ctf小学生_rsa writeup

最近在b站学习发现了一名大佬风二西，他讲解的RSA和流量题都非常基础，特别适合小白零基础入门。做RSA题目，不可避免要写脚本，风大把RSA这一块的脚本 用Python3做了一遍，专门出了一期合辑python3的RSA脚本，可以说是非常用心，就这还怕大家编程不过关，改代码都不会，又写了一款图形化界面的RSA解题工具 工具地址-见视频评论区。

CTF-小学生这个平台，也是风大搭建的， 大部分题目都是自己出的，题目难度非常友好。稍微难点的题目在风大的B站上都有讲解。

本文主要收录一下自己的解题过程。这些题目用风大的工具来解如同砍瓜切菜般流畅。这里为了积累一下代码，主要记录一下不用工具的其他方法。

风二西_RSA1

题目：e=1 (风二西原创题)

```python
import gmpy2
import libnum
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m=libnum.s2n(flag)

p=libnum.generate_prime(512)
q=libnum.generate_prime(512)
n=p*q
e=1
c=pow(m,e,n)
print("n=",n)
print("c=",c)
print("e=",e)
n= 90981495617756764768563887456121485632852821294992273406540426689270512099459758619024797604428497371505761088482731691740604544236399875476209161759486542642982460755167111605224305056733441714037861907483938283623389941146434398522162360139652831844867722242479359748886192684447486989079947504196734648421
c= 56006392793403067781861231386277942050474101531963376999457063633948500765747587998496106575433840765

```

RSA的原理中 $m^e \equiv c \pmod{n}$， 当 e 为1 时， $m^e \equiv m^1 \equiv m \equiv c \pmod{n}$ 由于m是小于n的，题目中给出的密文c就是m。
可以直接对c解码得到m

```python
import libnum
c= 56006392793403067781861231386277942050474101531963376999457063633948500765747587998496106575433840765
print(libnum.n2s(c))
# flag{046b9e03-474f-4ac0-9372-25bfc545dc08}
```

风二西_RSA2

题目：还是e=1 风二西原创题

```python
import gmpy2
import libnum
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m=libnum.s2n(flag)

p1=libnum.generate_prime(64)
q1=libnum.generate_prime(64)
p2=libnum.generate_prime(64)
q2=libnum.generate_prime(64)
p3=libnum.generate_prime(64)
q3=libnum.generate_prime(64)
e=1
c1=pow(m,e,p1*q1)
c2=pow(m,e,p2*q2)
c3=pow(m,e,p3*q3)
print("n1=",p1*q1)
print("c1=",c1)
print("n2=",p2*q2)
print("c2=",c2)
print("n3=",p3*q3)
print("c3=",c3)
n1= 172774622114813683746188230007837413819
c1= 170260248491697016437095929037490480036
n2= 160333927436069409658483084503168246581
c2= 45134242975344810542214361639231372051
n3= 170109598387116572557100744899522621873
c3= 47903985600747367026642413789127948969
```

这个题目中，加密指数e仍是等于1，但跟第一题不同的是，密文m的长度是大于n的，所以给出了多组的数据，这里需要用到中国剩余定理。解题脚本如下：

```python
n1= 172774622114813683746188230007837413819
c1= 170260248491697016437095929037490480036
n2= 160333927436069409658483084503168246581
c2= 45134242975344810542214361639231372051
n3= 170109598387116572557100744899522621873
c3= 47903985600747367026642413789127948969

def GCRT(mi, ai):
    # mi,ai分别表示模数和取模后的值,都为列表结构
    assert (isinstance(mi, list) and isinstance(ai, list))
    curm, cura = mi[0], ai[0]
    for (m, a) in zip(mi[1:], ai[1:]):
        d = gmpy2.gcd(curm, m)
        c = a - cura
        assert (c % d == 0) #不成立则不存在解
        K = c // d * gmpy2.invert(curm // d, m // d)
        cura += curm * K
        curm = curm * m // d
```

```python
        cura %= curm
    return (cura % curm, curm) #(解,最小公倍数)

C,N = GCRT([n1,n2,n3],[c1,c2,c3])
print(libnum.n2s(C))
# flag{cba55428-f26b-4065-9f34-81c5c8e2c637}
```

风二西_RSA3

题目：是同模？还是共模？ 风二西原创题

```python
import libnum
import gmpy2
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)

m=libnum.s2n(flag)
p=libnum.generate_prime(1024)
q=libnum.generate_prime(1024)
n1=p*q
n2=p*q
e1=2333
e2=23333
m=libnum.s2n(flag)
c1=pow(m,e1,n1)
c2=pow(m,e2,n2)
print("n1=",n1)
print("n2=",n2)
print("e1=",e1)
print("e2=",e2)
print("c1=",c1)
print("c2=",c2)
n1= 2914364542125004196461013151979631620937439720415546997643628297027022309322727011693614877504381563454278605395775464858854791668585594323374735508795025542008452920827295972679894477152981228021159524663232416431841456892162090322879231242294904925112467510535709600151190018238498213660846900447587735044376789897398958317312803043494088605279279781654078735861026379810951747640485788485373794685159902069522887437415446455442405264147381862861931554258095867832462525150868775528162072024799723923276854828384110339149801623963080648198067147537246333033069055966818243104668438970759683086807208275573580830072
n2= 2914364542125004196461013151979631620937439720415546997643628297027022309322727011693614877504381563454278605395775464858854791668585594323374735508795025542008452920827295972679894477152981228021159524663232416431841456892162090322879231242294904925112467510535709600151190018238498213660846900447587735044376789897398958317312803043494088605279279781654078735861026379810951747640485788485373794685159902069522887437415446455442405264147381862861931554258095867832462525150868775528162072024799723923276854828384110339149801623963080648198067147537246333033069055966818243104668438970759683086807208275573580830072
e1= 2333
e2= 23333
c1= 28464542187422191031850220803592681443982634383785165404787481127746742239081112082691277387380864221897493018787897237288845518265099977907474953791840485909853466983639444091059228300562651089136949321590723345012238904080799669440783536285513938852463305681933753888253442824802846555416205812335548719095747051066829873263830078172121545700751405449738971
```

```
671084533978188308628269580701777835171248459106590722728629846143810627614929043616330287139900536141060815400762292597226714159359740925698037765375797545038949245031095474474127089451563975157287814950177766322381926627164489617747258380900865129221049 59
c2= 254607406143010549413074171742773473225255537967961962887527699078639553987655153353807785291836214843391979149892566627741988643511776546241977759030297037568619554425840458619866368648759012261359767366719915192788058876171126797313042364147951410910739658163013440999459163818536388673288981323442142669333612391316448541178212048681715051335398611912622655994538300093335802968525181599840591207275218189243053492302669515539970053517790687828515927854296701309752510071220367335444874957037548953686384013477073841141654057254746472880454809045439345630926733935238742948307397294226538194212945717801022072 92072
```

这道题目给出两组密文，但加密模数n是相同的，加密指数e不同，且$GCD(e1,e2)=1$，是典型的共模攻击。这里抄写推导一下共模攻击的公式。

已知e_1,e_2，且$GCD(e_1,e_2)=1$， 由扩展欧几里得算法，存在一组整数s_1,s_2，使得：

$$e_1 \times s_1 + e_2 \times s_2 = 1$$

$$\because \begin{cases} c_1 \equiv m^{e_1} \pmod n \\ c_2 \equiv m^{e_2} \pmod n \end{cases}$$

则有：

$$\begin{aligned} m &\equiv m^1 \quad \pmod n \\ &\equiv m^{e_1 \times s_1 + e_2 \times s_2} \quad \pmod n \\ &\equiv m^{e_1 \times s_1} \times m^{e_2 \times s_2} \quad \pmod n \\ &\equiv [(m^{e_1})^{s_1} \quad \pmod n] \times [(m^{e_2})^{s_2} \quad \pmod n] \\ &\equiv c_1^{s_1} \quad \pmod n \times c_2^{s_2} \quad \pmod n \end{aligned} \tag{1}$$

由于c_1,c_2,s_1,s_2都是已知的，所以可求出m的值。

解题代码如下：

```
n= 2914364542125004196461013151979631620937439720415546997643628297027022309322727011693614877504381563454278605395775464858854791668585594323374735508795025542008452920827295972679894477152981228021159524663232416431841456892162090322879231242294904925112467510535709600151190018238498213660846900447587735044376789897398958317312803043494088605279279781654078735861026379810951747640485788485373794685159902069522887437415446455442405264147381862861931554258095867832462525150868775528162072024799723923276854828384110339149801623963080648198067147537246333033069055966818243104668438970759683086807208275573580830072 3
e1= 2333
e2= 23333
c1= 284645421874221910318502208035926814439826343837851654047874811277467422390811120826912773873808642218974930187878972372888455182650999779074749537918404859098534669836394440910592283005626510891369493215907233450122389040807996694407835362855139388524633056819337538882534428248028465554162058123355487190957470510668298732638300781721215457007514054497389715671084533978188308628269580701777835171248459106590722728629846143810627614929043616330287139900536141060815400762292597226714159359740925698037765375797545038949245031095474474127089451563975157287814950177766322381926627164489617747258380900865129221049 59
c2= 254607406143010549413074171742773473225255537967961962887527699078639553987655153353807785291836214843391979149892566627741988643511776546241977759030297037568619554425840458619866368648759012261359767366719915192788058876171126797313042364147951410910739658163013440999459163818536388673288981323442142669333612391316448541178212048681715051335398611912622655994538300093335802968525181599840591207275218189243053492302669515539970053517790687828515927854296701309752510071220367335444874957037548953686384013477073841141654057254746472880454809045439345630926733935238742948307397294226538194212945717801022072 92072

```

```python
assert gmpy2.gcd (e1, e2) == 1
_, s1, s2 = gmpy2.gcdext(e1, e2) # 结果第一位为标志位
m = pow(c1, s1, n) if s1 > 0 else pow(gmpy2.invert(c1, n), -s1, n) # s1,s2可能为负
m *= pow(c2, s2, n) if s2 >0 else pow(gmpy2.invert(c2, n), -s2, n)

print(libnum.n2s(int(m % n)))  #最后相乘的结果要模N
# flag{01645dce-022e-49cb-9523-4b4d8e84a0e2}
```

中间还有两个地方需要注意一下。一是s_1, s_2的值有一个为负值。
在数论中，求一个数a模n的-m次幂，等于这个数a模n逆元的m次幂

$$n^{-m} \equiv (n^{-1})^m \pmod n$$

另一处是，最后求得的m值还要再模n

风二西_RSA4

题目：好像给错密钥了 风二西原创题

题目给出了两个文件，一个是密文`flag.pem` 另一个是公钥文件`pubckey.pem`
用文本编辑器打开题目给的密钥，发现给的是一个私钥言文件：`-----BEGIN RSA PRIVATE KEY-----`
有了私钥了，那还不直接解呀，

代码如下：

```python
from Crypto.PublicKey import RSA
import libnum
with open("flag.pem", 'rb') as f:
    c = int(f.read().hex(),16)

with open("pubckey.pem", 'rb') as f:
    key = f.read()

rsakey = RSA.importKey(key)
# print(rsakey)

d = rsakey.d
n = rsakey.n
m = pow(c,d,n)
print(libnum.n2s(m))
# flag{947ce8a3-40ee-46c0-a00e-0026e583f8da}
```

当然，有了私钥后，解法有很多
推荐使用openssl工具

```
openssl rsautl -decrypt -inkey pubckey.pem -in flag.pem -raw
flag{947ce8a3-40ee-46c0-a00e-0026e583f8da}
```

用cyberchef也能解。

风二西_RSA5

题目：这次密钥没有给错啦

还是给的两个文件，公钥和密文。这一次没有给错，需要从公钥中提取n、e 。
使用openssl工具读

```
openssl rsa -pubin -in pubckey1.pem -text -modulus

Exponent: 65537 (0x10001)
Modulus=BD98A36A39B78C4069B35001669DCEAB74E64569C15B58349839222101D5C79BA3C3855998B4ABF074C2CFD4FA36B406D85C1B935E18EF791C3A2EA33A0834236483F8E024E50659107334C087139B7AC5069644B98615A38E433D79E9EDE167AA05601B915FE01E179E0C6854DAD5B3B58263E793DB62E7674EB72110D2C4C230561DCD8B3E75C03CFF48E1F5057C1FE4B4B8981EE2AF68DB6B14E763F9B2260B1F7C5E4DA274EC91B6A47B6016DA9ED4656E3628D2007F331C4828DD11A5CE028F3D7E49512E69D44C3461AE2CFFED8273DB9AEC17CCB8759D7DB321F6FFD4A6FE90C305430D4210C4DC98BA8E7BF4D8BB9D0CA87FD3D34B97AE4040A27405
```

读到的n是16进制数，共有2048位，一般来讲n超过512位就不建议分解了。
这里出题人为了练习大家工具的使用，给出的n是由两个相邻素数构成的，所以用yafu能分解出来。
解出结果如下：

```
***factors found***

P309 = 154707169869912965572264038807085588160825389898605792084774698420524113924278063744211942790334520717545961407718606461114604633654138753720598358336593011214128451966613914165721695134758193524181137401802709098935179245573044474461382557698071520126327208903405426672668774245334192540031083738467279228573
P309 = 154707169869912965572264038807085588160825389898605792084774698420524113924278063744211942790334520717545961407718606461114604633654138753720598358336593011214128451966613914165721695134758193524181137401802709098935179245573044474461382557698071520126327208903405426672668774245334192540031083738467279227529

ans = 1
```

有了p和q 我们直接去解密文就行了。

```python
from Crypto.PublicKey import RSA
import libnum
with open("flag1.pem", 'rb') as f:
    c = int(f.read().hex(),16)
with open("pubckey1.pem", 'rb') as f:
    key = f.read()

rsakey = RSA.importKey(key)
e = rsakey.e
n = rsakey.n

p = 154707169869912965572264038807085588160825389898605792084774698420524113924278063744211942790334520717545961407718606461114604633654138753720598358336593011214128451966613914165721695134758193524181137401802709098935179245573044474461382557698071520126327208903405426672668774245334192540031083738467279228573
q = 154707169869912965572264038807085588160825389898605792084774698420524113924278063744211942790334520717545961407718606461114604633654138753720598358336593011214128451966613914165721695134758193524181137401802709098935179245573044474461382557698071520126327208903405426672668774245334192540031083738467279227529

def rsa_decrypt(p,q,e,n,c):
```

```python
    """RSA解密函数"""
    phi_n = (p-1)*(q-1)
    assert(libnum.gcd(e,phi_n) == 1) # e与phi_n互素
    d = libnum.invmod(e,phi_n)
    m = pow(c,d,n)
    print(libnum.n2s(m))
    return m

rsa_decrypt(p,q,e,n,c)
# flag{c4a6661d-a8f9-4861-bfb9-871ea31a10ca}
```

风二西_RSA6

题目：不是那么简单了。听过OEAP吗？

百度一下，OEAP 是RSA的一种填充方式-最优非对称加密填充。
好了，不用再看其他了，直接做题

```python
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import libnum
with open("flag2.pem", 'rb') as f:
    ciphertext = f.read()
with open("prikey2.pem", 'rb') as f:
    key = f.read()

rsakey = RSA.importKey(key)
cipher = PKCS1_OAEP.new(rsakey)
message = cipher.decrypt(ciphertext)
print(message)
# b'flag{e5dca96d-f0cb-4bde-b657-2e2589958557}'
```

用openssl来解：

```
openssl rsautl -decrypt -inkey prikey2.pem -in flag2.pem -oaep
flag{e5dca96d-f0cb-4bde-b657-2e2589958557}
```

风二西_RSA7

题目：工具脚本一把梭吧

题目给出的加密指数e非常大。长度与n接近，由于$ed \equiv 1 \pmod{\varphi(n)}$，加密指数e大，对应的解密指数d就比较小。
Wiener 表示如果满足：

$$d < \frac{1}{3}n^{\frac{1}{4}}$$

那么一种基于连分数(一个数论当中的问题)的特殊攻击类型就可以危害 RSA 的安全。具体的原理和推导见Tr0y的RSA 大礼包 我的代码都是抄的。

```python
n= 762300022332431174941609258381030070780599877830122426681549284199147378290632948959222809643267041637609120761516346819035382113913182320432950545053690370374893567906659520404240737003404419760877462980687968070696223466768566052446629232963253328128447548594504195157
```

```python
72460413762564695491785275009170060931
e= 192520671180610666318316537368741687437592254047579964984523833378160718667002256503841810123627397583145162735749421195975790422094883838952768251931182979720309078991885204267419197375732300501126143508685168181127426637133446588254933775128863119608235849925311854442077052131091840762733768785240907623 27
c= 5112936446875965461069196902901807713568128645240372034293070122731827886790249980803978957762531800122509230190288710513105476217822524308843496118943022524100888059998675088164267173759188588177211293243341366112395166695520436585281705030872313310109018335291749094274409249549410869378310814604117324909 6

import gmpy2
import time

# 展开为连分数
def continuedFra(x, y):
    cF = []
    while y:
        cF += [x // y]
        x, y = y, x % y
    return cF

def Simplify(ctnf):
    numerator = 0
    denominator = 1
    for x in ctnf[::-1]:
        numerator, denominator = denominator, x * denominator + numerator
    return (numerator, denominator)

# 连分数化简
def calculateFrac(x, y):
    cF = continuedFra(x, y)
    cF = list(map(Simplify, (cF[0:i] for i in range(1, len(cF)))))
    return cF

# 解韦达定理
def solve_pq(a, b, c):
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) // (2 * a), (-b - par) // (2 * a)

def wienerAttack(e, n):
    for (d, k) in calculateFrac(e, n):
        if k == 0: continue
        if (e * d - 1) % k != 0: continue

        phi = (e * d - 1) // k
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return abs(int(p)), abs(int(q))
    print('not find!')

p, q = wienerAttack(e, n)

def rsa_decrypt(p,q,e,n,c):
    """RSA解密函数"""
    phi_n = (p-1)*(q-1)
```

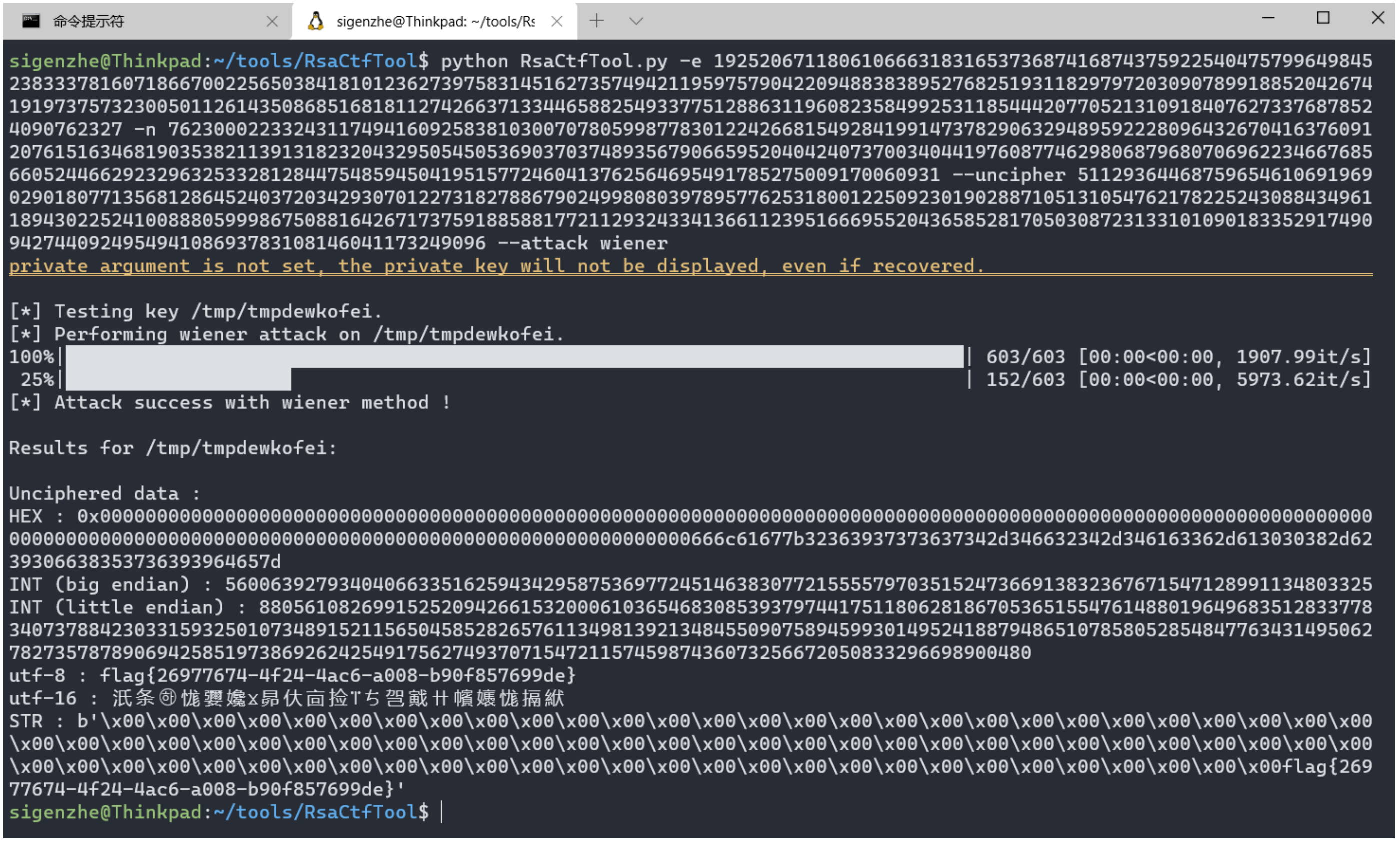
```python
    assert(libnum.gcd(e,phi_n) == 1) # e与phi_n互素
    d = libnum.invmod(e,phi_n)
    m = pow(c,d,n)
    print(libnum.n2s(m))
    return m

rsa_decrypt(p,q,e,n,c)
# b'flag{26977674-4f24-4ac6-a008-b90f857699de}'
```

当然，也有比较成熟的RSA工具可以直接解

```
python RsaCtfTool.py --attack wiener -e 192520671180610666318316537368741687437592254047579
9649845238333781607186670022565038418101236273975831451627357494211959757904220948838389527
6825193118297972030907899188520426741919737573230050112614350868516818112742663713344658825
493377512886311960823584992531185444207705213109184076273376878524090762327 -n 762300022332
4311749416092583810300707805998778301224266815492841991473782906329489592228096432670416376
0912076151634681903538211391318232043295054505369037037489356790665952040424073700340441976
0877462980687968070696223466768566052446629232963253328128447548594504195157724604137625646
95491785275009170060931 --uncipher 51129364468759654610691969029018077135681286452403720342
9307012273182788679024998080397895776253180012250923019028871051310547621782252430884349611
8943022524100888059998675088164267173759188588177211293243341366112395166695520436585281705
0308723133101090183352917490942744092495494108693783108146041173249096

utf-8 : flag{26977674-4f24-4ac6-a008-b90f857699de}
```



风二西_RSA8

题目：奇怪的N

```
n= 16167079541866110894139554776006805335583255507777902785370014044287629807792678603080624326904252123438379392991083602391399498701092433900653669386676307884918986949787175248927731572766954751107930313632638863848068063082267717308481084878455443339438202995673970739570255610513800186878694407787156984477 1
c= 91652340468387584012845155237237896957786753396661434559421169499111938419733760364914054180181470453332534789456757372866493406817246725731113863637159054175158914882334950110118713886213759125279941357012004180349611604118066085014934218543579248275421019690815403585470855502464076600672369539603525850924
e= 65537
```

拿到题目后，提示是奇怪的N，这个N到底奇怪在哪里，咱也不知道，先用yafu分解一波。

```
yafu-x64
factor(16167079541866110894139554776006805335583255507777902785370014044287629807792678603080624326904252123438379392991083602391399498701092433900653669386676307884918986949787175248927731572766954751107930313632638863848068063082267717308481084878455443339438202995673970739570255610513800186878694407787156984477 1)


fac: factoring 16167079541866110894139554776006805335583255507777902785370014044287629807792678603080624326904252123438379392991083602391399498701092433900653669386676307884918986949787175248927731572766954751107930313632638863848068063082267717308481084878455443339438202995673970739570255610513800186878694407787156984477 1
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
Total factoring time = 5.3048 seconds


***factors found***

P309 = 16167079541866110894139554776006805335583255507777902785370014044287629807792678603080624326904252123438379392991083602391399498701092433900653669386676307884918986949787175248927731572766954751107930313632638863848068063082267717308481084878455443339438202995673970739570255610513800186878694407787156984477 1

ans = 1
```

yafu很给力，用了5秒种就分解出来了，怎么只有一个呢，还是原数n。原来n是素数呀。 那就好办了。
根据RSA的原理，我们为什么要分解模数N？ 为的是计算$\varphi(n)$ 因为有$\varphi(n)$ 才能求出解密指数d
$\because n = pq$
$\therefore \varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$

既然n是素数，那么$\varphi(n) = n - 1$ 直接就求出$\varphi(n)$ 。那剩下的就是求解密指数d和明文m

```
import libnum
n= 16167079541866110894139554776006805335583255507777902785370014044287629807792678603080624326904252123438379392991083602391399498701092433900653669386676307884918986949787175248927731572766954751107930313632638863848068063082267717308481084878455443339438202995673970739570255610513800186878694407787156984477 1
c= 91652340468387584012845155237237896957786753396661434559421169499111938419733760364914054180181470453332534789456757372866493406817246725731113863637159054175158914882334950110118
```

```
713886213759125279941357012004180349611604118066085014934218543579248275421019690815403585470855502464076600672369539603525850924
e= 65537

phi_n = n - 1
d = libnum.invmod(e, phi_n)
m = pow(c,d,n)
print(libnum.n2s(m))
# b'flag{c9991f77-c898-406a-8785-5a6db8081533}'
```

风二西_RSA9

题目：不装了，摊牌了，都给你说啦。

```
import gmpy2
import uuid
import random
p=libnum.generate_prime(1024)
q=libnum.generate_prime(1024)
flag="flag{"+str(uuid.uuid4())+"}"
n=p*q
phi=(p-1)*(q-1)
while True:
    e=random.randint(10000,65537)
    if gmpy2.gcd(e,phi)==1:
        break
d=libnum.invmod(e,phi)
c=libnum.s2n(flag)
m=pow(c,d,n)
print("p=",p)
print("q=",q)
print("m=",m)
print("d=",d)
print("n=",n)
p= 97368485043593006405417010779079380120093793034626963175677083523770994936525207940968193918786949567460392401775664093619173263261961563254058029894381986376275758006361044924787173495349206752585567053148516364028668638365676608691913705470536048404291284013185217624584284180593606281872606674303227862923
q= 174034507670751544663833619122758532253821916016434057019886746556436073565496931298817122456263338650062754783803599969233256462434713772953714031268092314238169815901012809393600325432808839406464715247202866205781781379919342815514475667193698142923567276511836660769097557234679842172400378371421781964289
m= 10713159978080595248303368136468725248428004219265383017568301839870142446158283601395319409739267941310957665427316430008931224971372871063315206268306110608326979812846886034642104794304853873192876460915559231227506777599442060327993342928268503696889652417983518056546140617858431621418835939458147783959307745523951841137415442238550765735249662192808694822637569812286855868419594276671181075389377949452992838748913612817680444419095179848524538089268826799430007066454995062821698762487766844583651126504070187331835027249508337718321126942529727464313184539455069391263828081876598132257030625297646910710698
d= 93902374696256253277677727725018601047581014416761470564137333784378485597720904499064441463139653206082160541165140827005255632438437712049018395033074791289679254384075627183440329528751233648165068106388569258641607250419816471214651621908074360284965690311783928905531773993550875532096254557340164566101919950054217615657333580709018009530038656656376147711050802659505753704218821249929564155862367764811161261061717711560407127505601539512763877
```

```
119911944666539743684678619520585430350308169934780059800293152686102504488209423444323920487008224418498167752521850855932776977720137948334199460993604407720031 35
n= 1694547635720864412298198176937464629392610555347329769761469069265260171332222720866197531218893840755536086414858403809232375306355250466610171993481097363263456597501549452949187872745918123040683278839396624995572407884802195983677329647988221841356166802575612688016547168224649127552324065997647461818716635704026222331324275613285012416381212513831762078935831009497089741786327809138324211976558278245117317488673983328457959325296906397222649084947376075321906983415536418106255577602944933237768805265998149213477922664222500542744949440780605166536231957382670255900678321330626237690322914686981857315674 7
```

分析题目中给出的内容，发现m就是密文，我们要求的明文是c。字母换了换。
已知$m \equiv c^d \pmod n$,
由于 $ed \equiv 1 \pmod{\varphi(n)}$
所以 $c \equiv m^e \pmod n$

到这里我们发现，没有给出e的值。但没有关系，我们有$d, p, q, \varphi(n)$ 可以求。

```python
p= 97368485043593006405417010779079380120093793034626963175677083523770994936525207940968193918786949567460392401775664093619173263261961563254058029894381986376275758006361044924787173495349206752585567053148516364028668638365676608691913705470536048404291284013185217624584284180593606281872606674303227862923
q= 17403450767075154466383361912275853225382191601643405701988674655643607356549693129881712245626333865006275478380359996923325646243471377295371403126809231423816981590101280939360032543280883940646471524720286620578178137991934281551447566719369814292356727651183666076909755723467984217240037837142178196428 9
m= 10713159978080595248303368136468725248428004219265383017568301839870142446158283601395319409739267941310957665427316430008931224971372871063315206268306110608326979812846886034642104794304853873192876460915559231227506777599442060327993342928268503696889652417983518056546140617858431621418835939458147783959307745523951841137415442238550765735249662192808694822637569812286855868419594276671181075389377949452992838748913612817680444419095179848524538089268826799430007066454995062821698762487766844583651126504070187331835027249508337718321126942529727464313184539455069391263828081876598132257030625297646910710698
d= 93902374696256253277677727725018601047581014416761470564137333784378485597720904499064441463139653206082160541165140827005255632438437712049018395033074791289679254384075627183440329528751233648165068106388569258641607250419816471214651621908074360284965690311783928905531773993550875532096254557340164566101919950054217615657333580709018009530038656656376147711050802659505753704218821249929564155862367764811161261061717711560407127505601539512763877119911944666539743684678619520585430350308169934780059800293152686102504488209423444323920487008224418498167752521850855932776977720137948334199460993604407720031 35
n= 1694547635720864412298198176937464629392610555347329769761469069265260171332222720866197531218893840755536086414858403809232375306355250466610171993481097363263456597501549452949187872745918123040683278839396624995572407884802195983677329647988221841356166802575612688016547168224649127552324065997647461818716635704026222331324275613285012416381212513831762078935831009497089741786327809138324211976558278245117317488673983328457959325296906397222649084947376075321906983415536418106255577602944933237768805265998149213477922664222500542744949440780605166536231957382670255900678321330626237690322914686981857315674 7
import libnum
phi_n = (p-1)*(q-1)
e = libnum.invmod(d,phi_n)
c = pow(m, e, n)
print(libnum.n2s(c))
# flag{0ba0fcc8-88d2-4f78-b1e2-e3823539340c}
```

风二西_RSA10

题目：不装了，摊牌了，这次全部都参数都给你，包括e（风二西原创题）

```python
import uuid
import libnum
import gmpy2

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
e = 65537
m = libnum.s2n(flag.encode())
p1 = libnum.generate_prime(128)
q1 = libnum.generate_prime(128)
p2 = libnum.generate_prime(128)
q2 = libnum.generate_prime(128)
print("p1=",p1)
print("q1=",q1)
print("p2=",p2)
print("q2=",q2)
n1=p1*q1
n2=p2*q2
print("n1=",n1)
print("n2=",n2)
c1 = pow(m, e, n1)
c2 = pow(m, e, n2)
print("c1=",c1)
print("c2=",c2)
p1= 241529374856419543994843741620715478233
q1= 329891612475502969315412700917758756573
p2= 179415062328238613586720079938194290751
q2= 281209161331996176661322999324485217597
n1= 79678514931584446837886795964984740987618425126262080131520484181733127175509
n2= 50453159207651801862952938090505477143503284591035016948403490994601319545347
c1= 10906371165492800616190805676717306177005704888515733402096006986355132032250
c2= 47055855052437161522184969745110429012879528443871661682592147046669796586664
```

这道题目当时做的时候还卡了两天。是对中国剩余定理不够了解，看了风大的视频后才明白。
题目给出的两组模数n_1, n_2特别小，如果明文m大于模数n，是没有办法直接恢复的。
使用中国剩余定理，可以得到$c \pmod{n}, n = n_1 \times n_2$ 模数被扩展到512位。
我们尝试用这一组的 c, e, n来解。其中$\varphi(n) = \varphi(n_1)\varphi(n_2) = (p_1 - 1)(q_1 - 1)(p_2 - 1)(q_2 - 1)$

```python
p1= 241529374856419543994843741620715478233
q1= 329891612475502969315412700917758756573
p2= 179415062328238613586720079938194290751
q2= 281209161331996176661322999324485217597
n1= 79678514931584446837886795964984740987618425126262080131520484181733127175509
n2= 50453159207651801862952938090505477143503284591035016948403490994601319545347
c1= 10906371165492800616190805676717306177005704888515733402096006986355132032250
c2= 47055855052437161522184969745110429012879528443871661682592147046669796586664
e = 65537

import libnum
```

```python
import gmpy2
from functools import reduce
def CRT(mi, ai):
    # mi,ai分别表示模数和取模后的值,都为列表结构
    # Chinese Remainder Theorem
    assert (isinstance(mi, list) and isinstance(ai, list))
    M = reduce(lambda x, y: x * y, mi)
    ai_ti_Mi = [a * (M // m) * gmpy2.invert(M // m, m) for (m, a) in zip(mi, ai)]
    return reduce(lambda x, y: x + y, ai_ti_Mi) % M

c = CRT([n1,n2],[c1,c2])
phi_n = (p1-1)*(q1-1)*(p2-1)*(q2-1)

d = libnum.invmod(e, phi_n)
m = pow(c, d, n1*n2)
print(libnum.n2s(m))
# flag{b204d8a4-0186-48c4-9c5f-2d02d267c326}
```

风二西_RSA11

题目：很简单的

```
n= 2402438576165473029108412792590900622915797278696450348937297683358289425471810609861361183936792588570688309584438708660907863882101777257628990375920892592671725390629604688454167797154718295679433805383082825684737247057084904476690427408950519465993155695550296000374172597794486760547755879527113453
e= 3
c= 175676150266627654394509074891404164566854171033140030366264579869316998382531731238646439305983889007988827572538127555289345112927153391354273822296560289724299704969505044786520464609064991216105190142528210147105407231359976850587913961569714117627302606370251386092433653181453744354380262673514341
```

这个题目$e=3$是典型的低加密指数攻击。
$\because c \equiv m^3 \pmod{n}$

$\therefore m=\sqrt[3]{c+kn}$

一般情况下，k=0，可以直接开方。

```python
e = 3
c= 175676150266627654394509074891404164566854171033140030366264579869316998382531731238646439305983889007988827572538127555289345112927153391354273822296560289724299704969505044786520464609064991216105190142528210147105407231359976850587913961569714117627302606370251386092433653181453744354380262673514341
import gmpy2
import libnum
m = gmpy2.iroot(c,e)[0]
print(libnum.n2s(int(m)))
# flag{c47361b6-7627-4a4f-baf7-21001f967b56}
```

风二西_RSA12

题目：中国剩余定理吧，e是多少呢？

```python
import uuid
import libnum
import gmpy2
import random
flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m = libnum.s2n(flag)
p = libnum.generate_prime(1024)
q = libnum.generate_prime(1024)
n1=p*q
p = libnum.generate_prime(1024)
q = libnum.generate_prime(1024)
n2=p*q
p = libnum.generate_prime(1024)
q = libnum.generate_prime(1024)
n3=p*q
while 1:
    e=random.randint(10,20)
    print(e)
    if gmpy2.is_prime(e):
        break
c1=pow(m,e,n1)
c2=pow(m,e,n2)
c3=pow(m,e,n3)

print("n1=",n1)
print("n2=",n2)
print("n3=",n3)

print("c1=",c1)
print("c2=",c2)
print("c3=",c3)
n1= 17252219271006504217652265353315644822689102990264622695769709059400124061303815961435216889436124568403595230367283469177657079450282677401376417062215262907612929047987828085558578625928693928880671915353617997183001765819850856038580986341451354677532575518429983180603767256598249850685593173010163922025509601326326321500794396556036639237902931276631659616154399922016651201757258733618622358585354871247473186946333948694594826747068676868873107272038233419801485049530755094873655667843995961470816804517358517607436960955104105230718119794284575683888642896061373529872181319831826417002365270413786886230051
n2= 19392925441119564341555603187441887561478389980762223332359001899590543508270070429654062355050348208292024600480329586395279855747079257790641213158649916066398059169788447789331043090129266878932663273082843414238406770932078873343241723699886063134727135260077063212826785200734480057973678585768855053011930755630200714822903690528979185570885528147643237687242400511057790185488304578078790856756920715781906997572283811891912799721107417543601482580382717235530747319855573633433655463698238396450160804967421072666570521835633096983265294367798652838134815267442249458052921576108382421808074197496756326651569
n3= 16308931821160738277381987335188301508935609331979902438602520050075201076202607797174570476508752749939535558232194093916143544677788076366822749055817956020216396168552177076772752147703927950913692270761747051591209445357878745137592704614863982995068341117660355411746655172287336173901923306626058276062979558685583245075792769514424287449989413646849923270337368271506480123292135149298907255291763788483970298044852870009792041207447026753646424275224352404748955511851564024075916958820790827323891618229420999340841099848014212559544340277261118686124000247691040332469447586013408096906429110903073529177959
c1= 98626749160425299185360923381864280789821656448482221613525239970874508887826856666999
```

```
6847951244365280637841803438717743795168091292517068068919062777145241306310697080809891248377646719341230881649623908246452956514392532087121381507177426996878604461394499084196249530601094001617489710803912131948279913297986636209231828019500643894617602429551750854441781737312059554753531678468044053831648273296572848789992174657175515114338665837154883246900742720531354855559563140404249577415693939022593259289194459953438105757899587447271931995649255159342329359652305879885554769630544770966155000620489702683568172742
c2= 11681944755560676324959185187799099414481370223339263320227032572815322776036541246094706574111230437855451279755698238561172128989213481236089341690470560136591884018327949217955636371986098667096134214617286477749483182783065573382874881799002834245675144902527287434610910847735302841154506319451669009489558469224087072461557995913424041810554224864384732427620959534242904535632660788940451410731486496846273769609538396613198060913461220093973836108967904185914468061114191996582010088055074020730119033560498292937987199687933047535216255557403558690069800171880133754299157682045716215838413976578382030795387
c3= 12668278504770849369921196574913946259264733368388011806162184024969059726807829014394443240915036260802428499115569055027868892016678593665990881075365267878346964234838583098736850357277091593641798510490187797581111760056196425611939907216616878668062988351559417972184145303414105959155690055918434365061272003723017407148417783932381053632869804167234724945360916258465050598759067199479571813805864105608035898489711508922286844624373009203267737364854355266040795917980105237637939321394405488989345144898194063338461873834505409676538338655407786146050320466683048676780486336615260073541849747038981327813976
```

这个题目已经提示了中国剩余定理，并且给出了多组的nc对，只是没有给出e，
但是题目给出了e的范围是（10，20）之间的整数，而且还是素数。那么e只可能是(11,13,17,19)中的一个。
我们先用中国剩余定理求出一c，再尝试开方即可

```
import gmpy2
import libnum

n1= 17252219271006504217652265353315644822689102990264622695769709059400124061303815961435216889436124568403595230367283469177657079450282677401376417062215262907612929047987828085558578625928693928880671915353617997183001765819850856038580986341451354677532575518429983180603767256598249850685593173010163922025509601326326321500794396556036639237902931276631659616154399922016651201757258733618622358585354871247473186946333948694594826747068676868873107272038233419801485049530755094873655667843995961470816804517358517607436960955104105230718119794284575683888642896061373529872181319831826417002365270413786886230051
n2= 19392925441119564341555603187441887561478389980762223332359001899590543508270070429654062355050348208292024600480329586395279855747079257790641213158649916066398059169788447789331043090129266878932663273082843414238406770932078873343241723699886063134727135260077063212826785200734480057973678585768855053011930755630200714822903690528979185570885528147643237687242400511057790185488304578078790856756920715781906997572283811891912799721107417543601482580382717235530747319855573633433655463698238396450160804967421072666570521835633096983265294367798652838134815267442249458052921576108382421808074197496756326651569
n3= 16308931821160738277381987335188301508935609331979902438602520050075201076202607797174570476508752749939535558232194093916143544677788076366822749055817956020216396168552177076772752147703927950913692270761747051591209445357878745137592704614863982995068341117660355411746655172287336173901923306626058276062979558685583245075792769514424287449989413646849923270337368271506480123292135149298907255291763788483970298044852870009792041207447026753646424275224352404748955511851564024075916958820790827323891618229420999340841099848014212559544340277261118686124000247691040332469447586013408096906429110903073529177959
c1= 986267491604252991853609233818642807898216564484822216135252399708745088878268566669996684795124436528063784180343871774379516809129251706806891906277714524130631069708080989124837764671934123088164962390824645295651439253208712138150717742699687860446139449908419624953060109400161748971080391213194827991329798663620923182801950064389461760242955175085444178
```

```
1737312059554753531678468044053831648273296572848789992174657175515114338665837154883246900742720531354855559563140404249577415693939022593259289194459953438105757899587447271931995649255159342329359652305879885554769630544770966155000620489702683568172742
c2= 11681944755560676324959185187799099414481370223339263320227032572815322776036541246094706574111230437855451279755698238561172128989213481236089341690470560136591884018327949217955636371986098667096134214617286477749483182783065573382874881799002834245675144902527287434610910847735302841154506319451669009489558469224087072461557995913424041810554224864384732427620959534242904535632660788940451410731486496846273769609538396613198060913461220093973836108967904185914468061114191996582010088055074020730119033560498292937987199687933047535216255557403558690069800171880133754299157682045716215838413976578382030795387
c3= 12668278504770849369921196574913946259264733368388011806162184024969059726807829014394443240915036260802428499115569055027868892016678593665990881075365267878346964234838583098736850357277091593641798510490187797581111760056196425611939907216616878668062988351559417972184145303414105959155690055918434365061272003723017407148417783932381053632869804167234724945360916258465050598759067199479571813805864105608035898489711508922286844624373009203267737364854355266040795917980105237637939321394405488989345144898194063338461873834505409676538338655407786146050320466683048676780486336615260073541849747038981327813976

from functools import reduce
def CRT(mi, ai):
    # mi,ai分别表示模数和取模后的值,都为列表结构
    # Chinese Remainder Theorem
    assert (isinstance(mi, list) and isinstance(ai, list))
    M = reduce(lambda x, y: x * y, mi)
    ai_ti_Mi = [a * (M // m) * gmpy2.invert(M // m, m) for (m, a) in zip(mi, ai)]
    return reduce(lambda x, y: x + y, ai_ti_Mi) % M

c = CRT([n1,n2,n3],[c1,c2,c3])
for i in range(10,20):
    m = gmpy2.iroot(c,i)
    if m[1]:
        print(f"e = {i}")
        print(libnum.n2s(int(m[0])))
# e = 13
# b'flag{326177d1-448f-48a6-abdd-ee2085b28eaf}'
```

题目做到这里，结果已经出来了。这里再多讲几句，也是我当时遇到的疑惑。
既然加密指数比较小，为什么不直接对c_1, c_2, c_3进行开方？
以我们最后拿到的结果为例，明文位数为355位，e为13。m^e的位数在4351位。
而我们的模数n只有2048位，多余的部分进行了模运算，所以不能直接开方。
中国剩余定理计算后的模数为三个n的最小公倍数，位数达到了6142位，显然是能够覆盖m^e的。

风二西_RSA13

题目：一组NC,e很大。（风二西原创题）

```
import gmpy2
import libnum
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)

m1=libnum.s2n(flag[:21])
```

```python
m2=libnum.s2n(flag[21:])
p=libnum.generate_prime(1024)
q1=libnum.generate_prime(1024)
q2=libnum.generate_prime(1024)
n1=p*q1
n2=p*q2
e=65537
c1=pow(m1,e,n1)
c2=pow(m2,e,n2)
print("n1=",n1)
print("n2=",n2)
print("c1=",c1)
print("c2=",c2)
n1= 14344966075668457197926805093252019643096573510969721322719333921005737087381900449540855412424548701204134893993421731511948833767143851141078209382285846501386535725301259909239487164541050781598400430159674920170854404496253398222187307367926322625492497171740002969789815008039987485265892529966452609000393813395779113860224081603944565240880582129475822907507399743470085845133711696533090310948349344084563385664363625995999399255547593760021153639514720877652246066221504678395471466612295627656516850262088529244725814466732348833167839822545350219641690764981001428685344377698209527688737137187300399652993
n2= 12354218348882658855939557374675440240130593637412960903022784670883099080512624736064231265886849271910351937451997751749093644637238859633012706230887428376122240805168054483532576336852668163683367625468673502957503469509123739755819238781219308756042088817872420155442438195380803116150498980766272244743798236039791910368557658313836740247547559284129302204970113950049895139613500250641481434700366895302374051200743877172702615189237557067101522920447615629272986158643661393610029180590415625821891554368097074431230829448921269825593886700360922071375933729554358494473990274497958858233048122738508242423439
c1= 3691123354676066697059437937998979583921581870731150079375228838089668944155347573115895745939257000857077746797136413811508634456041342500337302101225465437119807883492455873112323783750970718926005237215996705821476132706795935577193711973896496979170206973337515061654337654274989697617761256418417804735475939505284666094122467326381674631123530317439987013961787043442129186254743721364414447230567446822751926033047452822024681275085831713997686837779667260888016072279588480296674703292539294133901503225011837599317942181453227103095922082421095286788644523456206687517063265624595424684296819286062613760042
c2= 5938408686309543013293741535567205151131688865994815122696735470416646965603186344765655319536668785559696396104442531438099421602193304800760866674863437267379695326984063895846835762801125090313445489510683263350964708721907377863520679697909959118156916287172182350863938539461166083625073905275280319815502103888903393913834228224181613416918191464536553417181893783896959345196250270792673371830038465845317178867867078669131493698960171589833480768462627600224827986249429611594755180062966252940786732981155206917654348842803640264421695660795812804373210578066663145314490990520854537552802784407185492236744

```

题目给出的两个模数n 不是互素的，有共同的公约数p，可以直接求出。
另外flag被分成了两块，分别求出两部分后合并一起就行。

```python
import libnum
n1= 14344966075668457197926805093252019643096573510969721322719333921005737087381900449540855412424548701204134893993421731511948833767143851141078209382285846501386535725301259909239487164541050781598400430159674920170854404496253398222187307367926322625492497171740002969789815008039987485265892529966452609000393813395779113860224081603944565240880582129475822907507399743470085845133711696533090310948349344084563385664363625995999399255547593760021153639514720877652246066221504678395471466612295627656516850262088529244725814466732348833167839822545350219641690764981001428685344377698209527688737137187300399652993
```

```
n2= 123542183488826588559395573746754402401305936374129609030227846708830990805126247360642
3126588684927191035193745199775174909364463723885963301270623088742837612224080516805448353
2576336852668163683367625468673502957503469509123739755819238781219308756042088817872420155
4424381953808031161504989807662722447437982360397919103685576583138367402475475592841293022
0497011395004989513961350025064148143470036689530237405120074387717270261518923755706710152
2920447615629272986158643661393610029180590415625821891554368097074431230829448921269825593
886700360922071375933729554358494473990274497958858233048122738508242423439
c1= 369112335467606669705943793799897958392158187073115007937522883808966894415534757311589
5745939257000857077746797136413811508634456041342500337302101225465437119807883492455873112
3237837509707189260052372159967058214761327067959355771937119738964969791702069733375150616
5433765427498969761776125641841780473547593950528466609412246732638167463112353031743998701
3961787043442129186254743721364414447230567446822751926033047452822024681275085831713997686
8377796672608880160722795884802966747032925392941339015032250118375993179421814532271030959
22082421095286788644523456206687517063265624595424684296819286062613760042
c2= 5938408686309543013293741535567205151113168886599481512269673547041664696560318634476565
5319536668785559696396104442531438099421602193304800760866674863437267379695326984063895846
8357628011250903134454895106832633509647087219073778635206796979099591181569162871721823508
6393853946116608362507390527528031981550210388890339391383422822418161341691819146453655341
7181893783896959345196250270792673371830038465845317178867867078669131493698960171589833480
7684626276002248279862494296115947551800629662529407867329811552069176543488428036402644216
95660795812804373210578066663145314490990520854537552802784407185492236744
e=65537
p = libnum.gcd(n1,n2)
print(p)

def de(p,q,n,e,c):
    phi = (p-1)*(q-1)
    d = libnum.invmod(e,phi)
    m = pow(c, d, n)
    return m

q1 = n1//p
q2 = n2//p
m1 = de(p,q1,n1,e,c1)
m2 = de(p,q2,n2,e,c2)

print(libnum.n2s(m1) + libnum.n2s(m2))
```

风二西_RSA14

题目：dp而已

```
n= 2262295349990640830156496792124598886049539109713295422027443292428750908563241794577088
1691534975145091374242390427640088367167203579325929612016723243658261593025738254367644091
6031180903709751697701274622814553111682767057975812367565665565724771477944825765665761415
0365736687770621682205134881509395282493512655414461616811799822654629898935897383568653593
6849349204968416839514589022517798087249227968029567878985549747691248450050900456070085755
1118834562858430677400649876424422653664460458412627262643268340282917260941393159801618738
87433830833454445599669910399735340574524109558630513082823400621144454351
e= 65537
c= 4697226541296714192497595229519045916674902115970897546284870763485626056152843554427993
8257701135022348486327478559484526182229484954723604141736712869566179586822555804984994096
9387805053378233945487237026625762800225165430228884868681682886651471380372547431200184327
2938469580339833821796230678064851622010795748370902906629234916934410260675804537952337230
```

```
0899557527812542650225178659123843514111567275581372438953214890733649204082908085182378787
9326475426928069825609696935156088233034235751099793215905063550820628171197918871238376618
29183552453536333713432175704374084931061318514343544246403425225613 6947
dp= 126729618715997639432376012471064242072409613989447605497725787230926131930047985649208
4933737479817085150849504245445606075365237394812485362639706832584778084898834649519014907
4386000050424608667835925716235725421761075053255329400061125729192540049883051949174136452
78986800555322869842110810763389 31327137
```

这道题目是dp泄漏，其中$d_p \equiv d \pmod{(p-1)}$。关于dp泄漏的相关知识，我都是看风大的dp视频学习的dp泄漏，这里将推导的过程摘抄如下：

$d_p \equiv d \pmod{p-1}$

两边同时乘以e

$e \cdot d_p \equiv e \cdot d \pmod{(p-1)}$
转换成普通的等式写法
$e \cdot d_p = e \cdot d + k_1(p-1)$
根据RSA的基本原理：
$\because e \cdot d \equiv 1 \pmod{\varphi(n)}$
$and \because \varphi(n) = (p-1)(q-1)$
$\therefore e \cdot d = 1 + k_2(p-1)(q-1)$
代入上式，消除d，得到
$e \cdot d_p = 1 + k_2(p-1)(q-1) + k_1(p-1)$
两边同时模上$(p-1)$ 得到：
$e \cdot d_p \equiv 1 \pmod{(p-1)}$
因为p为素数，根据欧拉函数的计算方法 $\varphi(p) = p - 1$
所以上式可写为：
$e \cdot d_p \equiv 1 \pmod{\varphi(p)}$
到这里，就可以明显看出 e, d_p 是关于模数p的一组RSA密钥对。

然后我们再来推导一下解密过程$m \equiv c^{d_p} \pmod{p}$
$m \equiv c^d \pmod{n}$
$m = c^d + kn$
因为$n = p \cdot q$ 上式两边同时模p
$m \pmod{p} \equiv (c^d \pmod{p}) + (kpq \pmod{p})$
当 $m < p$ 时，等式左边 $m \pmod{p} = m$
所以最终变成下面的式子
$m \equiv c^d \pmod{p}$
接下来我们要想办法将d给换成d_p
$\because d_p \equiv d \pmod{(p-1)}$
$\therefore d = d_p + k(p-1)$
代入上面的式子消除d得到
$m \equiv c^{(d_p + k(p-1))} \pmod{p}$
$m \equiv c^{d_p} \times c^{k(p-1)} \pmod{p}$

这里要用到费马小定理，当p是素数时，对于任意的a有：
$a^{p-1} \equiv 1 \pmod{p}$
所以上面的式子，可以变为：
$m \equiv (c^{d_p} \pmod{p}) \times ((c^k)^{p-1} \pmod{p})$
$m \equiv c^{d_p} \pmod{p}$

至此，我们求m的等式已经推导出来了。但是还是不够，我们不知道模数p.
求p的关键在于我们上面求出的$e \cdot d_p \equiv 1 \pmod{(p-1)}$
$e \cdot d_p = 1 + k \cdot (p-1)$
$\Rightarrow e \cdot d_p > k \cdot (p-1)$

$\because d_p \equiv d \mod (p-1)$
$\therefore d_p < (p-1)$
$\therefore k < e$
我们知道了k的范围，就可以通过爆破k来求p，至此，我们求解的关键思路都理清楚了。
求解代码如下：

```python
c= 4697226541296714192497595229519045916674902115970897546284870763485626056152843554427993825770113502234848632747855948452618222948495472360414173671286956617958682255580498499409693878050533782339454872370266257628002251654302288848686816828866514713803725474312001843272938469580339833821796230678064851622010795748370902906629234916934410260675804537952337230089955752781254265022517865912384351411156727558137243895321489073364920408290808518237878793264754269280698256096969351560882330342357510997932159050635508206281711979188712383766182918355245353633371343217570437408493106131851434354424640342522561369 47
dp= 1267296187159976394323760124710642420724096139894476054977257872309261319300479856492084933737479817085150849504245445606075365237394812485362639706832584778084898834649519014907438600005042460866783592571623572542176107505325532940006112572919254004988305194917413645278986800555322869842110810763389313271 37
e= 65537
import libnum
for k in range(e,1,-1): # 因为k与e相近，所以倒着求更快
    if (e*dp -1) % k == 0:
        p = (e*dp) // k + 1
        if libnum.prime_test(p):  # 能够求出多个p，要加一下素数的条件
            print(p)
            print(libnum.n2s(pow(c,dp,p)))
# b'flag{4943d89d-653a-455a-9ca5-9f1d3bdd9516}'
```

d_p泄露的题目还是比较常见的，其中最重要的一点是$m < p$，p都是512位起的，小了容易被分解，所以一般情况下明文`flag`都是小于p
今年的西湖论剑杯hardrsa就是一道dq泄漏。
关于上面的数论推导过程，如果不想看的话只需记住下面的结论即可：
$m \equiv c^{d_p} \pmod p$，
已知e, d_p, c，根据$(e \cdot d_p - 1) \div k = (p-1)$，$k < e$ 爆破k求p

风二西_RSA15

题目：dpdq跑脚本

```python
p= 11245499463097885000578465127602232754578619820574459743188868093765720319294 3
q= 11108177178097830044220820125625193310060722730881915649118288172371496891383 3
c= 784714058062701278289979845773696137695376868466715900847055678639088780525332621169192372484680870446239674610533199192404881981432254030628216401206642 6
dp= 99016059099144522019375365089687785694029213535292918424815544402513220169503
dq= 79504900574184798493105575420403885224379864982754477219462523963780735261625
```

这道题跟上一题有相似的地方，给出了p, q, d_p, d_q 没有给出e。
先看一下已知条件：

$$
\begin{cases}
c & \equiv m^e \pmod n \\
m & \equiv c^d \pmod n \\
\varphi(n) & = (p-1)\cdot(q-1) \\
e \cdot d & \equiv 1 \pmod{\varphi(n)} \\
d_p & \equiv d \pmod p \\
d_q & \equiv d \pmod q
\end{cases}
$$

目的很明确，要想得到m，就要得到c^d。
$m = c^d + k \cdot p \cdot q$ 对两边同时模上p和q，得到

$$
\begin{cases}
m_p & \equiv c^d \pmod p \\
m_q & \equiv c^d \pmod q
\end{cases}
$$

其中式1 可写为 $c^d = m_p + kp$的形式，然后代入到式2，得到
$m_q \equiv (m_p + kp) \pmod q$
两边同时减去m_p
$m_q - m_p \equiv kp \pmod q$
由于 p, q是素数，$GCD(p, q) = 1$，这里可以求得p模q的逆元p^{-1}
两边同乘以逆元p^{-1} 得到：
$(m_q - m_p)p^{-1} \equiv k \pmod q$
将得到的k值代入到 $c^d = m_p + kp$ 中，最终得到：
$c^d = m_p + ((m_q - m_p) \cdot p^{-1} \pmod q) \cdot p$

上式中的m_p, m_q 可以用以下过程推导：
$m_p \equiv c^d \pmod p$
将$d = d_p + k(p-1)$ 代入上式
$m_p \equiv c^{d_p + k(p-1)} \pmod p$ 可以写成：
$m_p \equiv (c^{d_p} \pmod p) \cdot ((c^k)^{(p-1)} \pmod p) \pmod p$

由费马小定理可知，当p为素数时，对于任意a，有 $a^{p-1} \equiv 1 \pmod p$
所以我们可以将c^k 看做是a，$(c^k)^{(p-1)} \equiv 1 \pmod p$
最终可以化简为：$m_p \equiv c^{d_p} \pmod p$
同理可得$m_q \equiv c^{dq} \pmod q$

最终求解的等式为：
$m \equiv (m_p + ((m_q - m_p) \cdot p^{-1} \pmod q) \cdot p) \pmod n$

等式右边的数字全部是已知的，写代码求解就完了。

```
p= 112454994630978850005784651276022327545786198205744597431888680937657203192943
q= 111081771780978300442208201256251933100607227308819156491182881723714968913833
c= 7847140580627012782899798457736961376953768684667159008470556786390887805253326211691923724846808704462396746105331991924048819814322540306282164012066426
dp= 99016059099144522019375365089687785694029213535292918424815544402513220169503
dq= 79504900574184798493105575420403885224379864982754477219462523963780735261625

import libnum
mp = pow(c,dp,p)
mq = pow(c,dq,q)
p_1 = libnum.invmod(p,q)
m = (mp + (((mq-mp)*p_1)%q)*p) % (p*q)
print(libnum.n2s(m))
# b'flag{96bd68e0-983e-4683-83c5-9cde3d18bea3}'
```

14和15题的推导过程看似很复杂，实际上不难，可以自已练习几次。

风二西_RSA16

题目：分解N 有惊喜哦

```
n= 24090704838485320092322602381433113998549491255884266387675228929661104335500411198637928025660996430319332941004040599396998234228244782303357297973137930526505489726115260523310511218709466132734290380881163955161225457364375241182423158684922361439555422326120328133588500102913773661266808325619034117428588147006907657756318674847307691315077012529837425655081122924789717051817547851005534496378358099420535561566073404733903281708969861402722313932576238811896701757123799634332216554086749941045956406802471674031944558913714990838086806396503547587896668889448877787128717078534093031525842561899729308823124832044199410217169735469224432542618693916399121293260293185310206776155217396940245472443313953985052989506641484804269835655950721932515156631880216443
e= 65537
c= 4644692044652159435824271269188149877191463739428505800552271160405169256688524376059318365826281434840310221997229754973778724418987980440341270686332667226078120978326105359289063570833817421216507957905303736889805936005004450684613091300319138800693741893618264241268661031157574584193673068534877291120825428318673498993334115269043893347607706429720486411605530004387437787775015099350192774546635451473835741062999624582951642205695330470935474403770491104296455303487414392560417505110011204285904941964895708840331681754455406209522192714176572299696109331233432919143365691231618894725659346560911188282377099473664930243190978118442970824453139977801843431119816228255364416557236747200455405887287572620725727247354320774485121369770634224919966258978988036
```

题目给出的n比较大，通过yafu分解，发现n是p的5次方。这道题目考察的知识点是欧拉函数$\varphi(n)$的计算方法。
我们来回忆一下RSA的原理，当 a与n互素时，$a^{\varphi(n)} \equiv 1 \pmod n$
其中为什么n要取值为两个大素数p和q的乘积？因为这样的n是难以分解的。
只要n不能分解，$\varphi(n) = (p-1)\cdot(q-1)$，$\varphi(n)$就算不出来，这样就保证了RSA算法的安全性。
并不是说n只能取p和q的乘积，取其他的n当然也是可以的，只不过安全性没有保证，例如这道题就能被解出来。
求解的关键就是欧拉函数$\varphi(n)$

欧拉函数定义

> 在数论，对正整数n，欧拉函数是小于n的正整数中与n互质的数的数目.
> 在1到8之中，与8形成互质关系的是1、3、5、7，所以 $\varphi(8) = 4$。
> 计算欧拉函数的有以下几种情况：

1. 当n为1时，$\varphi(1) = 1$。因为1与任何数都构成互质关系
2. 当n为质数时，$\varphi(n) = n - 1$。因为质数与小于它的每一个数都互质。
3. 当n为质数的次方时$n = p^k$，则$\varphi(n) = p^k - p^{k-1} = p^k(1-\frac{1}{p})$
4. 当n为两个互质的数之积$n = p \cdot q$, 则$\varphi(n) = \varphi(p)\cdot\varphi(q)$
5. 任意一个大于1的正整数n，都可以写成一系列质数的积$n = p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r}$
   根据第四条结论可以写成：
   $\varphi(n) = \varphi(p_1^{k_1})\varphi(p_2^{k_2})\cdots\varphi(p_r^{k_r})$
   根据第三条结论可以写成：
   $\varphi(n) = p_1^k p_2^k \cdots p_r^{k_r}(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_r})$
   最终的欧拉函数通用计算公式：
   $\varphi(n) = n\cdot(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_r})$

对于这道题目，$n = p^5$ 则有$\varphi(n) = n(1-\frac{1}{p}) = n - p^4$
有了$e, \varphi(n)$ 就正常解密就行。

```
n= 24090704838485320092322602381433113998549491255884266387675228929661104335500411198637928025660996430319332941004040599396998234228244782303357297973137930526505489726115260523310
```

```
511218709466132734290380881163955161225457364375241182423158684922361439555422326120328133588500102913773661266808325619034117428588147006907657756318674847307691315077012529837425655081122924789717051817547851005534496378358099420535561566073404733903281708969861402722313932576238811896701757123799634332216554086749941045956406802471674031944558913714990838086806396503547587896668889448877787128717078534093031525842561899729308823124832044199410217169735469224432542618693916399121293260293185310206776155217396940245472443313953985052989506641484804269835655950721932515156631880216443
e= 65537
c= 4644692044652159435824271269188149877191463739428505800552271160405169256688524376059318365826281434840310221997229754973778724418987980440341270686332667226078120978326105359289063570833817421216507957905303736889805936005004450684613091300319138800693741893618264241268661031157574584193673068534877291120825428318673498993334115269043893347607706429720486411605530004387437787775015099350192774546635451473835741062999624582951642205695330470935474403770491104296455303487414392560417505110011204285904941964895708840331681754455406209522192714176572299696109331233432919143365691231618894725659346560911188282377099473664930243190978118442970824453139977801843431119816228255364416557236747200455405887287572620725727247354320774485121369770634224919966258978988036
import libnum
# yafu 分解n得到p
p = 7522633459543261699928061990854146313832534923510670277767922552283252008995826136321031523873848172725576684552966982932229269010055777902040181800721643
phi_n = n - p**4
assert(libnum.gcd(e, phi_n) == 1)
d = libnum.invmod(e, phi_n)
print(libnum.n2s(pow(c,d,n)))
# b'flag{5f801865-d79a-4443-a62e-7e527ebaba33}'
```

风二西_RSA17

题目：似乎都给你说了，没那么简单

```
q= 9821019070409676333044072841065266478412490068866140735407083020681876874216236257486646854474910118979331501550740926276257953946766816940478029341936251
n= 20065300735475128590762092835634208578590188430910113938266923916956496499667511050434727958038855180430563677067665310984636341807198909493266030524407914174388530285921911038005426931867734450564693608543508971517875518246462535981518861032385301237643992100986849108856013524416300831289325154121110808247211536812259102311581603306878281780717796746256378731029854411630700458188720317727057026867196631023061032601638872839570025991211870975933208177308028632438360982879806030503684724554915521232091140311979213507503765831400499361130450254384304522697771978881836641395836048939329374662229617271102430460909
e= 65537
c= 10334353226901656582025695459419278099579299367981226010947137434420041484901529514923751765488009668423558942300563010202002037412708785365886627854514598472485113085663423766836258491855487234194680311886099056606577829335872927280970941190051132740285138145789581491571992748223980546807928715445558156932190046773138655188050614149792373562068124659615358354617809264387945436262858291997823711827227689741790876680419269257229837590349990437789831556244226011468099456803730676941298395849646182421743150816380697450542133246409603419842118484214641273072148680153105439037855464224041968796128112802295366103465
p = 12689067460968639894065349877170981382620686137782855931246710156025521091715625374157329608017968147868420672538526581198886713432258948704920071988814119
```

这道题目给16题类似，给出的n是2048位，p和q都是512位，

经过函数`libnum.extract_prime_power(n,p)`测试，发现$n = p^3 \cdot q$
根据欧拉函数通用计算公式：
$\varphi(n) = n \cdot (1 - \frac{1}{p})(1 - \frac{1}{q})$
展开后得到：
$\varphi(n) = (n + \frac{n}{pq} - \frac{n}{p} - \frac{n}{q})$

```python
q= 9821019070409676333044072841065266478412490068866140735407083020681876874216236257486646854474910118979331501550740926276257953946766816940478029341936251
n= 20065300735475128590762092835634208578590188430910113938266923916956496499667511050434727958038855180430563677067665310984636341807198909493266030524407914174388530285921911038005426931867734450564693608543508971517875518246462535981518861032385301237643992100986849108856013524416300831289325154121110808247211536812259102311581603306878281780717796746256378731029854411630700458188720317727057026867196631023061032601638872839570025991211870975933208177308028632438360982879806030503684724554915521232091140311979213507503765831400499361130450254384304522697771978881836641395836048939329374662229617271102430460909
e= 65537
c= 10334353226901656582025695459419278099579299367981226010947137434420041484901529514923751765488009668423558942300563010202002037412708785365886627854514598472485113085663423766836258491855487234194680311886099056606577829335872927280970941190051132740285138145789581491571992748223980546807928715445558156932190046773138655188050614149792373562068124659615358354617809264387945436262858291997823711827227689741790876680419269257229837590349990437789831556244226011468099456803730676941298395849646182421743150816380697450542133246409603419842118484214641273072148680153105439037855464224041968796128112802295366103465
p = 12689067460968639894065349877170981382620686137782855931246710156025521091715625374157329608017968147868420672538526581198886713432258948704920071988814119

import libnum
phi_n = n + n//(p*q) - n //p - n//q
assert(libnum.gcd(e,phi_n) == 1)
d = libnum.invmod(e, phi_n)
print(libnum.n2s(pow(c,d,n)))
# b'flag{5a3ecc6d-d052-404a-b162-9429fc181df1}'
```

风二西_RSA18

题目：咦，好像有问题？

```
p= 179604728901172884984546444098336951299567226578637956867234955273321950834253763455974908411986208869587996429208317419915541879829705695083899455584410844347380529355478252408255665437124675873347322572413775553300676365369297224009959989495411452768654413529438720408670663273782756277912404468420962465297
q= 173812808579656497809794049676143957530187304874982238246182308756968315404682483726756662221017110382295012325999009417177266772549812121470834293121487863992068948158282535382396055587629081656872380987341085769570778778669919312677737719340626660293905750991112916961702537446688565242955846309244801936781
e= 298
c= 10877632494141323696706173060086601121355884259234525310108848169016343902980600845727568845840567726459597646880426553618145392261925420045090468239137708021360087031552255008560875384082156512208224524860549260224921452504096206527857951948234851544501022714642000242331900097894216267245750310618972277142398661493280224445067919972096475478247725627682511339310203665988973259583184028008064493780090233902782416178723036468013857367850280612841182686428704717939119216042691392502115511334107971111076296702241411286865983278698999230209388059621673227839004725621075278503145071860800081826179749895009884713055
```

这个题目的考点是$e, \varphi(n)$不互素，就是说 $GCD(e, \varphi(n)) \neq 1$ 。
我们根据RSA的公式来推导一下
$GCD(e, \varphi(n)) = b$
则有
$e = a \cdot b$
$\varphi(n) = b \cdot c$
则$a, \varphi(n)$ 是互素的。
存在解密指数d_a，使得:
$a \cdot d_a \equiv 1 \pmod{\varphi(n)}$
上式可以写成:
$a \cdot d_a = 1 + k \cdot \varphi(n)$
根据RSA的原理，此时a, d_a 为模数n的一对加解密密钥
$c = m^e \pmod{n}$
两边同时做幂运算

$$\begin{aligned} c^{d_a} &\equiv m^{ed_a} \pmod{n} \\ &\equiv m^{abd_a} \pmod{n} \\ &\equiv (m^b)^{ad_a} \pmod{n} \\ &\equiv (m^b)^{1+k\varphi(n)} \pmod{n} \\ &\equiv ((m^b) \pmod{n}) \cdot ((m^{kb})^{\varphi(n)} \pmod{n}) \end{aligned}$$

根据欧拉定理，如果两个正整数a和n互质，则n的欧拉函数 φ(n) 可以让下面的等式成立:
$a^{\varphi(n)} \equiv 1 \pmod{n}$
将(m^{kb}) 看做a 所以上式可以变形为
$c^{d_a} \equiv m^b \pmod{n}$

所以，我们可以求出m^b 已知$b = GCD(e, \varphi(n))$ 当$m^b < n$时，对m^b 开b次方求m

解题代码如下:

```
p= 179604728901172884984546444098336951299567226578637956867234955273321950834253763455974908411986208869587996429208317419915541879829705695083899455584410844347380529355478252408255665437124675873347322572413775553300676365369297224009959989495411452768654413529438720408670663273782756277912404468420962465297
q= 173812808579656497809794049676143957530187304874982238246182308756968315404682483726756622210171103822950123259990094171772667725498121214708342931214878639920689481582825353823960555876290816568723809873410857695707787786699193126777377193406266602939057509911129169617025374466885652429558463092448019367811
e= 298
c= 108776324941413236967061730600866011213558842592345253101088481690163439029806008457275688458405677264595976468804265536181453922619254200450904682391377080213600870315522550085608753840821565122082245248605492602249214525040962065278579519482348515445010227146420002423319000978942162672457503106189722771423986614932802244450679199720964754782477256276825113393102036659889732595831840280080644937800902339027824161787230364680138573678502806128411826864287047179391192160426913925021155113341079711110762967022414112868659832786989992302093880596216732278390047256210752785031450718608000818261797498950098847130555
 
import gmpy2, libnum
phi = (p - 1) * (q - 1)
b = libnum.gcd(e, phi)
a = e // b
assert(libnum.gcd(a, phi) == 1)
d = libnum.invmod(a, phi)
m_b = pow(c,d,p*q)
```

```python
m, _ = gmpy2.iroot(m_b, b)
print(libnum.n2s(int(m)))
# b'flag{b970456b-32f7-4f29-8244-69709a9097ef}'
```

风二西_RSA19

题目：有点难了哦

```python
import gmpy2
import libnum
import random
import uuid
flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m=libnum.s2n(flag)

p=libnum.generate_prime(1024)
q=libnum.generate_prime(1024)
n=p*q
e=65537
c=pow(m*p+n,e,n)
print("n=",n)
print("c=",c)
print("e=",e)
n= 24981188020167643746074879674147956549430370314044132464039253351652835734440674909204189557556602865962429895374385855345228410925147709118740392159925942795088299679655602727568511595072696409696271371250102853677902616206682058248674337676916036346628088795691598346311821772518310067782592432491390315016938601034921878080792576848740835839533436309467949739957366439791446358471180018562594829965668319970676634275599934272794456322929434264075482862354958544593517741073415713467939034524174022554745424038830745700456334054077809280744596235294447559597633491714726093368901810800094506409706555205366712813489
c= 18621596046506896357501494490427254488179638636182636310535680038609096612801678121484736598560358324455851721295385341142207429201113106822997090878471278518783809844775558184834964034587247425952074737623038780244691628409045326000650516141599732305641793044557982746856322994617259408879266356396616639748741726355064689072113192676869563527139209384310754130692810070548558914951576972545449143198975499766952417464247791425484288188124235662918342094367901287461654602032582020287215884903753469091244847129233355340585693462406329376871222071849218639774125831290704426337294409031337314523781596602940531607723
e= 65537
```

这个题目是一道典型的n,c不互素。我们推导一下：

$c\equiv(m\cdot p+n)^e\pmod n$

根据多项式的定理，将上式变化为：

$c\equiv(m\cdot p)^e+k_1(mcdotp)^{e-1}n+\cdots+kr(m\cdot p)n^{e-1}+n^e\pmod n$可以明显看出除第一项外，后面的项都是$n$的倍数，模$n$后为零。所以化简为：

$c\equiv(m\cdot p)^e\pmod n$

将上式变化为等式

$c=m^e\cdot p^e+kn=(m^e\cdot p^{e-1}+kq)\cdot p$

c,n都是p的倍数，可以直接公约数求p

另外一种推导方法：

$$
\begin{aligned}
c &= (mp+n)^e + kn \\
&= (mp+pq)^e + kpq \\
&= (m+q)^e p^e + kpq \\
&= ((m+q)^e p^{e-1} + kq) \cdot p
\end{aligned}
$$

公约数求p，最后解密明文需要再除以p。代码如下：

```python
n= 2498118802016764374607487967414795654943037031404441324640392533516528357344406749092041895575566028659624298953743858553452284109251477091187403921599259427950882996796556027275685115950726964096962713712501028536779026162066820582486743376769160363466280887956915983463118217725183100677825924324913903150169386010349218780807925768487408358395334363094679497399573664397914463584711800185625948299656683199706766342755999342727944563229294342640754828623549585445935177410734157134679390345241740225547454240388307457004563340540778092807445962352944475595976334917147260933689018108000945064097065552053667128134 89
c= 1862159604650689635750149449042725448817963863618263631053568003860909661280167812148473659856035832445585172129538534114220742920111310682299709087847127851878380984477555818483496403458724742595207473762303878024469162840904532600065051614159973230564179304455798274685632299461725940887926635639661663974874172635506468907211319267686956352713920938431075413069281007054855891495157697254544914319897549976695241746424779142548428818812423566291834209436790128746165460203258202028721588490375346909124484712923335534058569346240632937687122207184921863977412583129070442633729440903133731452378159660294053160772 3
e= 65537
import libnum
p = libnum.gcd(n, c)
q = n // p
phi_n=(p-1)*(q-1)
d = libnum.invmod(e,phi_n)
m = pow(c, d, n)
print(libnum.n2s(m // p))
# b'flag{bdce2381-c20a-4348-94e2-b20ff798681b}'
```

风二西_RSA20

题目：e=2 参考

```python
p= 12214849142363951006035852024732600141537875696012306134020503181023551125349623365691393781979816113846766980205879443125836481504847495308850727486625637838140995195975068938417407118922723888410184656574105440676854775262484042647475019183800055089402257393087715416820713049817173556655351713296285685511 1
q= 14275557857411348268387515034537257736327711198373652070939019269101862885098255066960125537527643422975381120642565906242872177951816826362020535741467172579165592686281764845903159399807445478261998255899529689359241468769157926061939571306752269012756070682988021553280448319770624592170208788444586290831 1
n= 17437378565136796938110429615460109306513763994515441361700826128854509816577500859330683620324942811798278054720514628441058958796253590667597639564320226081313246692273260392020172668686109508486701867559622051759183126919749044186432732690010275229298964648509739981889494014237283495021762548037876030067888443161621760780099178071080783305102758292721482026470985598146884711243792995042285939445306378914554641176698591510703687482228868400552567474448925050103485709362418457756839725639751971140439438605380691812776257135683938131226117182375780319424250673794686144155555167297765901995956241688297360472752 1
c= 3136716033729239841651527855193478838856206237778382623476323002673660140993283468296727521269046606643005570111263565370046029024734759921698482194871684883853668185647585044859555880105903361901008649
e= 2
```

这个题目比较简单，$e = 2$ 由于$c << n$ 我们直接对c 开方就能得到结果。

```python
c= 3136716033729239841651527855193478838856206237778382623476323002673660140993283468296727521269046606643005570111263565370046029024734759921698482194871684883853668185647585044859555880105903361901008649
e= 2

import gmpy2, libnum
m,s = gmpy2.iroot(c,e)
print(libnum.n2s(int(m)))
# c= 31367160337292398416515278551934788388562062377783826234763230026736601409932834682967275212690466066430055701112635653700460290247347599216984821948716848838536681856475850448595555880105903361901008649
e= 2

import gmpy2, libnum
m,s = gmpy2.iroot(c,e)
print(libnum.n2s(int(m)))
```

这道题目给出的提示是 rabin攻击。这是一种e=2的特殊攻击。给出代码如下：

```python
from gmpy2 import *
import libnum
p=275127860351348928173285174381581152299
q=319576316814478949870590164193048041239
n=87924348264132406875276140514499937145050893665602592992418171647042491658461
e=2
c=45617141162985597041928941111553655146539175146765096976546144304138540198644
inv_p = invert(p, q)
inv_q = invert(q, p)
mp = pow(c, (p + 1) // 4, p)
mq = pow(c, (q + 1) // 4, q)
a = (inv_p * p * mq + inv_q * q * mp) % n
b = n - int(a)
c = (inv_p * p * mq - inv_q * q * mp) % n
d = n - int(c)

#因为rabin 加密有四种结果，全部列出。
for i in [a,b,c,d]:
    print(i)
    print(libnum.n2s(int(i))) # 作者: 风二西 https://www.bilibili.com/read/cv13467317/
```

原理我也没有学会，抄一下两个大佬的推导过程。
Rabin攻击
Rabin加密算法

Rabin算法的两个条件

1. p, q是不同的大素数，且$p \equiv q \equiv 3 \pmod 4$
2. $n = p * q$

由于$c \equiv m^2 \pmod n$ 得到：

$$\begin{cases} m^2 \equiv c \pmod p \\ m^2 \equiv c \pmod q \end{cases}$$

由于c是模p的二次剩余则有：

$$c^{\frac{p-1}{2}} \equiv (m^2)^{\frac{p-1}{2}} \pmod q \equiv m^{p-1} \pmod p \equiv 1 \pmod p$$

代入得到：

$$m^2 \equiv c \cdot c^{\frac{p-1}{2}} \pmod p \equiv c^{\frac{p+1}{2}} \pmod p$$

开方得到

$$\begin{cases} m_1 \equiv c^{\frac{p+1}{4}} \pmod p \\ m_2 \equiv p - c^{\frac{p+1}{4}} \pmod p \\ m_3 \equiv c^{\frac{q+1}{4}} \pmod q \\ m_4 \equiv q - c^{\frac{q+1}{4}} \pmod q \end{cases}$$

通过扩展欧几里得算法 找到一组y_p, y_q 使得$y_p \cdot p \ + \ y_q \cdot q = 1$
设 $a = y_q \cdot q, \ \ b = y_p \cdot p$

对上面的式子分成4组 $(m_1, m_3), (m_2, m_3), (m_1, m_4), (m_2, m_4)$ 分别使用中国剩余定理
得4个值，这其中有一个是明文数据：

$$\begin{cases} M_1 \equiv (a \cdot m_1 + b \cdot m_3) \pmod n \\ M_2 \equiv (a \cdot m_1 + b \cdot m_4) \pmod n \\ M_3 \equiv (a \cdot m_2 + b \cdot m_3) \pmod n \\ M_4 \equiv (a \cdot m_2 + b \cdot m_4) \pmod n \end{cases}$$

```python
import libnum
p=275127860351348928173285174381581152299
q=319576316814478949870590164193048041239
n=87924348264132406875276140514499937145050893665602592992418171647042491658461
e=2
c=45617141162985597041928941111553655146539175146765096976546144304138540198644

inv_p, inv_q, _ = libnum.xgcd(p, q)
# 对应的两组根+m1 -m1 +m3 -m3
m1 = pow(c, (p + 1) // 4, p)
m2 = p - m1
m3 = pow(c, (q + 1) // 4, q)
m4 = q - m3

# 应用中国剩余定理
a = q * inv_q
b = p * inv_p

M1 = (a * m1 + b * m3) % n
M2 = (a * m1 + b * m4) % n
M3 = (a * m2 + b * m3) % n
M4 = (a * m2 + b * m4) % n

for i in [M1,M2,M3,M4]:
    print(i)
    print(libnum.n2s(int(i)))
```

风二西_RSA21

题目： m高位攻击，要用sage啦

```python
import gmpy2
import libnum
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m=libnum.s2n(flag)

p=libnum.generate_prime(512)
q=libnum.generate_prime(512)
n=p*q
m1=((m>>12)<<12)
e=3
c=pow(m,e,n)
print("n=",n)
print("c=",c)
print("e=",e)
print("m1=",m1)
n= 87820675220050944276666536937178053591737658815392270191748297820779455516597258771409492985071487273959138596491450394909787320987214452112591216592868103533000899997083612075993361507842450673721035145566686926108394182525509613162304306962952268249573625177332604042538861617260231958301740214075681935343
c= 17567615026662847704454786331620768416060521733703490038323240504320122017081266100120166514560187605667730377010739485219729971005550490697251934739975176048069121237619559127207692797665390063307700872861542043521599636684832960839337575000458552797200831427154916379682200714875823128463806257998090 1
e= 3
m1= 5600639279342851869469562328978819703852191532461614748088879616452355348306858534620635600597556838 4
```

这个题要用到sage了。sage很强，但是我刚接触，不会用，脚本都是抄来的，也看不太懂。这里先贴上大佬的博客
sage中文文档 sage常用函数

这里给出了m的高440位，我们只需要推断剩余的低 72 位。记真实的m为$highM + x$，则:

$$m^3 - c = (highM + x)^3 - c = 0$$

```python
import libnum

def phase2(high_m, n, c):
    R.<x> = PolynomialRing(Zmod(n)) # 创建一个一元x多项式环
    m = high_m + x    # 创建 函数m
    M = m((m^3 - c).small_roots()[0]) # 求函数 m^3 - c == 0 时，m的值
    print(libnum.n2s(int(M)))

n= 87820675220050944276666536937178053591737658815392270191748297820779455516597258771409492985071487273959138596491450394909787320987214452112591216592868103533000899997083612075993361507842450673721035145566686926108394182525509613162304306962952268249573625177332604042538861617260231958301740214075681935343
c= 17567615026662847704454786331620768416060521733703490038323240504320122017081266100120166
```

```
6514560187605667730377010739485219729971005550490697251934739975176048069121237619559127207
6927976653900633077008728615420435215996366848329608393375750004585527972008314271549163796
822007148758231284638062579980901
e= 3
m1= 560063927934285186946956232897881970385219153246161474808887961645235534830685853462063
56005975568384

phase2(m1, n, c)
# b'flag{ca7e88b1-3d46-4a67-8727-d0e50e62ff97}'
```

风二西_RSA22

题目：p高位攻击

```
import gmpy2
import libnum
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m=libnum.s2n(flag)
p=libnum.generate_prime(512)
q=libnum.generate_prime(512)
n=p*q
e=65537
p1=((p>>128)<<128)
c=pow(m,e,n)
print("n=",n)
print("c=",c)
print("e=",e)
print("p1=",p)
n= 711759644798205901968482061074573277733154324154344536753140124243224322545154177389 4987
8162881094869716876604585411320034080384957472263356097541370946962740975900107459650767873
3539854519057597400760125635170569967866322869305555573382256417602078073357810135789832260
50609963293210018112831999179270189311
c= 1810519075209965095313047079691413669651037396580040902964697079885703073972734990174850
2043281854034802228984535317214219994808930782000992301001478330776033469798930060334281987
4844304019441508308558041085264542896282457594052497525452544054478430902888656080245590273
370891381605152553018951211121980623 10
e= 65537
p1= 794380026419085756150463406590658907454105789730121247405115072058529147160777429995188
79251146692294620768311478220294734079952087494100041042043849266 33
```

给出p的高位，缺少低128位，Coppersmith 可以解决多项式在模n的某个因数下的根。我们设$p = pHigh + x$，然后拿去求解方程

$$p = 0 \pmod{sth\ divides\ n}$$

```
import libnum

def phase3(high_p, n, c):
    R.<x> = PolynomialRing(Zmod(n), implementation='NTL')
    p = high_p + x
```

```python
    x0 = p.small_roots(X = 2^128, beta = 0.1)[0]

    P = int(p(x0))
    Q = n // P

    assert n == P*Q

    d = inverse_mod(65537, (P-1)*(Q-1))
    print(libnum.n2s(pow(c, d, n)))

n = 12784625729032789592766625203074018101354917751492952685083808825504221816847310910447532133616954262271205877651255598995305639194329607493047941212754523879402744065076183778452640602625242851184095546100200565113016690161053808950384458996881574266573992526357954507491397978278604102524731393059303476350167738237822647246425836482533150025923051544431330502522043833872580483142594571802189321599016725741260254170793393777293145010525686561904427613648184843619301241414264343057368192416551134404100386155751297424616254697041043851852081071306219462991969849123668248321130382231769250865190227630009181759219
c = 62782408615711924505647887580059895955377425016167078750608325396078823073758876178738568612582876566561756788790422803083953531798758960876153450000312824716423377479478423151821280427005640456571042661393826430299801542115339387972926355129202454375642270295647002295953722126917208461908136849869393055045615354362817030632420626621634838670700866112871743142623748651130976728617551823862023050720195286726128388098686875267654961395878528891498942922458284921839547167229541003685888183636336488516427698323731223583159185804490836937685548412761493354595554478716035204231837858803958791174102806757672279077 8
high_p = 97522826022187678545924975588711975512906538181361325096919121233043973599759518562689050415761485716705615149641768982838255403594331293651224395590747133152128042950062103156564440155088882592644046069208405360324372057140890317518802130081198060093576841538008960560391380395697098964411821716664506908672

phase3(high_p, n, c)
# b'flag{c892e01b-2072-4ba1-9ccc-a8cb750dfc1b}'
```

风二西_RSA23

题目：这个可能要用到 sage哦

```python
import gmpy2
import libnum
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m=libnum.s2n(flag)
p=libnum.generate_prime(1024)
q=libnum.generate_prime(1024)
e=3
n=p*q
c1=pow(m,e,n)
c2=pow(2022*m+2021,e,n)
print("n=",n)
print("c1=",c1)
print("c2=",c2)
n= 14878649872127265598577310327906463595445694442874089795687255281538461872507302213106609252847683579168034165663848207247237543878434015753285820194626800282560106819324725018489
```

```
799552817816357482707023208419773917552268593373266051925536823803922433826289681775091352739327068547898980980073772614794356799549887748179096814913218045652008996246897307599980509782851636741975770977621764836822844526917297449348396463498159059942283279937461265130763691271200145989215858421263513036329692599828739213845425872881926104230192768184772597911629323901226447821509143884759631327016403851417477302786822694718560058407
c1= 175676150266618239508960526375830661715292695339479889213804234972236462741647787911646459062650145646715037789068457022000633309221436502994229482255499464114071284485035672268297610232784725624147018434613265515082923894826193542573259322814615044856922994261638278977211750347399632281012125376134757
c2= 1452299739943355429620598102750144850501685988709347628843015433896415487158634337543587442222354838823916396917992859098096943617196325746698151841054343769923746178474076881188330275543501263415546830735826280866312328010914970316027841516534872949166723592426026231369229170099785732462711391063503783886347563

```

这道题目考点 Franklin-Reiter 相关消息攻击，m是下列方程的解

$$\begin{cases} x^e - c_1 = 0 \\ (2022 * x + 2021)^e - c_2 = 0 \end{cases}$$

所以$(x-m)$是这两个多项式的公因子，可以对两个多项式求gcd，可得到m。

```
n= 14878649872127265598577310327906463595445694442874089795687255281538461872507302213106609252847683579168034165663848207247237543878434015753285820194626800282560106819324725018489799552817816357482707023208419773917552268593373266051925536823803922433826289681775091352739327068547898980980073772614794356799549887748179096814913218045652008996246897307599980509782851636741975770977621764836822844526917297449348396463498159059942283279937461265130763691271200145989215858421263513036329692599828739213845425872881926104230192768184772597911629323901226447821509143884759631327016403851417477302786822694718560058407
c1= 175676150266618239508960526375830661715292695339479889213804234972236462741647787911646459062650145646715037789068457022000633309221436502994229482255499464114071284485035672268297610232784725624147018434613265515082923894826193542573259322814615044856922994261638278977211750347399632281012125376134757
c2= 1452299739943355429620598102750144850501685988709347628843015433896415487158634337543587442222354838823916396917992859098096943617196325746698151841054343769923746178474076881188330275543501263415546830735826280866312328010914970316027841516534872949166723592426026231369229170099785732462711391063503783886347563
e=3
a=2022
b=2021

import libnum
def franklinReiter(n,e,c1,c2,a,b):
    R.<X> = PolynomialRing(Zmod(n))
    f1 = X^e - c1
    f2 = (X*a+ b)^e - c2
    # coefficient 0 = -m, which is what we wanted!
    return Integer(n-(compositeModulusGCD(f1,f2)).coefficients()[0])

  # GCD is not implemented for rings over composite modulus in Sage
  # so we do our own implementation. Its the exact same as standard GCD, but with
  # the polynomials monic representation
def compositeModulusGCD(a, b):
    if(b == 0):
        return a.monic()
```

```
    else:
        return compositeModulusGCD(b, a % b)

m=franklinReiter(n,e,c1,c2,a,b)
print(m)
print(libnum.n2s(int(m)))
```

风二西_RSA24

题目：难题来了，数论推算 ,PQ的逆元

这个题目不会做，是看了风大的视频才明白的。用到了一些数论的推导，而且中间有一个很巧妙的推测。已知：

$$\begin{cases} n & = p \cdot q \\ \varphi(n) & = (p-1)\cdot(q-1) \\ p^{-1}\cdot p & \equiv 1 \pmod q \\ q^{-1}\cdot q & \equiv 1 \pmod p \end{cases}$$

我们将后两个式子先写成下面的格式：

$$\begin{cases} p^{-1}\cdot p\ -1 & = k_1 q \\ q^{-1}\cdot q\ -1 & = k_2 p \end{cases}$$

两边同时相乘，可以变化为：

$$(p^{-1}p-1)(q^{-1}q-1) = p^{-1}q^{-1}pq - p^{-1}p - q^{-1}q + 1 = k_1k_2pq$$

将$n = pq$代入上式，并移项得到

$$p^{-1}p + q^{-1}q - 1 = (p^{-1}q^{-1} - k_1k_2)n$$

由于k_1k_2都是未知的，我们设$(p^{-1}q^{-1} - k_1k_2) = k$， 将上式化简为：

$$p^{-1}p + q^{-1}q - 1 = kn$$

到这里，题目最关键的一步来了，这个k是多少呢，能不能爆破出来？ 这里是从位数来推断的。
根据题目给出的条件，我们知道p, q都是1024位的。题目给出的逆元位数$p^{-1}_{bits} = 1022, q^{-1}_{bits} = 1024$
那么 $p \cdot p^{-1}$ 是2046位， $q \cdot q^{-1}$ 是2048位。两式相加后减一，仍为2048位。
而 $n = p \cdot q$ n的位数本来就是2048， kn 是2048位。那么就推算出$k < 2$ 所以$k = 1$。
有了$k, \varphi(n)$，我们可以通过解二元一次方程的方法来求p, q

$$\begin{cases} pq = p^{-1}p + q^{-1}q - 1 \\ \varphi(n) = (p-1)(q-1) \end{cases}$$

我们用z3-solver库来解方程

```
e= 65537
phi= 289669791665701807921652963967247753045178169962130013789049409092482226217443631640606462808774147746285187939761343441847376197589441425501944464028457254935459367930986484251554764296689309380143042290911043235980420112711223528216417379959160685027196901567897469258794283470042963655647095550343013361820395959443359420742200579810890939662236405243694667077135327865913935605734300394196666535568542232217334902869437684869309903361161908227171694304499065226516857912444156615569722204110592819450300168775742957897272625424061486617469265557551381318977199403734488064656819757037414923761416473002221207641 76
c= 720145398292074537841545933602326576473741549031065757078743296389866423532323142546976378639116392401015537356833285793967458358811500970090359466006710851461179235194672218801099391897590970224027270931178916315599036922311596360019951482688270439303813604298778434700769575460359837832123520961908670114567264187749974973161834455504080742617408916975343280 0
```

```
38195372974661435728626712478453001807287092016059839073591842844494467689945527339487420875
46000272659481888813598906420475321213275955504329833382533662489816648992514563803119332339
0714418639197846253807136848080963690346391564786977660865660369588736
pinv= 3990913861889708706795491690112920070954661105864681049469132777538175123125175700965
2158182562111025345777419136131627317857034803133000960699371752937982327038307202755295559
8336430988033334559226040211239854323769161628306307774505125638062280684545072931673991950
31896286580600520567700823513451195630782
qinv= 1259792989303926581519990688924708421452688066029902518017553624578788650900875199398
1006689632888492516214067167728862793677591231105040620450278519243574602787123899574061202
0311103525540888827958654046791157003378716522058722618817211827933204346475175247869409531
11206392925407291354289458325607958449351 8



import z3
import libnum
s = z3.Solver()
p, q = z3.Ints('p q')
s.add(p*q == pinv * p + qinv * q - 1)
s.add(phi == (p-1)*(q-1))

print(s.check())
m = s.model()

p = m[p].as_long()
q = m[q].as_long()

d = libnum.invmod(e, phi)
flag = pow(c,d,p*q)
print(libnum.n2s(flag))

# b'flag{11d7f65c-a563-445d-802f-bee980b5bbaf}'
```

python中用来解方程的库还有一个 sympy 。用这个库的代码如下：

```
e= 65537
phi= 2896697916657018079216529639672477530451781699621300137890494090924822262174436316406064
6280877414774628518793976134344184737619758944142550194446402845725493545936793098648425155
4764296689309380143042290911043235980420112711223528216417379959160685027196901567897469258
7942834700429636556470955503430133618203959594433594207422005798108909396622364052436946670
7713532786591393560573430039419666653556854223221733490286943768486930990336116190822717169
4304499065226516857912444156615569722204110592819450300168775742957897272625424061486617469
265557551381318977199403734488064656819757037414923761416473002221207641 76
c= 7201453982920745378415459336023265764737415490310657570787432963898664235323231425469763
7863911639240101553735683328579396745835881150097009035946600671085146117923519467221880109
9391897590970224027270931178916315599036922311596360019951482688270439303813604298778434700
7695754603598378321235209619086701145672641877499749731618344555040807426174089169753432800
3819537297466143572862671247845300180728709201605983907359184284449446768994552733948742087
5460002726594818888135989064204753212132759555043298333825336624898166489925145638031193323
3907144186391978462538071368480809636903463915647869776608656603695887 36
pinv= 3990913861889708706795491690112920070954661105864681049469132777538175123125175700965
2158182562111025345777419136131627317857034803133000960699371752937982327038307202755295559
8336430988033334559226040211239854323769161628306307774505125638062280684545072931673991950
```

```
31896286580600520567700823513451195630782
qinv= 12597929893039265815199906889247084214526880660299025180175536245787886509008751993981006689632888492516214067167728862793677591231105040620450278519243574602787123899574061202031110352554088882795865404679115700337871652205872261881721182793320434647517524786940953111206392925407291354289458325607958449351 8
import libnum
import sympy
p, q = sympy.symbols('p q')
expr1 = pinv * p + qinv * q - 1 - p*q
expr2 = (p-1)*(q-1) - phi
r = sympy.solve([expr1,expr2],[p, q],domain=sympy.S.Integers)

p = int(r[0][0])
q = int(r[0][1])

d = libnum.invmod(e, phi)
flag = pow(c,d,p*q)
print(libnum.n2s(int(flag)))
```

另外，sage也可以用来解方程，代码如下：

```
p, q = var('p q')
r = solve([pinv * p + qinv * q - 1 == p*q, (p-1)*(q-1) == phi],[p,q])
print(r)
```

风二西_RSA25

题目：是共模，又不是共模 (共模攻击的变形题)

```
e1= 221
e2= 299
n= 2275295174529773616853737705527477811666547090906547712593588670506531592834390923903022613431765528681170682530779102893546869524893831487372831093901033203330304998428485467248041918665665189828424701004768109624645520196743649361962392281304228613376035953597439626622867784506147152229146217067501930017032067168611492879584139927265967174929502859845192471693543987630126266747764087783756995111831173921926491318724364607355049268349211334338482197506629124493385811648678273192893143164227480949140500195541202614919469992267037838017210047502489756722302332782633954755526697526355228736183846641025633258624360201870125218109554452508772508947185514678589080189896449890049072282413616153682452402384321900891607844669493347109328807309831784121735343364838290636225331018788503134917375296997419509914775813966996109240069229311307805947308080623407644357011972213093953364558394286147157130165814637410493092316128070710785493788227546047819882312581990547756084993358911572141628568734530391934835129180071295624280990617378123281885707612919541997261695035925738102398478264886205142519306102917741175524888427512263420605506492891813170455528105517101623298965390893642533347459239642526272418397871575551768178189690550568101943368015251517957829385435246635616189896739563913352737416931159538 1
c1= 1089259379322736511692570096545836418464912539666394379607353926098099764489196579945039942421872310755115855846572395798281019191606431606740256922923970611058766228300606223320760284904897739041472549199909478426879576584509027854134506069122784804934179235636186609010779484674058755051649382755076104491612074263471610095956875228420577363515701657064473288261811266760289249559134338319682534150358407476520003106390039978467136306254008385396827681246125942478602359610947665410394179608402223477751685693520168608901470394987396376317350417859728212089368194470152963547208733630814804529657025035896538757863938964737286369 78
```

```
7438792028437635517537165821402243263723304964391824021588375653615634427171788881437731162
5833162569949367074856096858774042469897229733611875690829200986478219874772214075323186921
7181012798512152254276514299744972189011171811739779818951455452894888971448377542063278900
7749889264235981756297479724075616434133570387516474603752821608489720416726277296675690330
7518956994642174111291907559634725813349461899233834828219448501415166737945826263270661137
5881636927289054102211694665257238703039780924949416397198637826561869695131679176123869601
1615833685071844848083324708272565174336420698972015040247551896080759693366526189834725701
538195207695641492616906562840203587702‌4
c2= 1623444913611030194891504403378280317888832078073024804136228940040252035920013520777794
3760130031834157459653769349644784416074505908375923587074809644462914149880308486547506012
3420097840970184733706276824699107779327930269426809903826537911640592522293558467627053309
4637263804815165007645433250385955044622748149653679723439697035184417863743467514917866972
6600889778928198863396178993733168723110759807564162708559847659193568123887346934279868402
8977952673061492319474370639867032312696974209893993745844085661037025325712464003345130349
3685173706478318420691233054554975614225463831311873987755558868032818120669945661088243325
5619371458577896477907097145406566530373627271122355178216486385390535222768060314016240225
5969282233812006972796280850233545219871869909534783019159100014111451843232957211373061429
4346179542492406880811928085016115252200676622304183535381938917081321340120438373965763583
5748876268394270380018185882452075591923060261233523855620542037849880300696552428428480222
4595940916712846360900844939206346374912066944666599066191687022350969781809803220970991254
2239870344738129341553053701481687513110031911106083998541831371324503877294172754100396976
7077460112175853658426974619722704529302630376077420225853320004188505444985005998303791976
068327646537731144344582820378421439088‌3
```

这道题目是共模攻击的形式，同一个明文分别经过$(e_1,n),(e_2,n)$加密，得到c_1,c_2其中的难点在于$gcd(e_1,e_2)\neq 1$。

我们先来回顾一下共模攻击的解题思路：
当e_1,e_2互素时，由扩展欧几里得算法，求得一组s_1,s_2 使得 $s_1e_1+s_2e_2=1$。

则有：

$$\begin{aligned} m &= m^1 \\ &= m^{s_1e_1+s_2e_2} \\ &= m^{s_1e_1}\cdot m^{s_2e_2} \\ &= (m^{e_1})^{s_1}\cdot(m^{e_2})^{s_2} \\ &\equiv c_1^{s1}\cdot c_2^{s_2} \pmod n \end{aligned}$$

本题中，因为$gcd(e_1,e_2)=13$，将两个设$e_1=13a,\ e_2=13b$，都除13后就是互素的了，
同理，可求得求得一组s_1,s_2 使得 $s_1a+s_2b=1$

$$\begin{aligned} c_1^{s1}\cdot c_2^{s_2} &\equiv (m^{e_1})^{s_1}\cdot(m^{e_2})^{s_2} \pmod n \\ &\equiv (m^{13})^{as_1}\cdot(m^{13})^{bs_2} \pmod n \\ &\equiv (m^{13})^{as_1+bs_2} \pmod n \\ &\equiv m^{13} \pmod n \end{aligned}$$

其实有一个定理叫 裴蜀定理 $s_1e_1+s_2e_2=gcd(e_1,e_2)$ 也可以直接得到这个结论。
这里还有一个坑点，就是 $c_1^{s1}\cdot c_2^{s_2}=m^{13}+kn$ 直接开13次方，开不出来，需要爆破k

```
e1= 221
e2= 299
n= 2275295174529773616853737705527477811666547090906547712593588670506531592834390923903022
6134317655286811706825307791028935468695248938314873728310939010332033303049984284854672480
4191866566518982842470100476810962464552019674364936196239228130422861337603595359743962662
2867784506147152229146217067501930017032067168611492879584139927265967174929502859845192471
6935439876301262667477640877837569951118311739219264913187243646073550492683492113343384821
```

```
9750662912449338581164867827319289314316422748094914050019554120261491946999226703783801721
0047502489756722302332782633954755526697526355228736183846641025633258624360201870125218109
5544525087725089471855146785890801898964498900490722824136161536824524023843219008916078446
6949334710932880730983178412173534336483829063622533101878850313491737529699741950991477581
3966996109240069229311307805947308080623407644357011972213093953364558394286147157130165814
6374104930923161280707107854937882275460478198823125819905477560849933589115721416285687345
3039193483512918007129562428099061737812328188570761291954199726169503592573810239847826488
6205142519306102917741175524888427512263420605506492891813170455528105517101623298965390893
6425333474592396425262724183978715755517681781896905505681019433680152515179578293854352466
356161898967395639133527374169311595381
c1= 1089259379322736511692570096545836418464912539666394379607353926098099764489196579945030
9942421872310755115855846572395798281019191606431606740256922923970611058766228300606223320
7602849048977390414725491999094784268795765845090278541345060691227848049341792356361866090
1077948467405875505164938275507610449161207426347161009595687522842057736351570165706447328
8261811266760289249559134338319682534150358407476520003106390039978467136306254008385396827
6812461259424786023596109476654103941796084022234777516856935201686089014703949873963763173
5041785972821208936819447015296354720873363081480452965702503589653875786393896473728636978
7438792028437635517537165821402243263723304964391824021588375653615634427171788881437731162
5833162569949367074856096858774042469897229733611875690829200986478219874772214075323186921
7181012798512152254276514299744972189011171811739779818951455452894888971448377542063278900
7749889264235981756297479724075616434133570387516474603752821608489720416726277296675690330
7518956994642174111291907559634725813349461899233834828219448501415166737945826263270661137
5881636927289054102211694665257238703039780924949416397198637826561869695131679176123869601
1615833685071844848083324708272565174336420698972015040247551896080759693366526189834725701
53819520769564149261690656284020358770024
c2= 1623444913611030194891504403378280317888832078073024804136228940040252035920013520777940
3760130031834157459653769349644784416074505908375923587074809644462914149880308486547506012
3420097840970184733706276824699107779327930269426809903826537911640592522293558467627053309
4637263804815165007645433250385955044622748149653679723439697035184417863743467514917866972
6600889778928198863396178993733168723110759807564162708559847659193568123887346934279868402
8977952673061492319474370639867032312696974209893993745844085661037025325712464003345130349
3685173706478318420691233054554975614225463831311873987755558868032818120669945661088243325
5619371458577896477907097145406566530373627271122355178216486385390535222768060314016240225
5969282233812006972796280850233545219871869909534783019159100014111451843232957211373061429
4346179542492406880811928085016115252200676622304183535381938917081321340120438373965763583
5748876268394270380018185882452075591923060261233523855620542037849880300696552428428480222
4595940916712846360900844939206346374912066944666599066191687022350969781809803220970991254
2239870344738129341553053701481687513110031911106083998541831371324503877294172754100396976
7077460112175853658426974619722704529302630376077420225853320004188505444985005998303791976
06832764653773114434458282037842143908833

import gmpy2, libnum

b = libnum.gcd(e1, e2)
e1 = e1 // b
e2 = e2 // b
s1, s2, _ = libnum.xgcd(e1, e2)
m = pow(c1,s1,n) if s1>0 else pow(gmpy2.invert(c1, n),-s1,n)
m *= pow(c2,s2,n) if s2>0 else pow(gmpy2.invert(c2, n),-s2,n)
for k in range(10000):
    m1 = m + kn
    flag, s = gmpy2.iroot(m1, b)
    if s:
        print(k)
        print(libnum.n2s(int(flag)))
```

```
            break
# k = 2
# b'flag{788fadf3-1608-4fe3-9857-c79ca88163d0}'
```

风二西_RSA26

题目：光靠共模攻击是不行的。

```
e1e2= 83317
n= 193614427105727459712656611799124286143359788622944995544787081549619007255712030607961048462893972422073045323142401369620041008591203508661772003897230656587627041952583323147912862488423092973480391110452661853559034005904708201838772528961665482167313713649793785075267448614416052194784105679435849093994584178807888273185975396927413848697772491573381649565162330813817294743116040828921864901730332446935516170946354306972058049695018775926423163208730842471850933762776475794806434863691459251957341811930159004827373205486969285888707122931862520134571312512094738096567775433745005920078084044070595615858755695275464976525185800454352105145464605085843206063141225208824260046092586081479036679233509525608623439785266614194579233777300389037251299203351461254190469563210007190223034040180075144717769988281547447852286934222306851084945150831050865160027422584551430484413467606865083527713817553597684864890702798920788447168486375144859798680524494684144830276720752373480011903734615354948022119386832049765667730500495478077124251949130964011657288623786111875102282224286797553070562761334975367358632044783215499584359468539736873865894978369517833994925408789526186317926250251266206080245594712931317689880775895023256513579768229336545508466150395297553268624608684994068889691840421280711
c1= 8461455935702774839606732696628583481106108739457157757237961493721249315707058365854463354773540401038228236301572933195823206925383589280380438344346918293151928169930134045632956081184945062566817678757614816611860006425866597730747864519352309046720733870943424680296477704991108084039103348714387678260925701357278152801810444616098214964231942511332731906589339642434586792884729500618636404879133808745489823990051381479316035290316511507860259556699539853817794899071305575419968794233519130191693519669424965740754005557268523536259961342893331243822271702601491251166024629032785331163334845458532041066873062181204642511730061193181806412423310871852101503714865811232852678040552266896756835104015126426669170036333066668010437674021104622132437422087899276215087590613842963706972249810831528040008304175911799215946803926073839260039708714727246670180210772138254229999545871929350538204078637835690649108982156556159082202035891312400182426109033284706424624286874595070797624804888642649414098331802113837492380725023537502746076689560748513729573164798419260068335949704460969862627638288475890309143274515210188524564546972793051029662396980537597835874860939675476734819945549433268818379519178647303476509820821756282974287073312744339124424284365074314353072957540119217061097316369179812
c2= 620484664278552134047051354633523906425675807747346030313615222632142657386671327686830346762780781887846412400102594889347283368420308222631760811633964265352600525058848871928735904035279057295999558409318833984969521795647202287545955687149129932286878710833407300733250173179609602250640675611880883164608474340397954328170406912064023332883998009290857269846287187888156728145277024309531510105331797866833685076835273931526615042292719970926967658919153638762985362453791732734631621502983351581711188066449777097203043897589205329057225446193852593056040301734809364853181753118739604843784216536562612033307359103893806510482236157475021740603255590914121641809865052126419196638531405221094438510231726208366630512008162663744010330103156199459170979721924714894281792427651932643530734043790246173905509400532261288534889712214483873969552991657537356198890952147322594953541508366871552561029095676172024539741525694906063413062943730465813047155464168544196529490081901356676689826701155342513241655234549380041635792150618405682571311160853526719882812618473055821608217492122688669131559593093840370537049894579011917334477512297137182557102626033423909278319394496823138854073015927815382203590762482110896381565689852266596980436613904791614372199019839
```

```
459027972129220041158226866818118745059
```

这道题目在25题的基础上又增加了难度，没有给出具体的e_1, e_2，只给出了$e1 \cdot e_2 = 83317$ 。
对83317进行因式分解 ，得到$83317 = 13^2 \times 17 \times 29$, 只有3个素因子，我们可以遍历所有情况

```
e1e2= 83317
n= 19361442710572745971265661179912428614335978862294499554478708154961900725571203060796104846289397242207304532314240136962004100859120350866177200389723065658762704195258332314791286248842309297348039111045266185355903400590470820183877252896166548216731371364979378507526744861441605219478410567943584909399458417880788827318597539692741384869777249157338164956516233081381729474311604082892186490173033244693551617094635430697205804969501877592642316320873084247185093376277647579480643486369145925195734181193015900482737320548696928588870712293186252013457131251209473809656777543374500592007808404407059561585875569527546497652518580045435210514546460508584320606314122520882426004609258608147903667923350952560862343978526661419457923377730038903725129920335146125419046956321000719022303404018007514471776998828154744785228693422230685108494515083105086516002742258455143048441346760686508352771381755359768486489070279892078844716848637514485979868052449468414483027672075237348001190373461535494802211938683204976566773050049547807712425194913096401165728862378611187510228222428679755307056276133497536735863204478321549958435946853973687386589497836951783399492540878952618631792625025126620608024559471293131768988077589502325651357976822933654550846615039529755326862460868499406888969184042128071
c1= 846145593570277483960673269662858348110610873945715775723796149372124931570705836585446335477354040103822823630157293319582320692538358928038043834434691829315192816993013404563295608118494506256681767875761481661186000642586659773074786451935230904672073387094342468029647770499110808403910334871438767826092570135727815280181044461609821496423194251133273190658933964243458679288472950061863640487913380874548982399005138147931603529031651150786025955669953985381779489907130557541996879423351913019169351966942496574075400555726852353625996134289333124382227170260149125116602462903278533116333484545853204106687306218120464251173006119318180641242331087185210150371486581123285267804055226689675683510401512642666917003633306666801043767402110462213243742208789927621508759061384296370697224981083152804000830417591179921594680392607383926003970871472724667018021077213825422999954587192935053820407863783569064910898215655615908220203589131240018242610903328470642462428687459507079762480488864264941409833180211383749238072502353750274607668956074851372957316479841926006833594970446096986262763828847589030914327451521018852456454697279305102966239698053759783587486093967547673481994554943326881837951917864730347650982082175628297428707331274433912442428436507431435307295754011921706109731636917981 2
c2= 620484664278552134047051354633523906425675807747346030313615222632142657386671327686830346762780781887846412400102594889347283368420308222631760811633964265352600525058848871928735904035279057295999558409318833984969521795647202287545955687149129932286878710833407300733250173179609602250640675611880883164608474340397954328170406912064023332883998009929085726984628718788815672814527702430953151010533179786683368507683527393152661504229271997092696765891915363876298536245379173273463162150298335158171118806644977709720304389758920532905722544619385259305604030173480936485318175311873960484378421653656261203330735910389380651048223615747502174060325559091412164180986505212641919663853140522109443851023172620836663051200816266374401033010315619945917097972192471489428179242765193264353073404379024617390550940053226128853488971221448387396955299165753735619889095214732259495354150836687155256102909567617202453974152569490606341306294373046581304715546416854419652949008190135667668982670115534251324165523454938004163579215061840568257131116085352671988281261847305582160821749212268866913155959309384037053704989457901191733447751229713718255710262603342390927831939449682313885407301592781538220359076248211089638156568985226659698043661390479161437219901983945902797212922004115822686681811874505 9

import libnum , gmpy2
```

```python

def common_modulus(n, c1, c2, e1, e2):
    _, s1, s2 = gmpy2.gcdext(e1, e2)
    # 若s1<0，则c1^s1==(c1^-1)^(-s1)，其中c1^-1为c1模n的逆元。
    m = pow(c1, s1, n) if s1 > 0 else pow(gmpy2.invert(c1, n), -s1, n)
    m *= pow(c2, s2, n) if s2 > 0 else pow(gmpy2.invert(c2, n), -s2, n)
    return m % n

def decrypto(m,b,n):
    for k in range(1000):
        mb = m + k*n
        flag, _ = gmpy2.iroot(mb,b)
        if _:
            print(k,flag)
            print(libnum.n2s(int(flag)))
            break

def main():
    for e1 in range(3, e1e2):
        if e1e2 % e1 == 0:
            e2 = e1e2 // e1
            b = libnum.gcd(e1,e2)
            print(b,e1,e2)
            m = common_modulus(n, c1, c2, e1, e2)
            decrypto(m,b,n)

if __name__ == '__main__':
    main()

# b'flag{6ff24d84-4310-4ff3-a5d0-d96558f21f18}'
```

这里需要注意的一点是$c_1^{s_1}c_2^{s_2} \equiv m^b \pmod n$算出来$m^b$后直接开方结果不对时，一定要再尝试$m = \sqrt[b]{m^b + kn}$ 血泪教训，我就在这里卡了很长时间。

风二西_RSA27

题目：费马小定理 参考

```python
import libnum
import gmpy2
import uuid
from Crypto.Util.number import *
flag = "flag{" + str(uuid.uuid4()) + "}"
m = libnum.s2n(flag)

e = 65537
p = getPrime(1024)
q = getPrime(1024)
n = p * q
c = pow(m, e, n)
hint = pow(2020 * p + 2021, q, n)
print(f'n={n}')
print(f'c={c}')
print(f'hint={hint}')
```

```
n=270207252611605985410773577376507757951825559988568107375715080449499287340674444416603662703927324560518074393015645526722009755823505779670014867406681938357045591293784108282665545361481516158788083279883330609241654107620160822683229364654138802366340832138347395492346317427664168767498089789349442626513076006218688549441640606421897493659679784978316980026699747444879260824122729986468510476381831269457840609570753937375375706450866724735712810537988916856465618285884480400733733634545844687538605298497490930814341443606615668158866306999332322630794135620694764218021927356933860298627250504692098457103 83
c=101888073853876177081905754739055029941516771480798208738869805715550519007018102082183511387213064166006883137030845808081836342012315991341235494489624433765145604891308606943639019335976763735555995556472321287179935711858228948187041436753186905772213306185337395921655643967299379836593372328224425555042626946751997517306644501205697278358509965664361295437309320409893652334247910938439411540030529503063599948919553366076900652133048727382806742136307366113519827053733942990976536534970177560362115501256070931092167294830903917291340622369082825571495758122201428728558369325904595120286180762643322355188 29
hint=151799721459757332854193818142355280112889914234841216535438451569131215133205048796476660672984157512342648974353388989330737134200241762762211643943697816767816041281491688341268555172123001588647978001213360421947519652684930103272025984465727644753438946131520626094366997151939144795721138002125259651401060158386369149797356186067682076516975483644408064257708711334394168761576869858369392555987479733398661258643039829568138462875091910288297389264049926194592429047290158237305535265725753726685596691245996146133917970153936411711772821294975037523708000886340179722085358998707046122744730420646750335931 48
```

这道题目的考点是公约数分解n。特点是题目给出一些条件，通过数论的变形和推导，
将已知的条件变为p的倍数，然后$p = GCD(kp, n)$来分解n。
这里将hint 设为h,下面是推导过程：

$$
\begin{aligned}
h &\equiv (2020 \cdot p + 2021)^q \pmod n \\
&= (2020 \cdot p + 2021)^q + kn \\
&= (2020 \cdot p)^q + (2020 \cdot p)^{q-1} \cdot 2021 + \cdots + 2021^q + kpq \\
&\equiv 2021^q \pmod p
\end{aligned}
$$

但是由于q是未知的，根据费马定理，当p为素数时， $a^p \equiv a \pmod p$，

$$
\begin{aligned}
h &\equiv 2021^q \pmod p \\
&\equiv (2021^q)^p \pmod p \\
&\equiv 2021^{p \cdot q} \pmod p \\
&\equiv 2021^n \pmod p \\
&= 2021^n + kp
\end{aligned}
$$

所以，$p = GCD(2021^n - h, n)$求公约数，

解题脚本如下：

```
n=270207252611605985410773577376507757951825559988568107375715080449499287340674444416603662703927324560518074393015645526722009755823505779670014867406681938357045591293784108282665545361481516158788083279883330609241654107620160822683229364654138802366340832138347395492346317427664168767498089789349442626513076006218688549441640606421897493659679784978316980026699747444879260824122729986468510476381831269457840609570753937375375706450866724735712810537988916856465618285884480400733733634545844687538605298497490930814341443606615668158866306999332322630794135620694764218021927356933860298627250504692098457103 83
c=101888073853876177081905754739055029941516771480798208738869805715550519007018102082183511387213064166006883137030845808081836342012315991341235494489624433765145604891308606943639019335976763735555995556472321287179935711858228948187041436753186905772213306185337395921655643967299379836593372328224425555042626946751997517306644501205697278358509965664361295437309320409893652334247910938439411540030529503063599948919553366076900652133048727382806742136307366113519827053733942990976536534970177560362115501256070931092167294830903917291340 6
```

```
2236908282557149575812220142872855836932590459512028618076264332235518829
hint=15179972145975733285419381814235528011288991423484121653543845156913121513320504879647666067298415751234264897435338898933073713420024176276221164394369781676781604128149168834126855517212300158864797800121336042194751965268493010327202598446572764475343894613152062609436699715193914479572113800212525965140106015838636914979735618606768207651697548364440806425770871133439416876157686985836939255598747973339866125864303982956813846287509191028829738926404992619459242904729015823730553526572575372668559669124599614613391797015393641171177282129497503752370800088634017972208535899870704612274473042064675033593148
e = 65537
# 公约数求P
p = libnum.gcd(hint-pow(2021,n,n), n)
phi = (p-1)*(n//p -1)
d = libnum.invmod(e,phi)
print(libnum.n2s(pow(c,d,n)))
# b'flag{6b2bfc19-4ffe-473d-895c-bb2fa7278110}'
```

这种题目还是比较多的，通常要对给出的条件进行仔细的思考，用已知条件求 kp

风二西_RSA28

题目：怎么求公约数 参考学习

```
import gmpy2
import libnum
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m=libnum.s2n(flag)
p=libnum.generate_prime(512)
q=libnum.generate_prime(512)
e=65537
n=p*q
h=20211102
hc=pow(h+p*1111,e,n)
c=pow(m,e,n)
print("hc=",hc)
print("n=",n)
print("c=",c)
hc= 71505320953946158049530109094654497075489963071106175336722892393493112481336409391299522595724154571954223093317880494307263262649833222750675105885636892419350501821324979706283867758665536771783209816719106279467902518895579024290387800216711663670572861182058425925280993190282267615052256942516011995207
n= 768565111924278526459630410430727911487034226651296630507124927007604892477887438181995890720697589345702188049364792673192880934361115480559229168987827643332469463268236538773576951791651388638436573287642652045471470920744998322211389971310112227229173384467558283211420675016811320764610063323866424473 7
c= 392461793871251922715546203139663117360323480781831217070129592043679080704727645069842358271792067188381725868110666820349079679437228412577659222836925264226539165065778106294308539634480577015742099129366603964868473655797971477234373781228800754931718921910491052370058017876495870808406006705854092930 46
```

跟27题的思路差不多，我们来推导求公约数的等式

$$\begin{aligned} hc &\equiv (h+ap)^e \pmod n \\ &= (h+ap)^e + kn \\ &= h^e + k_1 h^{e-1}\cdot(ap) + \cdots + (ap)^e + kpq \\ &= h^e + kp \end{aligned}$$

由于题目中给出了h，所以，$p = GCD(hc - h^e, n)$

```python
hc= 7150532095394615804953010909465449707548996307110617533672289239349311248133640939129952259572415457195422309331788049430726326264983322275067510588563689241935050182132497970628386775866553677178320981671910627946790251889557902429038780021671166367057286118205842592528099319028226761505225694251601199 5207
n= 7685651119242785264596304104307279114870342266512966305071249270076048924778874381819958907206975893457021880493647926731928809343611154805592291689878276433324694632682365387735769517916513886384365732876426520454714709207449983222113899713101122272291733844467558283211420675016811320764610063323866424 4737
c= 3924617938712519227155462031396631173603234807818312170701295920436790807047276450698423582717920671883817258681106668203490796794372284125776592228369252642265391650657781062943085396344805770157420991293666039648684736557979714772343737812288007549317189219104910523700580178764958708084060067058540929 3046
e=65537
h=20211102

import libnum
p = libnum.gcd(hc-pow(h,e,n),n)
phi = (p-1)*(n // p -1)
d = libnum.invmod(e,phi)
print(libnum.n2s(pow(c,d,n)))
# b'flag{45faf677-a3ed-4200-a0a8-ee1f239c140d}'
```

风二西_RSA29

题目：数论，又见数论 参考

```python
import gmpy2
import libnum
import uuid

flag="flag{"+str(uuid.uuid4())+"}"
print(flag)
m=libnum.s2n(flag)
p=libnum.generate_prime(512)
q=libnum.generate_prime(512)
e=65537
n=p*q

h1=pow(2022*p+2021*q,1919,n)
h2=pow(2021*p+2022*q,9191,n)

c=pow(m,e,n)
print("h1=",h1)
print("h2=",h2)
print("n=",n)
print("c=",c)
h1= 3085582262796298958522904886463567232054467209078529715572342346678604636305077016691 13
```

```
376420230737269389407208113351501583566179358674249136579529163273304942971258270292123269525610520304081548562794446989762626091606446538341770661351624504578786119786484459801312165629288249645748360616947564661546672O5
h2= 40100423593623305059775303455521238466361560139512541341649592368069344035986841719639287569549223369845132085965748305686682111656643181380183441717688410643280141958261131108758470255679260104010792458818255865919591927360182698571973058572267041626051012344432873060584028954870019976713790755601324558548
n= 64102959876468100680156640535847855388761634133282097987245513821195616433464232166471238446539383399142190819132167640251487788433828354971655930602252481995598958979413328369264306739790569021167918377152867054737871100808301104788028284764159363852402951908183073134132550874656189587590198702783318894869
c= 45131183832310284041286970164837452402860781494367814170537748979786683176908409834474718536824887130743650179867181711815561375866637642188028690304179190358058486755191379316599103162356440279017384835373685350107932236214472686050587719705355548868691431799158695967203810074232266157701183923093912519832
```

已知

$$\begin{cases} h_1 \equiv (2022p+2021q)^{1919} \pmod n \\ h_2 \equiv (2021p+2022q)^{9191} \pmod n \end{cases}$$

因为上式有两个因子，我们需要通过配平消元的办法，先将两式的次方配齐

$$\begin{cases} h_1^{9191} \equiv (2022p+2021q)^{1919\times 9191} \pmod n \\ h_2^{1919} \equiv (2021p+2022q)^{9191\times 1919} \pmod n \end{cases}$$

我们为了写起来和看起来都简洁一些，我们设 $a = 1919 \times 9191$
将上式分解因式，除了两个最高次项，其余项均为n的倍数，模n后为0，可以直接省掉。

$$\begin{cases} h_1^{9191} \equiv (2022p)^a + (2021q)^a \pmod n \\ h_2^{1919} \equiv (2021p)^a + (2022q)^a \pmod n \end{cases}$$

接下来，我们将因子p的系数配齐

$$\begin{cases} 2021^a \times h_1^{9191} \equiv (2021\times 2022p)^a + (2021\times 2021q)^a \pmod n \\ 2022^a \times h_2^{1919} \equiv (2021\times 2022p)^a + (2022\times 2022q)^a \pmod n \end{cases}$$

至此，可以通过两式相减消掉p

$$2022^a \times h_2^{1919} - 2021^a \times h_1^{9191} \equiv (2022^{2a} - 2021^{2a})q^a$$

其中等式右边中p的倍数，等式左边，都是已知的，我们可以通过公约数求p

$$q = GCD((2022^a \times h_2^{1919} - 2021^a \times h_1^{9191}), n)$$

最终解题代码如下：

```
h1= 3085582262796298958522904886463567232054467209078529715572342346678604636305077016691133764202307372693894072081133515015835661793586742491365795291632733049429712582702921232695256105203040815485627944469897626260916064465383417706613516245045787861197864844598013121656292882496457483606169475646615466720 5
h2= 40100423593623305059775303455521238466361560139512541341649592368069344035986841719639287569549223369845132085965748305686682111656643181380183441717688410643280141958261131108758470255679260104010792458818255865919591927360182698571973058572267041626051012344432873060584028954870019976713790755601324558548
n= 64102959876468100680156640535847855388761634133282097987245513821195616433464232166471238446539383399142190819132167640251487788433828354971655930602252481995598958979413328369264
```

```
306739790569021167918377152867054737871100808301104788028284764159363852402951908183073134132550874656189587590198702783318894869
c= 45131183832310284041286970164837452402860781494367814170537748979786683176908409834474718536824887130743650179867181711815561375866637642188028690304179190358058486755191379316599103162356440279017384835373685350107932236214472686050587719705355548868691431799158695967203810074232266157701183923093912519832
e=65537
import libnum

kq = pow(2022,1919*9191,n)*pow(h2,1919,n) - pow(2021,1919*9191,n)*pow(h1,9191,n)
q = libnum.gcd(kq,n)
phi = (q-1)*(n//q - 1)
d = libnum.invmod(e,phi)
print(libnum.n2s(pow(c,d,n)))
# b'flag{af2371d6-ec8f-4f6f-924f-1c79e589b74d}'
```

中间的推导过程并不难想到，大胆去配平消元就好。

风二西_RSA30

题目： 简单解方程

```
import gmpy2
import libnum
import uuid

flag = "flag{" + str(uuid.uuid4()) + "}"
print(flag)
m = libnum.s2n(flag)
p = libnum.generate_prime(512)
q = libnum.generate_prime(512)
e = 65537
n = p * q
c = pow(m, e, n)
print("n=", n)
print("c=", c)
print("p+q:",p+q)
n= 96062491922080597831076518829216179918322824065648971872102662775708777389887258150456155926941199903319698900582336453690573807802679073691482797405893634581015816005853579232347512891669641228487952407080600046410083092215377365583621813092025715603671457491536970441472196963166133282824110392206475350597
c= 95857187009859870259610943591376704955772085571949125404184273414950492142547967617438560419529667029365675923587785478389531642998532642316223736947740729506296730470892468184491112789008312202904669826080520947868869318569542681551593258308795840060344552496526770905550765856978520074236388622120131381301
p+q: 20195887423550203161018939851084462918495401395674541827243191714303569974797909610430557278214309022383954568418589711220111848670523732316137094883861338

```

题目给出了$p \cdot q, p + q$ 主解二元一次方程，我们用z3来解一下。

```
n= 96062491922080597831076518829216179918322824065648971872102662775708777389887258150456155926941199903319698900582336453690573807802679073691482797405893634581015816005853579232347
```

```
51289166964122848795240708060004641008309221537736558362181309202571560367145749153697044147219696316613328282411039220647535059 7
c= 958571870098598702596109435913767049557720855719491254041842734149504921425479676174385604195296670293656759235877854783895316429985326423162237369477407295062967304708924681844911127890083122029046698260805209478688693185695426815515932583087958400603445524965267709055507658569785200742363886221201313813 01
p_add_q= 2019588742355020316101893985108446291849540139567454182724319171430356997479790961043055727821430902238395456841858971122011184867052373231613709488386133 8
e = 65537

import z3
import libnum
s = z3.Solver()
p, q = z3.Ints('p q')
s.add(p*q == n)
s.add(p+q == p_add_q)
print(s.check())
m = s.model()
p = m[p].as_long()
q = m[q].as_long()

phi = (p-1)*(q-1)
d = libnum.invmod(e,phi)
print(libnum.n2s(pow(c,d,n)))
```

其实这个是目不用解方程也可以做，我们分解n求p,q的目的是为了求$\varphi(n)$

$$\varphi(n)=(p-1)\cdot(q-1)=p\cdot q-p-q+1=n-(p+q)+1$$

所为直接求$\varphi(n)$ 就好

```
n= 96062491922080597831076518829216179918322824065648971872102662775708777389887258150456155926941199903319698900582336453690573807802679073691482797405893634581015816005853579232347512891669641228487952407080600046410083092215377365583621813092025715603671457491536970441472196963166133282824110392206475350597
c= 95857187009859870259610943591376704955772085571949125404184273414950492142547967617438560419529667029365675923587785478389531642998532642316223736947740729506296730470892468184491112789008312202904669826080520947868869318569542681551593258308795840060344552496526770905550765856978520074236388622120131381301
p_add_q = 20195887423550203161018939851084462918495401395674541827243191714303569974797909610430557278214309022383954568418589711220111848670523732316137094883861338
e = 65537

import libnum
phi = n - p_add_q + 1
d = libnum.invmod(e,phi)
print(libnum.n2s(pow(c,d,n)))

```

风二西_RSA31

题目：当你解出了明文，还得仔细看！

```
#注意观察明文
n= 27152096646604174584541110941581410451706936234950433944454829478126765067854634425761491672022598028471340676614489859162382599253554902887677076563729604289758463105669247743311906659558339021244234498063397569289071162675892746393503215060911067550966329213505631293393676410927001597077840198977308763467542978561105731508748580178079605291778164546774140731872866882752423119375772724541384514710947056813631477363702334603917935903416955788565572435287712705681243770248555782575194036387356188142823385916249657277369839794730090202325274362413080356223468403772759930347794789121677902685120543077310086015037
e= 65537
c= 21443739309978447227736451001666170124885976252173384066522567693199913833470633181001421312248259220431809057077638108492961175643785737880298724858401707850136018919695497133986915849318968489410858983588924559567391007284301871327840830753271265356448644435575748270707708411862922209056079148640459280609327772569776896261454282346619281434271559204413500698227235523589671199746420143669211017617342511933595379138778353672461721473899834348353845812953011585780791078305526237431422069349754889549870442138813464931326899843790536081090030164534687111986162321589476564155630639474375975686543488045729544622254
```

这个题目可以直接用yafu分解n, 唯一的考点是求出的M是一串十进制串，有经验的CTF选手，直接转就行了。

```
n= 27152096646604174584541110941581410451706936234950433944454829478126765067854634425761491672022598028471340676614489859162382599253554902887677076563729604289758463105669247743311906659558339021244234498063397569289071162675892746393503215060911067550966329213505631293393676410927001597077840198977308763467542978561105731508748580178079605291778164546774140731872866882752423119375772724541384514710947056813631477363702334603917935903416955788565572435287712705681243770248555782575194036387356188142823385916249657277369839794730090202325274362413080356223468403772759930347794789121677902685120543077310086015037
e= 65537
c= 21443739309978447227736451001666170124885976252173384066522567693199913833470633181001421312248259220431809057077638108492961175643785737880298724858401707850136018919695497133986915849318968489410858983588924559567391007284301871327840830753271265356448644435575748270707708411862922209056079148640459280609327772569776896261454282346619281434271559204413500698227235523589671199746420143669211017617342511933595379138778353672461721473899834348353845812953011585780791078305526237431422069349754889549870442138813464931326899843790536081090030164534687111986162321589476564155630639474375975686543488045729544622254

# yafu分解n
p= 164778932654038558026261134999073961465358808246217916967962415917932091397475639188179693609940047908381244899421763740304296371731318237936200964004573947387825028650437441779273221595695746575146054960578600741938064796273254573537044302774086650825405412053939974593899948205949918654575790188229081547583
q =  164778932654038558026261134999073961465358808246217916967962415917932091397475639188179693609940047908381244899421763740304296371731318237936200964004573947387825028650437441779273221595695746575146054960578600741938064796273254573537044302774086650825405412053939974593899948205949918654575790188229081547139

import libnum
def decipher_rsa(p,q,n,e,c):
    """RSA 解密函数
    """
    phi = (p-1)*(q-1)
    assert(libnum.gcd(e,phi) == 1)
    d = libnum.invmod(e,phi)
    m = pow(c, d, n)
    print(m)
```

```
    print(libnum.n2s(m))
    return m
m = decipher_rsa(p,q,n,e,c)
# 1021089710312398519754100100975045545357574552499953455657564845495249995497975099559951125
i = 0
s = str(m)
flag = ""
while i < len(s):
    if int(s[i:i+3]) < 128:
        flag += chr(int(s[i:i+3]))
        i += 3
    else:
        flag += chr(int(s[i:i+2]))
        i += 2
print(flag)

# flag{b3a6dda2-6599-41c5-8980-141c6aa2c7c3}
```

风二西_RSA32

题目：虽然没有告诉n，但是也是很简单

```
import gmpy2
import libnum
import uuid

flag = "flag{" + str(uuid.uuid4()) + "}"
print(flag)
m = libnum.s2n(flag)
p = libnum.generate_prime(1024)
q = gmpy2.next_prime(p)
e = 65537
n = p * q
phi=(p-1)*(q-1)
c=pow(m,e,n)
print("phi_n=",phi)
print("e=",e)
print("c=",c)
phi_n= 18590425039387729477598499397958448933845815900201823943032033095387848517617985895183963311626779220158042898611002867883550450494234043927249433394809188770501576117460050981691459129568811385646016788313022583930570275813634475658073903709593181423178645283642400103397763686197859087416365313566109522366803252857378015780981917721951719389946181943906426584195667576984192318817794766561272057971941403123707762513160727748633560329846219337191486777070040039921031811587821278604522949099783474985661534460618415046887625346698700392659238012087685132893594172694069190610344623932768431091964949802609612200252
e= 65537
c= 127631662760993568148141523777579643612388743709910933296235327516012792474149800770027015706265746238926440993807711315660587045513946605661018800791574743927111088056981530333398841902627062120705775512007392105489797942994921838899400825917709993336556585913600918386807883097848422018470020275120216287577893446604089907085869857043399843007119290070813582944928672904040997529818111543201394331427077570153846719652621085655649571550456574878687869562950716059482323405211558145336456429948750950680720474037092923604743301074240998705207
```

```
30003737940248614665445043210424108188954371602300714689258339214629272624
```

这道题目的考点是分解相邻素数，然后求出模数n

$\varphi(n) = (p-1)\cdot(q-1)$

则有：

$p-1 < \sqrt[2]{\varphi(n)} < q-1$

p, q 是相邻素数，可以用gmpy2直接解。

```python
phi_n= 18590425039387729477598499397958448933845815900201823943032033095387848517617985895183963311626779220158042898611002867883550450494234043927249433394809188770501576117460050981691459129568811385646016788313022583930570275813634475658073903709593181423178645283642400103397763686197859087416365313566109522366803252857378015780981917721951719389946181943906426584195667576984192318817794766561272057971941403123707762513160727748633560329846219337191486777070040039921031811587821278604522949099783474985661534460618415046887625346698700392659238012087685132893594172694069190610344623932768431091964949802609612200252
e= 65537
c= 12763166276099356814814152377757964361238874370991093329623532751601279247414980077002701570626574623892644099380771131566058704551394660566101880079157474392711108805698153033339884190262706212070577551200739210548979794299492183889940082591770999333655658591360091838680788309784842201847002027512021628757789344660408990708586985704339984300711929007081358294492867290404099752981811154320139433142707757015384671965262108565564957155045657487868786956295071605948232340521155814533645642994875095068072047403709292360474330107424099870520730003737940248614665445043210424108188954371602300714689258339214629272624
e = 65537

import gmpy2, libnum

r, _ = gmpy2.iroot(phi_n,2)
q = gmpy2.next_prime(r)
p = phi_n // (q-1) + 1
d = libnum.invmod(e,phi_n)
print(libnum.n2s(int(pow(c,d,p*q))))

```

风二西_RSA33

题目：离散对数

```python
import gmpy2
import libnum
import uuid
import random
flag = "flag{" + str(uuid.uuid4()) + "}"
print(flag)
e= libnum.s2n(flag)
n=2**512
m = random.randint(2, n-1) | 1
c=pow(m,e,n)
print("m=",m)
print("c=",c)
```

```
m= 159288671018830898156155351952186977642819454341811868237859000974938396086998292311538087829037519284750454510893392746402516320589181991767853498381730 9
c= 127148035320289412438096069744379878623265212620047268349994947026897472539252109086272152965166318993376579500721899030345136877918412444873302145547849 73
```

这个题目里，e是flag, m是一个随机数， $c \equiv m^e \pmod{n}$ 。要求的是e。这个题我不会估做，
后来在网上查询离散对数相关的知识，发现sage能解，就直接用sage解就行了。

```
#通用的求离散对数的方法
x=discrete_log(a,base,ord,operation)

#求离散对数的Pollard-Rho算法
x=discrete_log_rho(a,base,ord,operation)

#求离散对数的Pollard-kangaroo算法(也称为lambda算法)
x=discrete_log_lambda(a,base,bounds,operation)

#小步大步法
x=bsgs(base,a,bounds,operation)
```

本题解题代码

```
n = 2**512
m= 1592886710188308981561553519521869776428194543418118682378590009749383960869982923115380878290375192847504545108933927464025163205891819917678534983817309
c= 1271480353202894124380960697443798786232652126200472683499949470268974725392521090862721529651663189933765795007218990303451368779184124448733021455478497 3
x=discrete_log(c,mod(m,n))
print(x)
print(libnum.n2s(x))
# flag{08c1288c-138f-48b8-93b0-914effbebf39}
```

风二西_RSA34

题目：二项式与公约数

```
from Crypto.Util.number import *
import libnum
import os
import uuid

flag = "flag{" + str(uuid.uuid4()) + "}"
print(flag)
p = getPrime(1024)
q = getPrime(1024)
n = p*q
g = n+1
m = bytes_to_long(flag.encode()+os.urandom(80))
assert m < n
c=(pow(g,p,n*n)*pow(m,n,n*n))%(n*n)
print("c="+str(c))
print("n="+str(n))
```

```
print("hint="+str(pow(m,n,n*n)))
c=90517278441887371373270055898918199632588864841951444859752221268427839849296895366289701
0162459720878338826781745670048648503779257335410151406677417672179943629405238436257761900
8747797513236856454871711189417252476701775088356304816913860000182982796031718538697794846
6007953921262011282836107888467567231820714066294280789197833321732390361410950804728346481
5109670669749578130718034005367412888820840187084195735110169860263051517794259176586440054
6034894197132602494822518768208868958129657539806294090033689807268735476205866730844345953
4402252635055443568328697047638514813723303232832606946060964799074828196565910982335906346
4740041165085947578112934084444546578892243586755065620072226406559570695996411425392411720
2355779240876861448182227545041804590123720663354906825137092489741459510328558644238989242
9017893044142451943551306922376161362494671438674407472487396331603117112936866113003452728
9076196976788125801482545219357263063182186736412009825867346742491995907489707140750618968
0843034091404592590299410066318122405353009166089642672869003032638896103223806983822935837
5497020376870439969697409432742647037016736463549070519091640911597954142054204331427390334
155169385426645784602445641703465715235570242558548 4
n=31016237680452528704873191037134067101278463443982527817436364102617348902532256530861676
4482010757468278345429565063666988254234224838931893523292566714878740821560522146419459470
9660631338030673778984877923736484532335137711567424894776691967643358467729995017656498696
1205465375287101064714101688614722710223640975579844338149091130387005331087496684778954404
0656143424280136970818935857220861272398643867943516345975732298191791164301050004228836641
3101694950744953279149437240763657647274817336710078246887422192026763119848691345169262140
482628027433903144922410891151160136938596244964790606500413289011031519 1
hint=2182073957684785574711997671940940438767245921385887745709089904684289506051492853797 9
2082880818082732758456157750106025376469170078549528685829802378925626144504311460262428580
9204056723416739774684060041671087954444199945685286648981537791696192424405086817591860171
5371152766128316671076833144110650655415140708975887924665712105923415725508396497303762849
1442405875716725032650227392405274334487258752469294042631778616110618699764229701945343859
2398887933056209604804635382576440204875173613710988830025265399024597544335711447174912921
0596714186114182204244369362909046244871250053260733692850201708915425065397987864614326456
8658353591859972526675416599548031535473192600081383998056552022089590117980379324695272326
1909354953379672400497763071278898311658556957822939088542733819157608322674265325541579823
1040166410655221706264214761181286352676308737415721776432306584167227253962281247253519243
9927792375314310035581298527907756661627438862033009575874759150646763139825124510422557834
8290030905933267811961441758440937090719450194026868452595822668240335669432274613545614807
2132060336336381842338258240685724676482265328998254170842306963639727489954516116217493089
70083168382393710943149972676235785170284313148452452 39
```

这个题目也是公约数分解n的思路，不过推导起来还是有难度的，我也是看了风大的方法，才学会。

已知：

$$
\begin{aligned}
h &\equiv m^n \pmod{n^2} \\
c &\equiv g^p \cdot m^n \pmod{n^2}
\end{aligned}
$$

将$g = (n+1), h$代进去。2式变为：

$$c \equiv (n+1)^p \cdot h \quad \pmod{n^2}$$

两边同时乘上h模n^2的逆元 h^{-1}

$$c \cdot h^{-1} \equiv (n+1)^p \quad \pmod{n^2}$$

这里我们需要用到二项式定理。

$$(a+b)^n = \sum_{r=0}^{n} C_n^r a^{n-r} b^r = C_n^0 a^n + C_n^1 a^{n-1} b + \cdots + C_n^r a^{n-r} b^r + \cdots + C_n^n b^n$$

使用二项式定理分解$(n+1)^p$

$$(n+1)^p = n^p + C_p^1 n^{p-1} + C_p^2 n^{p-2} + \cdots + C_p^{p-1} n + 1$$

其中二项式的系数

$$C_p^r = \frac{p!}{r!(p-r)!} = \frac{p \cdot (p-1)!}{r!(p-r)!}$$

我们来重点看一下中间的项的系数，除了第一项和最后一项，其余的分子都是有一个p，
已知p是素数，p 与小于p的数都互质。那么，所有分子上的p都无法被约分。
又知道C_p^r 是整数，那么可以将$C_p^r = k_r p$

$$\begin{aligned}(n+1)^p &= 1 + n^p + k_1 p n^{p-1} + k_2 p n^{p-2} + \cdots + k_{p-1} p n \\ &= 1 + pq \cdot n^{p-1} + k_1 p n^{p-1} + k_2 p n^{p-2} + \cdots + k_{p-1} p n \\ &= 1 + kpn\end{aligned}$$

到了这里，基本就完成，将结果代入：

$$c \cdot h^{-1} = 1 + k_1 pn + k_2 n^2$$

将1移到左边

$$c \cdot h^{-1} - 1 = k_1 pn + k_2 pqn = pn(k_1 + k2_q) = kpn$$

最终得到

$$\frac{c \cdot h^{-1}}{n} = kp$$

至此，我们终于找到了kp 而等式左边都是已知，可以通过公约数分解n
另外还有一个点，我们的加指数是 n , 模数是n^2 $\varphi(n^2) = p(p-1)q(q-1) = n(p-1)(q-1)$
加密指数与$\varphi(n^2)$不互素，不能直接解，需要再变形一下。

$$h = m^n + kn^2$$

两边同时模n

$$h \equiv m^n \pmod{n}$$

最终的解题代码如下：

```
c=90517278441887371373270055898918199632588864841951444859752221268427839849296895366289701
0162459720878338826781745670048648503779257335410151406677417672179943629405238436257761900
8747797513236856454871711189417252476701775088356304816913860000182982796031718538697794846
6007953921262011282836107888467567231820714066294280789197833321732390361410950804728346481
5109670669749578130718034005367412888820840187084195735110169860263051517794259176586440054
6034894197132602494822518768208868958129657539806294090033689807268735476205866730844345953
4402252635055443568328697047638514813723303232832606946060964799074828196565910982335906346
4740041165085947578112934084444546578892243586755065620072226406559570695996411425392411720
2355779240876861448182227545041804590123720663354906825137092489741459510328558644238989242
9017893044142451943551306922376161362494671438674407472487396331603117112936866113003452728
9076196976788125801482545219357263063182186736412009825867346742491995907489707140750618968
0843034091404592590299410066318122405353009166089642672869003032638896103223806983822935837
5497020376870439969697409432742647037016736463549070519091640911597954142054204331427390334
1551693854266457846024456417034657152355702425585484
n=31016237680452528704873191037134067101278463443982527817436364102617348902532256530861676
4482010757468278345429565063666988254234224838931893523292566714878740821560522146419459470
9660631338030673778984877923736484532335137711567424894776691967643358467729995017656498696
1205465375287101064714101688614722710223640975579844338149091130387005331087496684778954404
```

```
065614342428013697081893585722086127239864386794351634597573229819179116430105000422883664
1310169495074495327914943724076365764727481733671007824688742219202676311984869134516926214
048262802743390314492241089115116013693859624496479060650041328901103151 91
hint=21820739576847855747119976719409404387672459213858877457090899046842895060514928537979
2082880818082732758456157750106025376469170078549528685829802378925626144504311460262428580
9204056723416739774684060041671087954444199945685286648981537791696192424405086817591860171
5371152766128316671076833144110650655415140708975887924665712105923415725508396497303762849
1442405875716725032650227392405274334487258752469294042631778616110618699764229701945343859
2398887933056209604804635382576440204875173613710988830025265399024597544335711447174912921
0596714186114182204244369362909046244871250053260733692850201708915425065397987864614326456
8658353591859972526675416599548031535473192600081383998056552022089590117980379324695272326
1909354953379672400497763071278898311658556957822939088542733819157608322674265325541579823
1040166410655221706264214761181286352676308737415721776432306584167227253962281247253519243
9927792375314310035581298527907756661627438862033009575874759150646763139825124510422557834
8290030905933267811961441758440937090719450194026868452595822668240335669432274613545614807
2132060336336381842338258240685724676482265328998254170842306963639727489954516116217493089
700831683823937109431499726762357851702843131484524 5239
import libnum
h1 = libnum.invmod(hint,n*n)
p = libnum.gcd((c*h1-1)//n, n)
q = n // p
phi_n = (p-1)*(q-1)

d = libnum.invmod(n,phi_n)
m = pow(hint%n,d,n)
print(libnum.n2s(m))
# b"flag{3fc79e66-f746-43ed-8d43-12d62083bb28}
```

二项式定理在RSA的推导中很常见，我们要记住这个推导的结论 $(a+1)^p = a^p + kpa + 1$
其实我自己在做题的时候，跟上面的思路不一致，我是直接用费马小定理来处理的

$$\begin{aligned}
c \cdot h^{-1} &= (n+1)^p + kn^2 \\
&\equiv (n+1)^p \pmod{p} \\
&\equiv n+1 \pmod{p} \\
&= 1 + kp
\end{aligned}$$

在求出来 $GCD(c \cdot h^{-1} - 1, n)$ 值为n。
所以再尝试 $GCD(\frac{c \cdot h^{-1} - 1}{n}, n)$ 得到了p，歪打正着吧。

风二西_RSA35

题目：C怎么算?

```
import gmpy2
import libnum
import uuid

flag = "flag{" + str(uuid.uuid4()) + "}"
print(flag)
m = libnum.s2n(flag)
p = libnum.generate_prime(512)
q = gmpy2.next_prime(p)
n = p * q
e = 65537
c = pow(m, e, n)
```

```python
c1=c%p
c2=c%q
print("n=", n)
print("e=", e)
print("c1=", c1)
print("c2=", c2)
n= 9039452968991383976520530052811080314127909039986059636787094969749632482099833796198542503462050819165561599198204091355897717698394490678004224219235843616663176501720954726353467034178487217873629855104841605346081837335486947075126739413483486684190458621421778462110847197677983208994774236830631568250 7
e= 65537
c1= 54470824811861173656723297358770373327761872517730443063598664008355963090565143903804526961545418910643071499427654106146480890042584367646220840195315 05
c2= 45353819392843306646340474808065751971076055659995998200543108635298080673073694829440909750145444491554254936956497864502416960377628551546862958558139 41

```

这个题目没有给c，给出了$c_1 = c \pmod{p}$，$c_2 = c \pmod{q}$。那而且p, q还是相邻的素数，
可以直接分解n，求出p, q。再用中国剩余定理求c

```python
n= 90394529689913839765205300528110803141279090399860596367870949697496324820998337961985425034620508191655615991982040913558977176983944906780042242192358436166631765017209547263534670341784872178736298551048416053460818373354869470751267394134834866841904586214217784621108471976779832089947742368306315682507
e= 65537
c1= 5447082481186117365672329735877037332776187251773044306359866400835596309056514390380452696154541891064307149942765410614648089004258436764622084019531505
c2= 4535381939284330664634047480806575197107605565999599820054310863529808067307369482944090975014544449155425493695649786450241696037762855154686295855813941
import gmpy2, libnum
from functools import reduce
def CRT(mi, ai):
    # mi,ai分别表示模数和取模后的值,都为列表结构
    # Chinese Remainder Theorem
    assert (isinstance(mi, list) and isinstance(ai, list))
    M = reduce(lambda x, y: x * y, mi)
    ai_ti_Mi = [a * (M // m) * gmpy2.invert(M // m, m) for (m, a) in zip(mi, ai)]
    return reduce(lambda x, y: x + y, ai_ti_Mi) % M

r = gmpy2.iroot(n,2)
q = gmpy2.next_prime(r)
p = n // q
c = CRT([p,q],[c1,c2])

phi = (p-1)*(q-1)
d = libnum.invmod(e, phi)
m = pow(c, d, n)
print(libnum.n2s(m))

# b'flag{184abec1-8e0a-465d-b297-3b1257c29fdd}'
```

风二西_RSA36

题目：最小公倍数

```python
from gmpy2 import lcm, invert
import libnum
from Crypto.Util.number import *
import uuid
import gmpy2

flag = "flag{" + str(uuid.uuid4()) + "}"
print(flag)
e = 65537
p = getPrime(512)
q = getPrime(512)
n = p**4*q

c = pow(libnum.s2n(flag), e, n)
print("c=",c)

h1 = (invert(e, lcm(p - 1, q - 1))) % (p - 1)
print("h1=",h1)

b = 449703347709287328982446812318870158230369688625894307953604074502413258045265502496365998383562119915565080518077360839705004058211784369656486678307007348691991136610142919372779782779111507129101110674559235388392082113417306002050124215904803026894400155194275424834577942500150410440057660679460918645357376095613079720172148302097893734034788458122333816759162605888879531594217661921547293164281934920669935417080156833072528358511807757748554348615957977663784762124746554638152693469580761002437793837094101338408017407251986116589240523625340964025531357446706263871843489143068620501020284421781243879675292060268876353250854369189182926055204229002568224846436918153245720514450234433170717311083868591477186061896282790880850797471658321324127334704438430354844770131980049668516350774939625369909869906362174015628078258039638111064842324979997867746404806457329528690722757322373158670827203350590809390932986616805533168714686834174965211242863201076482127152571774960580915318022303418111346406295217571564155573765371519749325922145875128395909112254242027512400564855444101325427710643212690768272048881411988830011985059218048684311349415764441760364762942692722834850287985399559042457470942580456516395188637916303814055777357738894264037988945951468416861647204658893837753361851667573185920779272635885127149348845064478121843462789367112698673780005436144393573832498203659056909233757206537514290993810628872250841862059672570704733990716282248839
a = 2021*p**3 + 2022 + 2023*p**4

h2 = pow(2, a, b)
print("h2=",h2)
c= 32461937291123838150015437738232132778136420835501172981733602990740273486953741568090810560778161966164945158511260774070358644248232323420087723385263256644196533073406984717180336745536558689984942092573081904079903490501386556080018394074410235002595317034773282955985118051270196236522237314803148550104312339887381647416993827777740145028707473690705297024951376596720342536992121548930518678367840538408090222135227089379157055839665026501357827048478457911999587594147965082828271987647298414995124826623670109344083974284447663062399760045862177923687178825610208304215451278227328518587611057109711358853294506221375136820608279989589848521299683305678250602533798187554673551619882364862267158445783085217884637473353723296775978631814700691324134568859907895
h1= 646806614281203781124581143102982262071843125617265982731803098076674064687504375436269187136135479626076054485134838660805902017846974147255250842817328 9
h2= 35147827320166121137029599251390377701223298325605632623892633301137141662971616209010 4
```

```
6196742651568212232312644404431068925420642237795571905785014389467892540304028877311655408
```

这道题目是2021年西湖论剑的原题。解法很简单，跟dp泄露的方法差不多。已知：

$$h_1 = invert(e, lcm(p-1, q-1)) + k(p-1)$$

两边同时乘以e，

$$\begin{aligned} e \cdot h_1 &= e \cdot invert(e, lcm(p-1, q-1)) + k(p-1) \\ &= 1 + k_1(lcm(p-1, q-1)) + k(p-1) \end{aligned}$$

$lcm(p-1, q-1)$ 一定是能够整除 $p-1$ 的。所以可变化为：

$$e \cdot h_1 = 1 + k(p-1)$$

因为 $h_1 < p-1 \Rightarrow e > k$ 。我们可以通过爆破k来解p

解题代码如下：

```
c= 3246193729112383815001543773823213277813642083550117298173360299074027348695374156809081
```

```
5282255786648450786968075596257254398192572271596503659829888593426660292392370791690250937349603952086338579528401425115381259226152373927907631243544385590573151233057856693516335946126800186613423181380364592545178717514530790443222349215542405323488583752908361247646804134607043045490032191842955374212731374369553677685011922864865926420624453951900046392828931218124653503519327715679986527308746117949838044771452356792790424284145175044597323770578812221224613738284 7
e = 65537
import gmpy2
import libnum

for k in range(2,63337):
    tmp = (h1*e-1)
    p = tmp // k + 1
    if tmp % k ==0 and gmpy2.is_prime(p):
        print(p)
        m = pow(c,h1,p)
        print(libnum.n2s(m))

# b'flag{93776440-8fc4-429c-aca2-ddb54767d0bc}'
```

风二西_RSA37

题目：没有告诉n

```
import libnum
from Crypto.Util.number import *
import uuid
import gmpy2

flag = "flag{" + str(uuid.uuid4()) + "}"
print(flag)
m=libnum.s2n(flag)
e = 65537
p = getPrime(512)
q = gmpy2.next_prime(p)
n=p*q
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
c=pow(m,e,n)
print("c=",c)
print("e=",e)
print("d=",d)
c= 35017095079265838979734409416685141495410012751608421171984571206906907112637156478970636762948745062984603962185272983251479813367054508240145465939059405315210753494552412701642616337156914318259643364976440231342251176780836484538401507518406607672852280282429279213480081726239103353793034991628250368074
e= 65537
d= 10159486972054693132085407812207471352544021493731257009387580579544293941915068742654169629567487877929954891465578843731430912561695790903356228092971786660606577981749730231546736362340387230523928011914079016254871539712051056198870497621869815576436149752790615441331384819333802888170817008138831105084 1

```

这道题目给出了c, d, e 没有给出模数n，，所以套路和泄露dp 爆破p 是一样的。

$$e \cdot d = 1 + k\varphi(n)$$

$$d < \varphi(n) \Rightarrow e > k$$

我们爆破k，求$\varphi(n)$。
同时$\varphi(n) = (p-1)(q-1)$ 那么显然有：

$p - 1 < p < \sqrt{\varphi(n)} < q - 1 < q$
p, q是相邻素数，$\sqrt{\varphi(n)}$的下一个素数一定是q

```
c= 35017095079265838979734409416685141495410012751608421171984571206906907112637156478970636762948745062984603962185272983251479813367054508240145465939059405315210753494552412701642616337156914318259643364976440231342251176780836484538401507518406607672852280282429279213480081726239103353793034991628250368074
e= 65537
d= 101594869720546931320854078122074713525440214937312570093875805795442939419150687426541696295674878779299548914655788437314309125616957909033562280929717866606065779817497302315467363623403872305239280119140790162548715397120510561988704976218698155764361497527906154413313848193338028881708170081388311050841

import gmpy2,libnum
k_phi = e*d -1

for k in range(2,e):
    if k_phi % k == 0:
       phi = k_phi // k
       r, _ = gmpy2.iroot(phi,2)
       q = gmpy2.next_prime(r)
       if phi % (q-1) == 0:
          p = phi // (q-1) + 1
          m = pow(c,d,p*q)
          print(k)
          print(libnum.n2s(int(m)))

# b'flag{809ec7a7-7b78-4a30-80f7-e05d9aa5798f}'
```

风二西_RSA38

题目：确实是模不互素

```
e= 65537
n1= 76640333332630162663123751606067435314336340654460103173534423466657674524375020626274205515475640036025332334046453527002176045523313406356544627207141064266737007436752732620171567525475073130377747115360345241545410630734723724753756651027457666316761714427366775234473273724373416332117675121006167145404262026766633414550316157472176323
n2= 124440313565503113000305130236245224122660032520451764026576620601165065563101713415477663266674267245631322714735471550234417000764056645642165031070037322521161537165526540073654737460344673101304077315130176073622527457476522713203621742470310504203143711273625273411067051223022646617226224236404164207644473450316235021766627574031513547
c1= 35746561223300251331740725746543176021307604676022271355436612677312027341112473231206167707223243600125163327527673606771420747073156663072514253675300213137523770242670170012456726576735146623016624071313751660414571706041731612540715367514460256503435430051736012262234256065342703214571774067503205305210726131331540216432603612121112173
```

```
c2= 621750474546653732542746313530056117224745465074566706165107075634173543125341601554640053415122322270675701303574461257310407160207635744242323202060310260301302114441760033011144277626132365706411625701062740125303240101517177410173515212625302262756723462025630036411620704104474405626704547314525103236477152034522405100343203115534513
```

这道题目的难点在于进制，观察题目中给出的数字，没有数字$8,9$。
这么多的数字中没有这两个，显然是八进制，我们转换8进制后，就是普通的共模攻击，再用模不互素能够解出p,q

```
e= 65537
n1= 76640333332630162663123751606067435314336340654460103173534423466657674524375020626274205515475640036025332334046453527002176045523313406356544627207141064266737007436752732620171567525475073130377747115360345241545410630734723724753756651027457666316761714427366775234473273724373416332117675121006167145404262026766633414550316157472176323
n2= 124440313565503113000305130236245224122660032520451764026576620601165065563101713415477663266674267245631322714735471550234417000764056645642165031070037322521161537165526540073654737460344673101304077315130176073622527457476522713203621742470310504203143711273625273411067051223022646617226224236404164207644473450316235021766627574031513547
c1= 357465612233002513317407257465431760213076046760222713554366126773120273411124732312061677072232436001251633275276736067714207470731566630725142536753002131375237702426701700124567265767351466230166240713137516604145717060417316125407153675144602565034354300517360122622342560653427032145717740675032053052107261313315402164326036121211121 73
c2= 621750474546653732542746313530056117224745465074566706165107075634173543125341601554640053415122322270675701303574461257310407160207635744242323202060310260301302114441760033011144277626132365706411625701062740125303240101517177410173515212625302262756723462025630036411620704104474405626704547314525103236477152034522405100343203115534513

import libnum

n1,n2,c1,c2 = [int(i,8) for i in [n1,n2,c1,c2]]
p = libnum.gcd(n1,n2)
d1 = libnum.invmod(e,(p-1)*(n1//p - 1))
print(libnum.n2s(pow(c1,d1,n1)))
# b'flag{d32e6316-b05f-49ce-b584-cc2ef17541c0}'
```

这个题目在buu-ctf中也有一道类似的套路题rsa4，不过是5进制的。
这个套路只能说是全靠经验了，没有看出来就白瞎。

风二西_RSA39

题目：费马小定理

```
import libnum
import uuid
from Crypto.Util.number import *
import gmpy2

flag = "flag{" + str(uuid.uuid4()) + "}"
print(flag)
m=libnum.s2n(flag)


p = getPrime(512)
```

```python
q = getPrime(512)
n=p*q
hint = gmpy2.lcm(p - 1 , q - 1)
e=54722
c=pow(m,e,n)

print("n=",n)
print("e=",e)
print("c=",c)
print("hint=",hint)
n= 151017833652993344982456748947512913199036110563022758585644059291398765148383440007471971157926046710864521816236947556883446962426290165088769500162686035905917820395869193634692025452526067521211836355773131074882644525209815424268468179258864945171582638832417258825878705998609543403359555089614991139843
e= 54722
c= 1638192451785037616945368623786536722996997096038693336564874830885530504652713643854961650124406964049881328158632703635679349262162163496128186523918615830920027250061369538588536486447527147085847914668475027075775170465707957219454106036231404954907894059642988304849892426725298100425762546723405676770 9
hint= 7550891682649667249122837447375645659951805528151137929282202964569938257419172000373598557896302335543226090811847377844172348121314508254438475008134300565899828879080636324852008367580415541248346872499915556145600062380702317612639530709039521510115250116586415858055588230227970292634033248235681426206 4
```

题目给出了 $LCM(p-1,q-1)$ 这是一个最小公倍数。

关于最小公倍数有一个重要的性质，两数的乘积等于最小公倍数与最大公约数的乘积。

$$a \cdot b = LCM(a,b) \cdot GCD(a,b)$$

在这道题目中，我们设公约数为 $k = GCD(p-1,q-1)$ 则有：

$\varphi(n) = h \cdot k$

我们从题目中观察到，h的位数为1023, n的位数为1024. $\varphi(n) = (p-1)\cdot(q-1)$的位数为1024。
那么k的位数为1，所以$k = 2^1 = 2$。

```python
n= 1510178336529933449824567489475129131990361105630227585856440592913987651483834400074719711579260467108645218162369475568834469624262901650887695001626860359059178203958691936346920254525260675212118363557731310748826445252098154242684681792588649451715826388324172588258787059986095434033595550896149911398 43
e= 54722
c= 16381924517850376169453686237865367229969970960386933365648748308855305046527136438549616501244069640498813281586327036356793492621621634961281865239186158309200272500613695385885364864475271470858479146684750270757751704657079572194541060362314049549078940596429883048498924267252981004257625467234056767709
hint= 75508916826496672491228374473756456599518055281511379292822029645699382574191720003735985578963023355432260908118473778441723481213145082544384750081343005658998288790806363248520083675804155412483468724999155561456000623807023176126395307090395215101152501165864158580555882302279702926340332482356814262064

import gmpy2, libnum

phi = 2 * hint  # 推算 gcd(p-1,q-1) == 2
b = libnum.gcd(e,phi)  # e,phi 不互素
d = libnum.invmod(e//b, phi)
```

```python
m = pow(c,d,n)
flag,_ = gmpy2.iroot(m,b)
print(libnum.n2s(int(flag)))
# b'flag{f23a4c10-a31f-4ab3-8f64-800bc0862c48}'
```

这道题目刚巧k比较小，运气了。正常我们是通过一个位数差，来判断出来k的大致范围。然后遍历爆破。