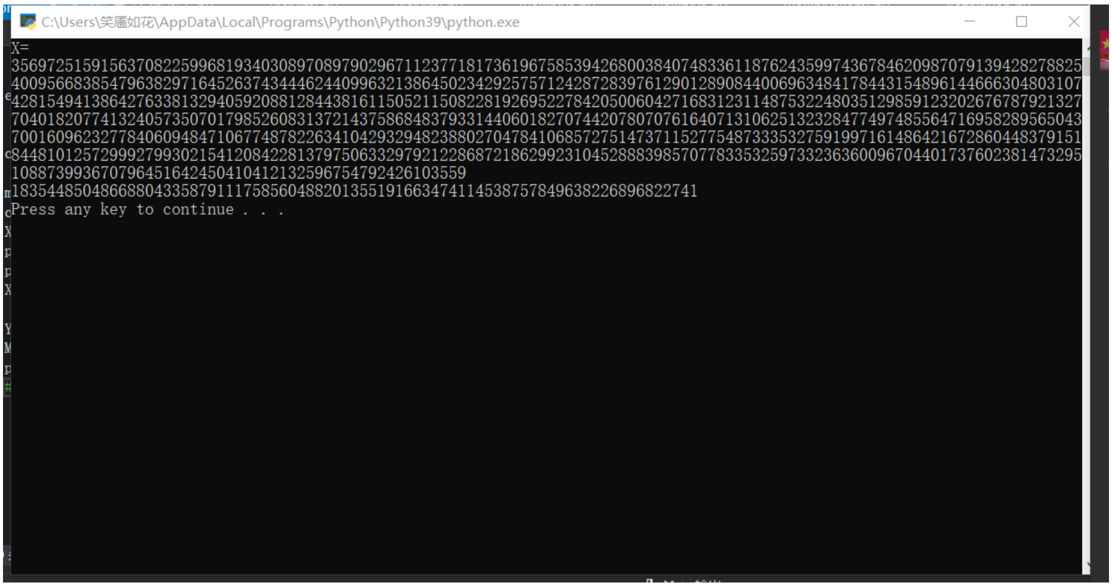
将X作为选择明文提交，并收到Y = X^d mod n。

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：



```
输入 '-h' 获取帮助...

-d 356972515915637082259968193403089708979029671123771817361967585394268003840748336118762435997436784620987079139428278825400956683854796382971645263743444624409963213864502342925757124287283976129012890844006963484178443154896144666304803107428154941386427633813294059208812844381611505211508228192695227842050060427168312311487532248035129859123202676787921327704018207741324057350701798526083137214375868483793314406018270744207807076164071310625132328477497485564716958289565043700160962327784060948471067748782263410429329482388027047841068572751473711527754873335327591997161486421672860448379151844810125729992799302154120842281379750633297921228687218629923104528883985707783353259733236360096704401737602381473295108873993670796451642450410412132596754792426103559

签名结果:

36708970097337608671758223517120976402710383326948229077515699276453793645482

-s 18354485048668804335879111758560488201355191663474114538757849638226896822741


答案正确ヽ(°▽°)ノ
Please input your student id: 8208201004
```

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

```python
import libnum
import gmpy2
n =

e = 65537

c =

m2=2
c2=

X=(c*(2**e))%n
print("X=")
print(X)
X=

Y=36708970097337608671758223517120976402710383326948229077515699276453793645482
M=(Y*gmpy2.invert(2,n))%n
print(M)
#M=18354485048668804335879111758560488201355191663474114538757849638226896822741
```

三、 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

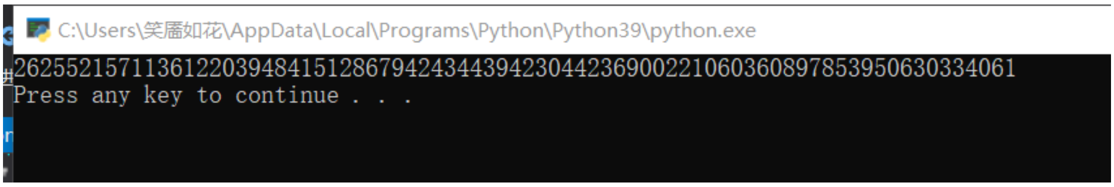
经过数学计算：

  (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：



```
输入 '-h' 获取帮助...

-d 3569725159156370822599681934030897089790296711237718173619675853942680038407483361187624359974367846209870791394282788254009566838547963829716452637434446244099632138645023429257571242872839761290128908440069634841784431548961446663048031074281549413864276338132940592088128443816115052115082281926952278420500604271683123114875322480351298591232026767879213277040182077413240573507017985260831372143758684837933144060182707442078070761640713106251323284774974855647169582895650437001609623277840609484710677487822634104293294823880270478410685727514737115277548733353275919971614864216728604483791518448101257299927993021541208422813797506332979212286872186299231045288839857077833532597332363600967044017376023814732951088739936707964516424504104121325967547924261035559

签名结果：

36708970097337608671758223517120976402710383326948229077515699276453793645482

-s 18354485048668804335879111758560488201355191663474114538757849638226896822741


答案正确ヽ(°▽)ノ
Please input your student id: 8208201004
```

代码：

```python
n = 
e1 = 42131
c1 = 
e2 = 37561
c2 = 
import libnum
import gmpy2
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
s1=s[1]
s2=s[2]
if s1<0:
    s1=-s1
    c1=gmpy2.invert(c1,n)
if s2<0:
    s2=-s2
    c2=gmpy2.invert(c2,n)
m=int(pow(c1,s1,n)*pow(c2,s2,n)%n)
print(m)

m=58753995245939403348632161371730171834688257191720851947508380454128181010988
```

四、 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

$c_i = m^3 \bmod N_i$

可以设 c 且 c 满足 $0<c<N_1*N_2*N_3$

则 $c=c_1 \bmod N_1$ ，$c=c_2 \bmod N_2$ ，$c=c_3 \bmod N_3$

$c=m^3 \bmod N_1*N_2*N_3$

由于 $m<N_i$ 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。

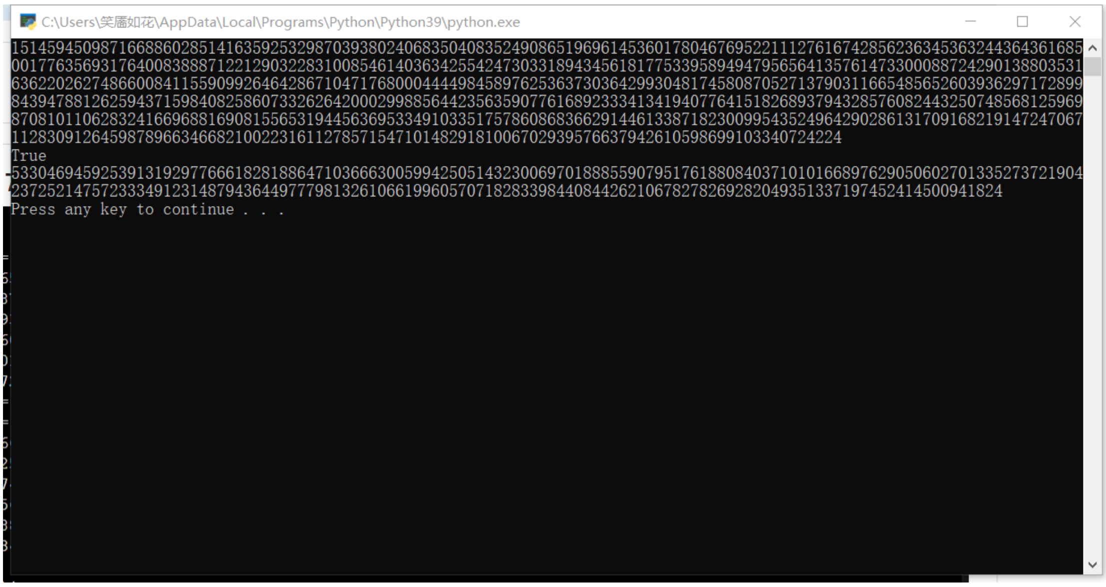
本题的一大难点是对一个大数进行开立方根的运算普通的 python 开根方法 c**（1/3）和 pow（c,(1/3)）均会带来精度损失得不到正确的结果。因此我查阅了各种方法，终于发现了 gmpy2 库中的函数 iroot（m，a）表示对 m 开 a 次方，返回只有两个，第一个为结果，第二个为 bool 类型，True 表示能被开方成为整数。

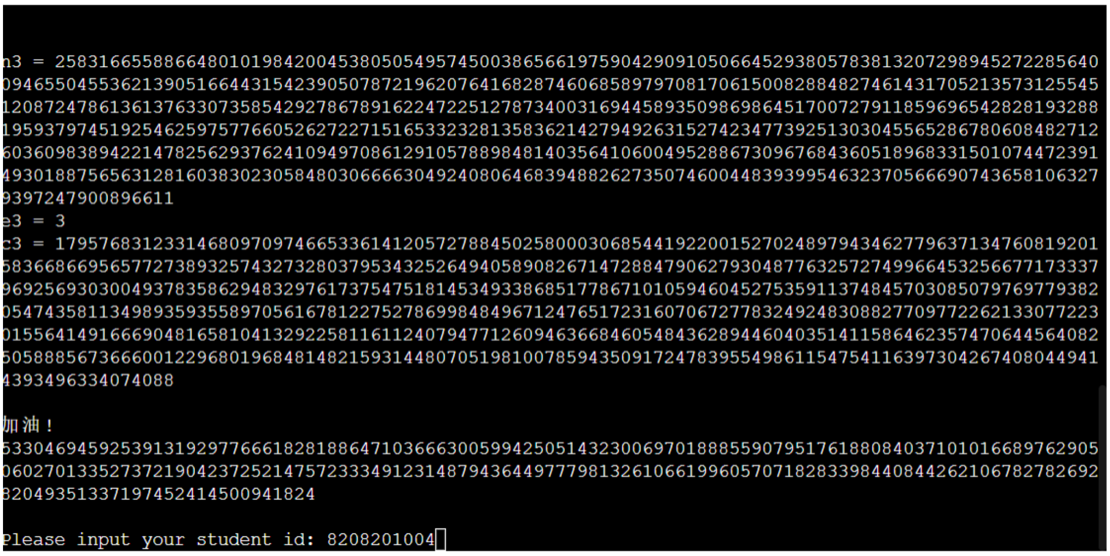
代码：

```python
from gmpy2 import*

n1 =
e1 = 3
c1 =

n2 =
e2 = 3
c2 =

n3 =
e3 = 3
c3 =
N=n1*n2*n3
N1=N//n1
N2=N//n2
N3=N//n3
niN1=invert(N1,n1)
niN2=invert(N2,n2)
niN3=invert(N3,n3)
m3=(c1*N1*niN1+c2*N2*niN2+c3*N3*niN3)%N
print(int(m3))
status = iroot(m3, 3)[1] #status 为 1 表示改数能被开立方成整数
print(status)
```

```python
m =  iroot(m3, 3)[0] #开立方后的结果
print(int(m))
```

结果：





五、 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：

1.1 (k,n)—门限方案的引入

定义 设 n 个参与者 P_i 各自拥有关于 S 的子秘密为 $s_i, s_i \in S, 1 \leqslant i \leqslant n$，组成集合 $A \subseteq P(\{1,2,\cdots,n\})$。

一个 $\mathbf{A}$——秘密共享方案是由 $(S,(I_1,I_2,\cdots,I_n))$ 按照如下条件生成的：

①对于任意 $A \in \mathbf{A}$，对于集合 $\{I_i \mid i \in A\}$ 而言求解 S 是容易的；

②对于任意 $A \in P(\{1,2,\cdots,n\}) \setminus \mathbf{A}$，对于集合 $\{I_i \mid i \in A\}$ 而言得不到有关 S 的任何信息。

集合 $\mathbf{A}$ 被称作授权接入结构或简称接入结构。其中一类重要的秘密共享方案就是门限类秘密共享方案，在这种方案中，如果需要门限为 k，且 $2 \leqslant k \leqslant n$，那么最小的接入结构就是

$$\mathbf{A}_{\min} = \{A \in P(\{1,2,\cdots,n\}) \mid |A| = k\}。$$

在这种情况下，一个 $\mathbf{A}$——秘密共享方案就称作为一个 (k,n) - 门限（秘密共享）方案。

1.2 Asmuth-Bloom 门限方案

1.2.1 参数选取

设秘密分发者为 Dealer，它选取一大素数 q，正整数 S，以及 n 个严格递增的数 $m_1, m_2, \cdots, m_n$。

满足以下条件：

① $s < q$；

② $(m_i, m_j) = 1, (\forall 1 \leqslant i,j \leqslant n, i \neq j)$；

③ $(q, m_i) = 1, (\forall 1 \leqslant i \leqslant n)$；

④ $N = \prod_{i=1}^{k} m_i > q \prod_{i=1}^{k-1} m_{n-i+1}$。

以 S 作为秘密，再随机选取 A，满足：$0 \leqslant A \leqslant [N/q] - 1$，并公布 q。

1.2.2 具体算法

秘密分发者 Dealer 计算 $y = s + Aq$（由上知 $y < N$）和 $y \equiv y_i (\bmod m_i)$，并将 (m_i, y_i) 分发给秘密保管者 P_i 作为 P_i 保管的子秘密，集合 $\{(m_i, y_i)\}_{i=1}^{n}$ 即构成一个 (k,n) 门限方案。因为当 k 个秘密保管者（记为 $i_1, i_2, \cdots, i_k$）提供出各自保管的子秘密时，由 $\{(m_{i_j}, y_{i_j})\}_{j=1}^{k}$ 建立方程组

$$\begin{cases} y \equiv y_{i_1} (\bmod m_{i_1}) \\ y \equiv y_{i_2} (\bmod m_{i_2}) \\ \vdots \\ y \equiv y_{i_k} (\bmod m_{i_k}) \end{cases}$$

根据中国剩余定理可求得：$y \equiv y' (\bmod N')$，式中 $N' = \prod_{j=1}^{k} m_{i_j} \geqslant N$。因为 $y < N \leqslant N'$，所以 $y = y'$ 是唯一的，再由 $s = y - Aq$ 即得到秘密 s。这里由于 q 是公开的，所以得到 $[s]_q = [y]_q$（即 $s \equiv y (\bmod q)$），再由 $s < q$ 就得到了秘密 $s = [y]_q$。

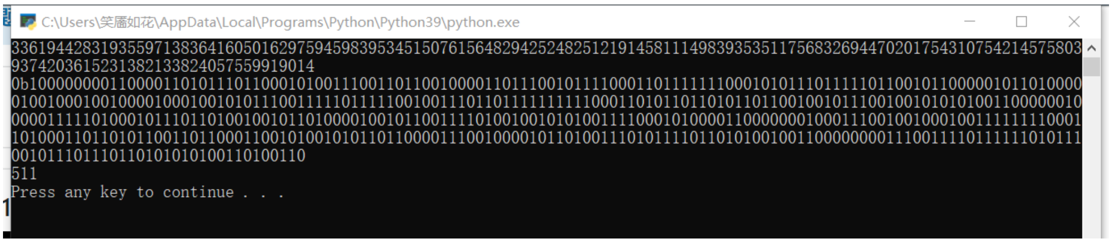
代码实现：

```python
import gmpy2

p = 7529798267235042898855178499498848600121015614651697417353317759886656208617212153259336992119662886165904913991744162144246330962249963474324196458856146796644435576662331922539436456771622265867072597 8
```

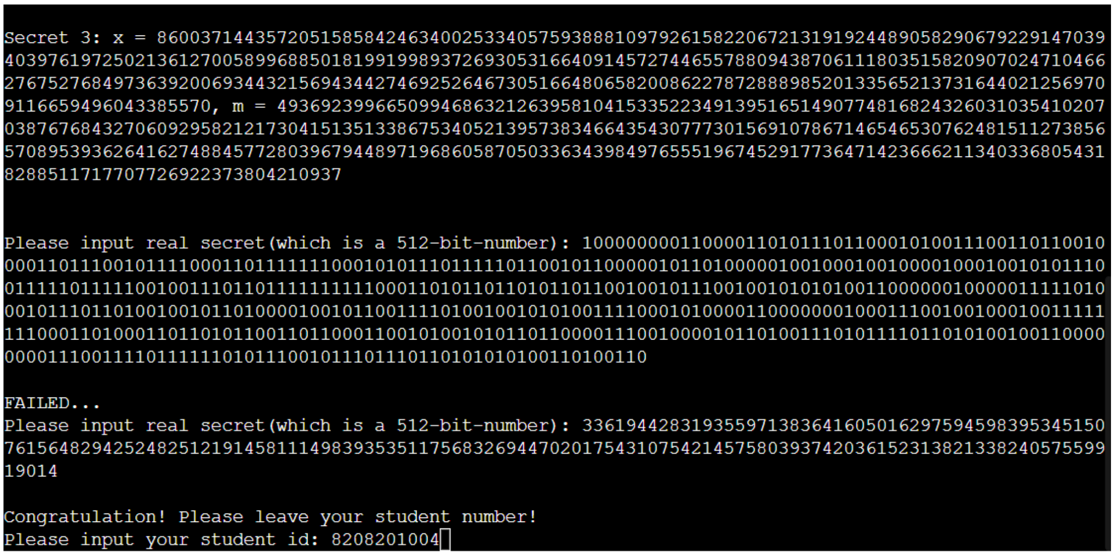
```
3212070571582625166138445389975802487442785676999894664276779822080400611576514387245013421805401


x1 = 54221978911948699451080919207804870371197109074259509922022488136996718858502924130029942605665157731909071026317540997654476419941048421259319732649894355086917543753319521800522672953415423954210234393574733138326549142425963278835183213270005039033665220995012047634212734961880592507135746380463260174818
m1 = 152907914649023715406015997278717584439726711589576713324827955419928922242042827610398882782946082470043773891850346728835994511479302856796032357673475374787417825643956495833920216689910590476761395592809983986398786226475806755960495406121994243465081767531670587775068735762154790201776599042843130603693


x2 = 154193930153055172150768078993280823005404869057828334502211120769077651390498877324992904514987138296690612604505278511028216409355770777373197454000031201252948371386608665167801724866025354511527074890367382052642960154486043744423603153689669308674010179416173073286129376243765931435178227789514359022981
m2 = 311083617279867968458837797409994220728749200222056947534226031475888988609136933735179101424289832249251205537137950510713332097751165362045555277314499319598384509174982178476642581906202374587748418097509684660939361644989304514358502280861624749482566278363668877073819192523956174250049481732836412437403


x3 = 86003714435720515858424634002533405759388810979261582206721319192448905829067922914703940397619725021361270058996885018199199893726930531664091457274465578809438706111803515820907024710466276752768497363920069344321569434427469252646730516648065820086227872888985201335652137316440212569709116659496043385570
m3 = 493692399665099468632126395810415335223491395165149077481682432603103541020703876768432706092958212173041513513386753405213957383466435430777301569107867146546530762481511273856570895393626416274884577280396794489719686058705033634398497655519674529177364714236662113403368054318288511717707726922373804210937


Mlist=[m1,m2,m3]
Xlist=[x1,x2,x3]
M=1
for i in Mlist:
```

```python
    M*=i

A=[]
W=[]
T=[]

for i in range(3):
    A.append(Xlist[i]%Mlist[i])
    W.append(M//Mlist[i])
    T.append(gmpy2.invert(W[i],Mlist[i]))
x=0
for i in range(3):
    x+=A[i]*T[i]*W[i]
x=x%M
s=x%p
print(s)
print(bin(s))
#print(len(bin(s))-2)
```

实现：



```
C:\Users\笑靥如花\AppData\Local\Programs\Python\Python39\python.exe
336194428319355971383641605016297594598395345150761564829425248251219145811149839353511756832694470201754310754214575803
9374203615231382133824057559919014
0b10000000011000011010111011000101001110011011001000011011100101111000110111111110001010111011111011001011000000101101000001001000100100001000100101011100111110111110010011101101111111111000110101101101011011001001011100100101010100110000001000001111101000101110110100100101101000010010110011110100100101010011110001010000110000000010001110010010001001111111000110100011011010110011011000110010100101011011000011100100001011010011101011110110101001001100000000111001111011111101011100101110111011010101010100110100110
511
Press any key to continue . . .
```



```
Secret 3: x = 86003714435720515858424634002533405759388810979261582206721319192448905829067922914703940397619725021361270058996885018199199893726930531664091457274465578809438706111803515820907024710466276752768497363920069344321569434427469252646730516648065820086227872888985201335652137316440212569709116659496043385570, m = 493692399665099468632126395810415335223491395165149077481682432603103541020703876768432706092958212173041513513386753405213957383466435430777301569107867146546530762481511273856570895393626416274884577280396794489719686058705033634398497655519674529177364714236662113403368054318288511717707726922373804210937


Please input real secret(which is a 512-bit-number): 10000000011000011010111011000101001110011011001000011011100101111000110111111110001010111011111011001011000000101101000001001000100100001000100101011100111110111110010011101101111111111000110101101101011011001001011100100101010100110000001000001111101000101110110100100101101000010010110011110100100101010011110001010000110000000010001110010010001001111111110001101000110110101100110110001100101001010110110000111001000010110100111010111101101010010011000000000111001111011111101011100101110111011010101010100110100110

FAILED...
Please input real secret(which is a 512-bit-number): 3361944283193559713836416050162975945983953451507615648294252482512191458111498393535117568326944702017543107542145758039374203615231382133824057559919014

Congratulation! Please leave your student number!
Please input your student id: 8208201004
```