buuctf_crypto

原创

reus09 于 2021-04-03 00:21:52 发布 156 收藏

分类专栏： ctf_Crypto

本文链接：https://blog.csdn.net/resu09/article/details/115410230

0x01 变异的凯撒密码

```
题目`afZ_r`:
97\102\90\95\114
明文`flag{`:
102\108\97\103\123
相差:
5\6\7\8\9
```

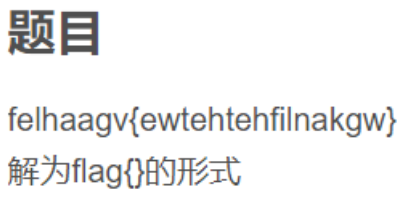
复制

脚本如下：

```
m = 'afZ_r9VYfScOeO_UL^RWUc'
s = ''
for i in range(len(m)):
    s += chr(ord(m[i])+i+5)

print(s)
```

flag{Caesar_variation}

0x02 篱笆树的影子



由题目篱笆 为 fence 猜测为栅栏密码加密

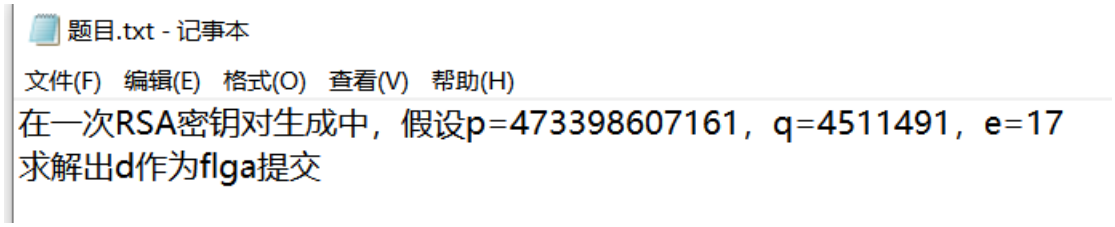
法1：

有题目得，解为flag{}的形式，所以felhaagv解密为flag的形式，所以可以看出该栅栏密码为两行
f l a g
e h a v
所以写出剩下的，解出密码为
flag{wethinkwehavetheflag}

或者使用在线网站进行解密

0x03 RSA

题目：



脚本如下：

```
p = 473398607161
q = 4511491
e = 17

pini = (p-1) *(q - 1)
d = invert(e,pini)
print(d)
```

flag{125631357777427553}

0x03 丢失的md5

首先打开之后是一个py文件，本地跑一下，出错

```
import hashlib
for i in range(32,127):
    for j in range(32,127):
        for k in range(32,127):
            m=hashlib.md5()
            m.update('TASC'+chr(i)+'O3RJMV'+chr(j)+'WDJKX'+chr(k)+'ZM')
            des=m.hexdigest()
            if 'e9032' in des and 'da' in des and '911513' in des:
                print des
```

首先补全print() 的括号

然后报错见下图

```
PS D:\课件\ctf\Crypto\python> python -u "d:\课件\ctf\Crypto\python\test.py"
Traceback (most recent call last):
  File "d:\课件\ctf\Crypto\python\test.py", line 14, in <module>
    m.update('TASC'+chr(i)+'O3RJMV'+chr(j)+'WDJKX'+chr(k)+'ZM')
TypeError: Unicode-objects must be encoded before hashing
```

报错是因为是在进行md5哈希运算前，需要对数据进行编码

正确代码：

```
import hashlib
for i in range(32,127):
    for j in range(32,127):
        for k in range(32,127):
            m=hashlib.md5()
            m.update('TASC'.encode('utf-8')+chr(i).encode('utf-8')+'O3RJMV'.encode('utf-8')+chr(j).encode('u
tf-8')+'WDJKX'.encode('utf-8')+chr(k).encode('utf-8')+'ZM'.encode('utf-8'))
            des=m.hexdigest()
            if 'e9032' in des and 'da' in des and '911513' in des:
                print (des)
```

0x04 rsarsa

题目

```
Math is cool! Use the RSA algorithm to decode the secret message, c, p, q, and e are parameters for the RSA
algorithm.



p =  96484230290105156765905517400104265349457376392357398006439893520398525072984913995610350091634270503701075707336333509116912802977771602006252816653784 83
q =  11874843837980297032092405848653656852760910154543380907650040190704283358909208578251063047732443992230647903887510065547947313543299303261986053486569407
e =  65537
c =  83208298995174604174773590298203639360540024871256126892889661345742403314929861939100492666605647316646576486526217457006376842280869728581726746401583705899941768214138742259689334840735633553053887641847651173776251820293087212885670180367406807406765923638973161375817392737747832762751690104423869019034

Use RSA to find the secret message
```

脚本如下:

```
import gmpy2
p = 96484230290105156765905517400104265349457376392357398006439893520398525072984913995610350091634270503701075707336333509116912802977771602006252816653784 83
q =  11874843837980297032092405848653656852760910154543380907650040190704283358909208578251063047732443992230647903887510065547947313543299303261986053486569407
e = 65537
c = 83208298995174604174773590298203639360540024871256126892889661345742403314929861939100492666605647316646576486526217457006376842280869728581726746401583705899941768214138742259689334840735633553053887641847651173776251820293087212885670180367406807406765923638973161375817392737747832762751690104423869019034
n = p*q
pini = (p-1) * (q-1)
d = gmpy2.invert(e,pini)

m = gmpy2.powmod(c,d,n)

print(m)
print(hex(m))
print(bytes.fromhex(hex(m)[2:]))
```

0x05 Windows 系统密码

题目：

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
ctf:1002:06af9108f2e1fecf144e2e8adef09efd:a7fcb22a88038f35a8f39d503e7f0062:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SUPPORT_388945a0:1001:aad3b435b51404eeaad3b435b51404ee:bef14eee40dffbc345eeb3f58e290d56:::
```

windows 系统密码 都是以md5的形式储存的

所以进行md5暴力破解

得到flag

0x06 中文电码

题目

6060461526236008178312161216211963 86

感觉没有提示

看大佬的wp 发现是中文电码

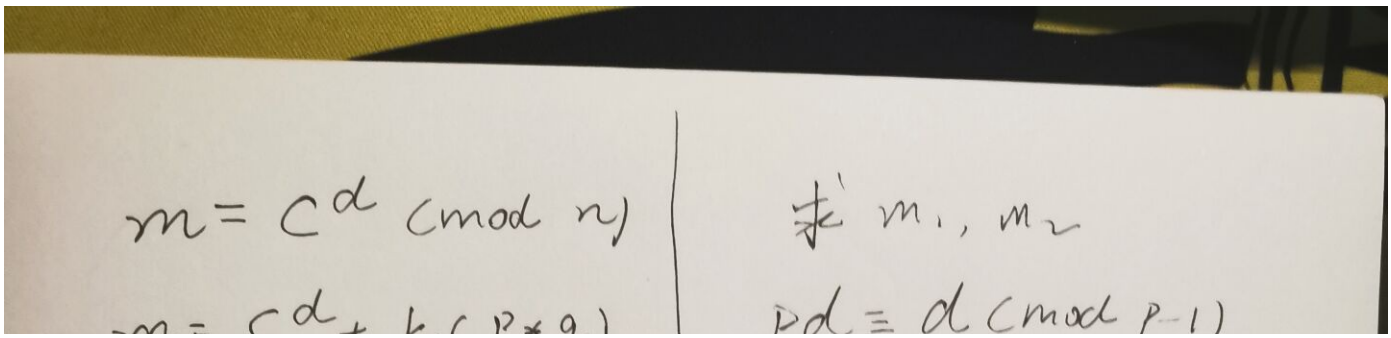
在线网站破解得到flag

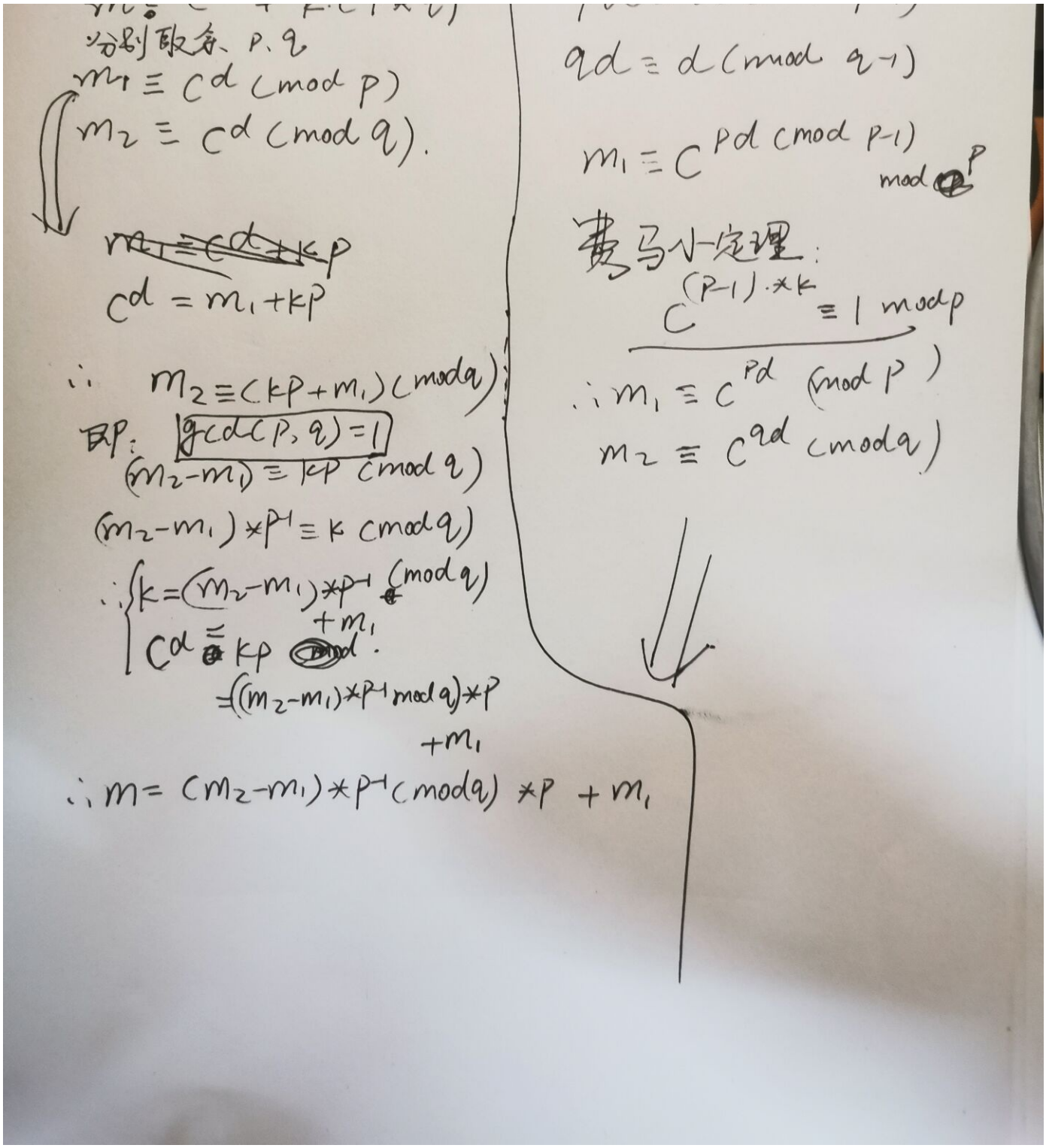
在线网站： https://dianma.bmcx.com/

0x07 RSA1

题目

```
p = 863763376725700856709965348654109117132049150943361544753916243791124417588566780639841179052408355344515811350222774520620532769093950403299469990205322 9
q = 126406749739964727691760479371708834209270508214800105815931371353724738805956137373376306297525773461470392840300825934907766305725849599542052 33688022846 9
dp = 650079570221683462110904235119326153065004384105625293093094966335862501688183284072806602615026469307610935487409984138045488171609777830726811691058292 9
dq = 783472263673553449019532580386470672380574033551303889137911760438881683674556098098256795673512201963002175438762767516968043599582527539160811120550041
c = 247223054038873820735673164676490806626315529059602293990791079956021544181760563358006388875276141640735304376570850796761573502053519452229893513160764865735995760419783398722659250627643185360890073102702785261596789374319038628924007479155251189839599706079341429747366757843259934459420313721073421038 52
```





因此脚本如下：

```
import gmpy2
p = 86376337672570085670996534865410911713204915094336154475391624379112441758856678063984117905240835534451
581135022277452062053276909395040329946999020532 29
q = 12640674973996472769176047937170883420927050821480010581593137135372473880595613737337630629752577346147
039284030082593490776630572584959954205336880228469
dp = 65007957022168346211090423511932615306500438410562529309309496633586250168818328407280660261502646930 76
109354874099841380454881716097778307268116910582929
dq = 78347226367355344901953258038647067238057403355130388913791176043888168367455609809825679567351220196 30
021754387627675169680435995825275391608111205500 41
c = 24722305403887382073567316467649080662631552905960229399079107995602154418176056335800638887527614164073
530437657085079676157350205351945222989351316076486573599576041978339872265925062764318536089007310270278526
159678937431903862892400747915525118983959970607934142974736675784325993445942031372107342103852
```

```
I = gmpy2.invert(q,p)
m1 = pow(c,dp,p)
m2 = pow(c,dq,q)
m = (((m1-m2)*I)%p)*q+m2
print(m)                                #10进制明文
print(hex(m)[2:])                       #16进制明文
print(bytes.fromhex(hex(m)[2:]))        #16进制转文本
```

```
PS D:\课件\ctf\Crypto\python> python -u "d:\课件\ctf\Crypto\python\RSA\已知pd_qd_求明文.py
116302080902045061763029611715390220427211378079118188766378217591 01
6e6f784354467b57333163306d335f37305f4368316e343730776e7d
b'noxCTF{W31c0m3_70_Ch1n470wn}'
```

0x08 RSA3

题目

```
c1=223220352756632370416468937704519335093247019134843033380762106035426127589562628696408224864701211494244
855713610074212936755163388221952803137949911360481409188424712198402635363388862504926827394364100134366511
617207258554848666900847887213495556620198790815011132229961233055330093259643777988927031615218528059568112
195638833128963301562986216746843539195475581279209257068428089147621990110549558165349776752673950095753478
203870734839284250665363614827748923709695207403042874565555089333727823275065690107725374975417643114290522
162911989320926177926452539014789108015928782035648611189120454649598325660513 61
n=227080788158850114624620490643391858987124392772268310734578884031293785473502924202670165518190524307790 0
475584664904400102414148528328648313070261605727469847361114950879886970634750193158311763271070078722801648
012767739364992953041659868602735421642256593445901516192761360790283154285797785961259628235367932777330372
700440726219723158632459918198357262240459035408454178806226216451014060586812241038809017442014775240855412
978976090230089804627390900785281847403077069964764736301510211895673767394135421769269604496969530850643657
314256557348758350703735694484803986438233921626667067356748887150892531115480 1
e1=11187289
c2=187020100451870155565486916423949828356692621472302127313099386752264585552104259724294184492734105353879
859310367118542656239050668056657518032691068807467690034789007910995902395139254497488140759040174715855728
484735564905654500626647064491284158347879619472662597897859629222387011340797204142284140661930714953046123
410529874556159300235368238014992697733571860874527475008406404193650115544211830375056534612867327409837027
408226711480456194976671845861236572856040618756539095678223289140653377977334446403515187754876498199782623
636172657979828431796308887294072384966509877204287082171152579890078673316983 97
e2=9647291
```

```
PS D:\课件\ctf\Crypto\python> python -u "d:\课件\ctf\Crypto\python\RSA\已知pd_qd_求明文.py
116302080902045061763029611715390220427211378079118188766378217591 01
6e6f784354467b57333163306d335f37305f4368316e343730776e7d
b'noxCTF{W31c0m3_70_Ch1n470wn}'
```

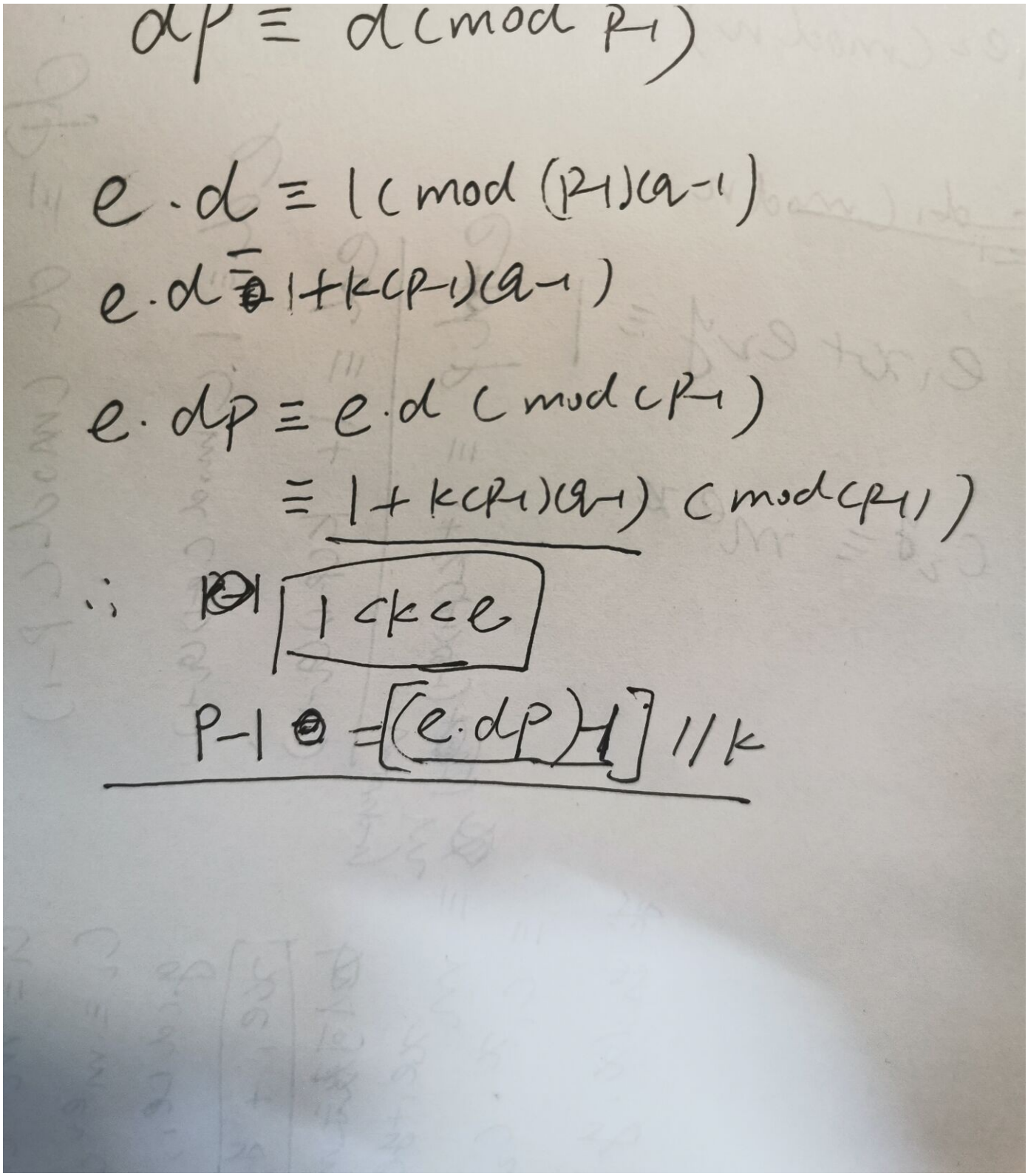
脚本如下：

```
import  gmpy2
import libnum
def exgcd(a, b):
    if b==0: return 1, 0
    x, y = exgcd(b, a%b)
    return y, x-a//b*y
e1 = 2767
e2 = 3659
n = 21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908
85746026672692561582689530337780161482936403462447519585999794314630558831593913077745048519629076624961234 0
05435462251620768154297375625767738809192654965516249087384995578376866302913864707987427824086793212719668 6
25880014691162073070673410361183317973326409647528649198806399043108538049907500562980770240667670784132466 0
97117325310095636252834668475295993747385263014589379605667579364643079357826541825591937632379604458855972 6
70385842931178470524506984593831680268157565365377088361552573569030667463516711 1
c1 = 20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403 74
10868879690407219595331900583427620573594326637178258263654449969154690390564284161661739209582430448314049 2
41134425126175994268761411842121216775003712369371275718028913217065876103936394468688369871703018130182184 0
88869682638821230841556074940763302569342851713707585865354151361628611388987289105851383788845308198574786 0
97911269713086243184549059929194053557514927891100093131384172651261172737108138439231433812762048025159105 2
74688832242748299624796365274223501902107176947629080969446002670333518139294485 99
c2 = 11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181 50
90815450270565237900225982339180112610119731963863956893715267747855823261219591861955860698515924676378193 6
66240441336610163733608851589569552636456143458813504940123282752158213069552127882826178126865488831510668 6
61490603634829587083647269829087983401822887021010233938397814273865372304594365126130473115858750680082108 1
89969414601565893141350104383624475224282068849449526398266772478190668127068357731070595670828223123007210 4
98270136604186102651892888402471865981457417240843516335084927077552068862028762 27
x,y = exgcd(e1,e2)
m = pow(c1,x,n)*pow(c2,y,n) % n
print(m)
print(hex(m))
print(bytes.fromhex(hex(m)[2:]))
```

0x09 RSA2

题目

```
e = 65537
n = 24825400785152624117772152669890180298583276617622160961225887737162058006043310153832803030521991869764 3619814200930679612109885533801335348445023751670478437073055544724280684733298051599167660303645183146161497485358633681492129668802402065797789905550489547645118787266601929429724133167768465309665906113
dp = 90507449805234690464302513287951833069192517457305400462187725331868267505542197094355201669552856036483444630319693920705664292714809329037444021050365 7

c = 14042367097625269680753367358620940057566428210068411978420352712452118899640382659743688376604187906749 4280957410201958935737360380801845453829293997433414188838725751796261702622028587211560353362847191060306578510511380965162133472698713063592621028959167072781482562673683090590521214218071160287665180751
```



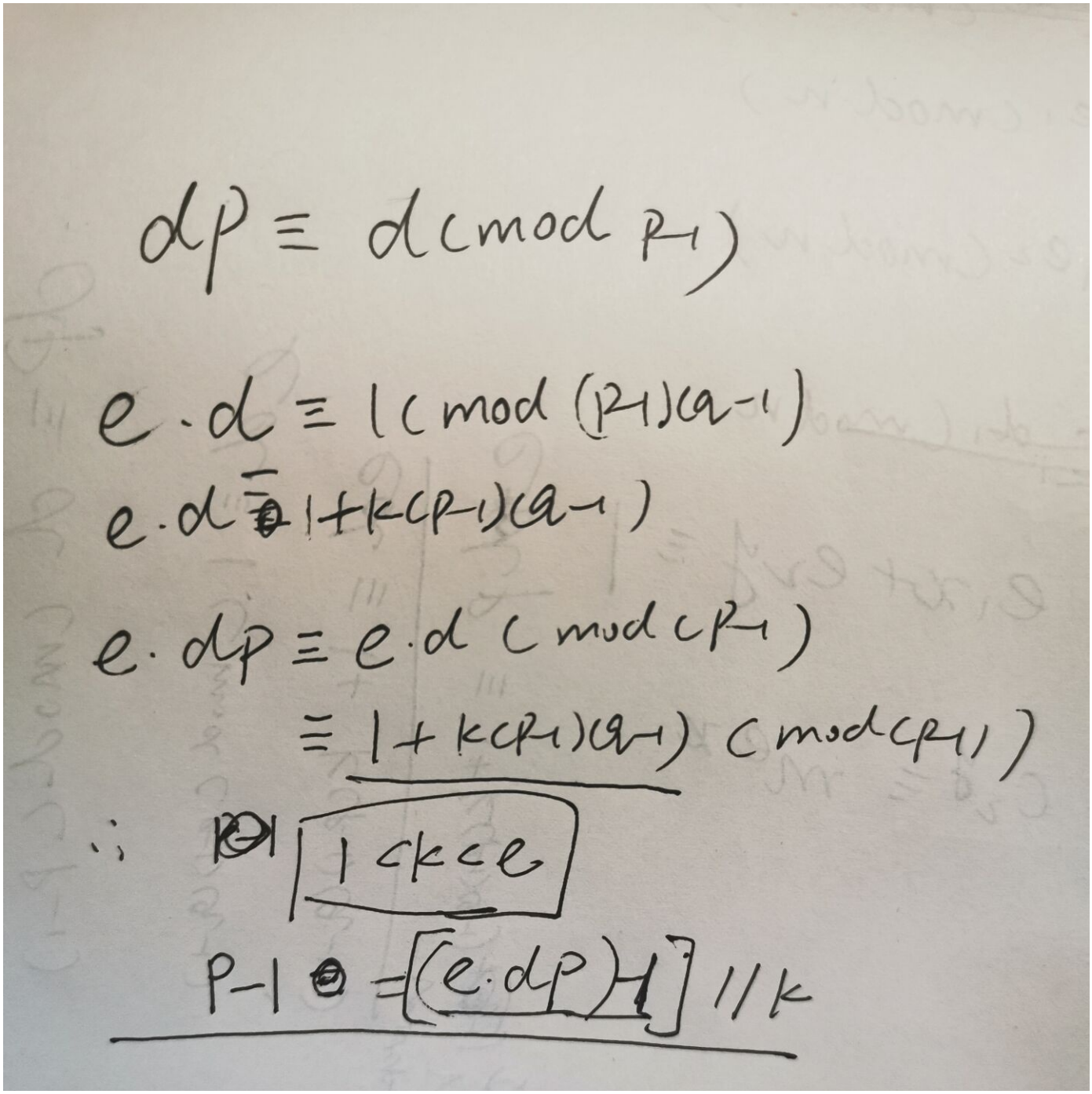
代码如下：

```
import gmpy2

e = 65537
n = 24825400785152624117772152669890180298583276617622160961225887737162058006043310153832803030521991869764
36198142009306796121098855338013353484450237516704784370730555447242806847332980515991676603036451831461614 9
74853586336814921296688024020657977899055504895476451187872666019294297241331677684653096659061 13
dp = 90507449805234690464302513287951833069192517457305400462187725331868267505542197094355201669552856036 48
34446303196939207056642927148093290374440210503657
c = 14042367097625269680753367358620940057566428210068411978420352712452118899640382659743688376604187906749
42809574102019589357373603808018454538292939974334141888387257517962617026220285872115603533628471910603065 7
```

```
for i in range(1,e):
    if( (e*dp)%i == 1):
        p = (e * dp - 1) // i  + 1
        if(n % p != 0):
            continue
        q = n // p
        phin = (q-1) * (p-1)
        d = gmpy2.invert(e,phin)
        m = gmpy2.powmod(c,d,n)
        print(m)
        print(hex(m)[2:])
        print(bytes.fromhex(hex(m)[2:]))
```

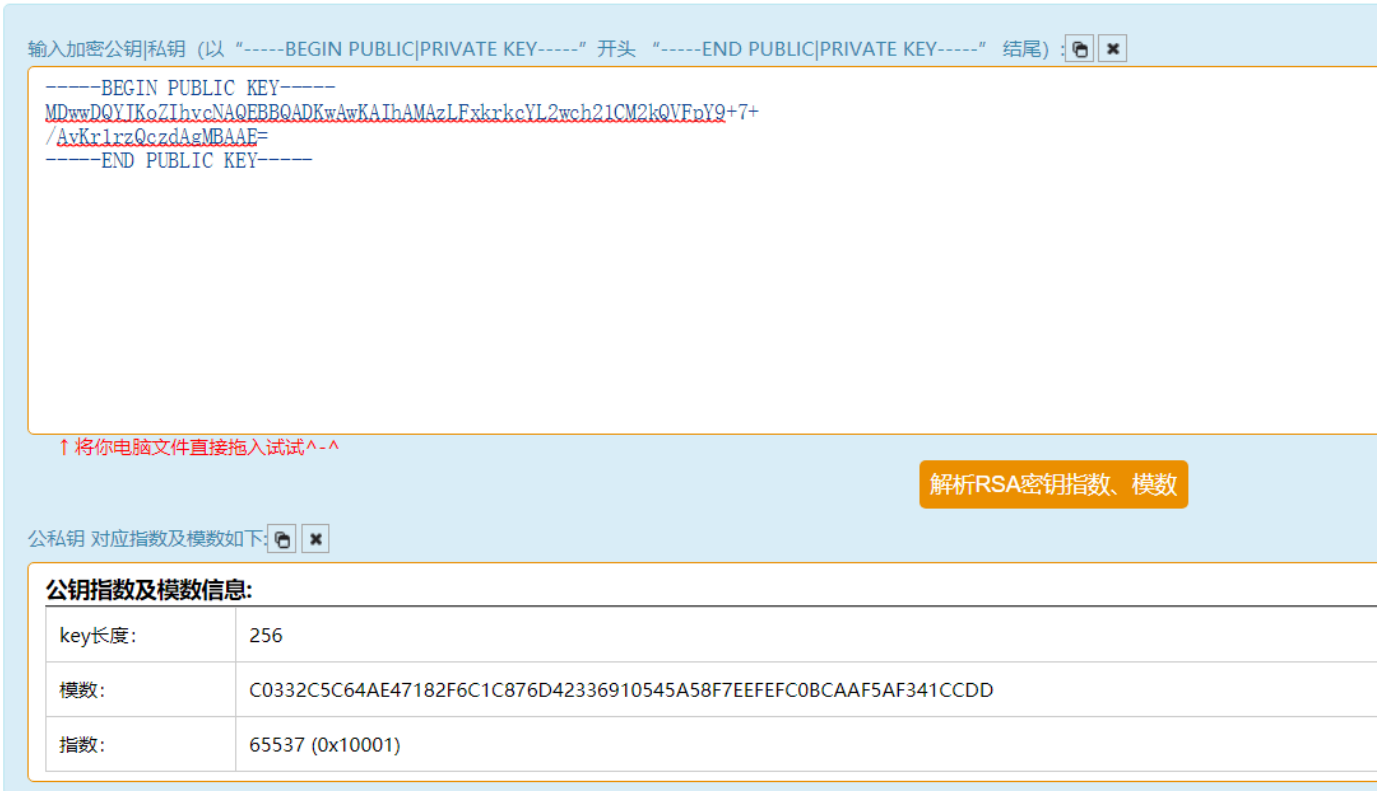
0x0A RSA

题目：（属于已知公钥文件



```
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAMAzLFxkrkcYL2wch21CM2kQVFpY9+7+
/AvKr1rzQczdAgMBAAE=
-----END PUBLIC KEY-----
```

rsa 的公钥 和 传输的 文件 flag.enc



脚本如下：

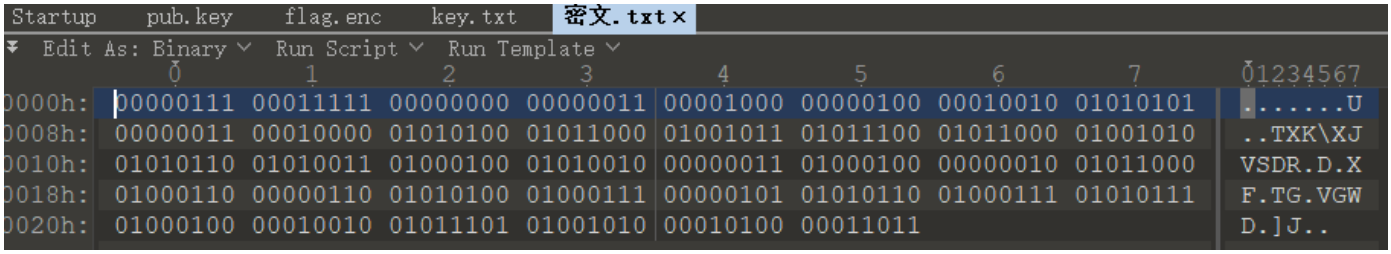
```
import gmpy2
import rsa

e=65537
n=869344822960481191906660620034948005889056560172030256172166540583783221035 17
p=285960468890451637935629440372639283459
q=304008741604601924494328155975272418463

phin = (p-1) * (q-1)
```

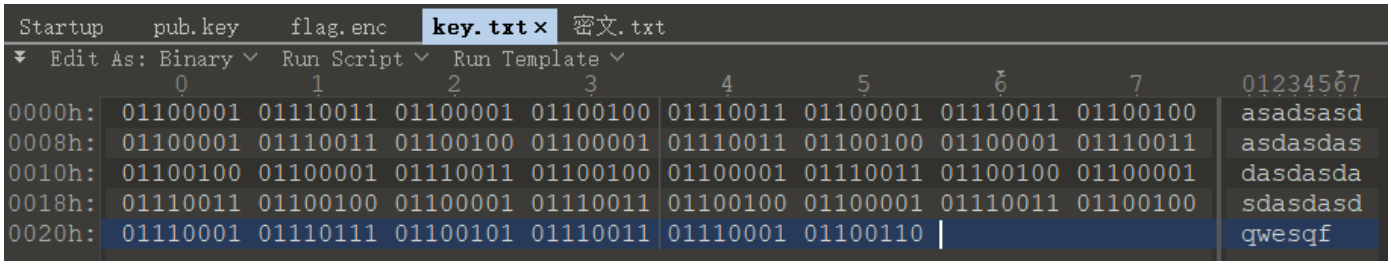
```
d=gmpy2.invert(e, phin)

key=rsa.PrivateKey(n,e,int(d),p,q)

with open("flag.enc","rb") as f:
    f=f.read()
    print(rsa.decrypt(f,key))
```

0x0B 异性相吸

解释一下，yxx这道题实际上和异性相吸时一样的解题思路，因为它——yxx可能是少打了x(个人猜测)，因为异性相吸的首字母就是yxxx，很相似。

而题目就是南邮ctf的异性相吸，最近做的这种题的可能有印象，就是给了一个明文.txt和一个密文.txt，给了提示的题，而这道yxx没给提示





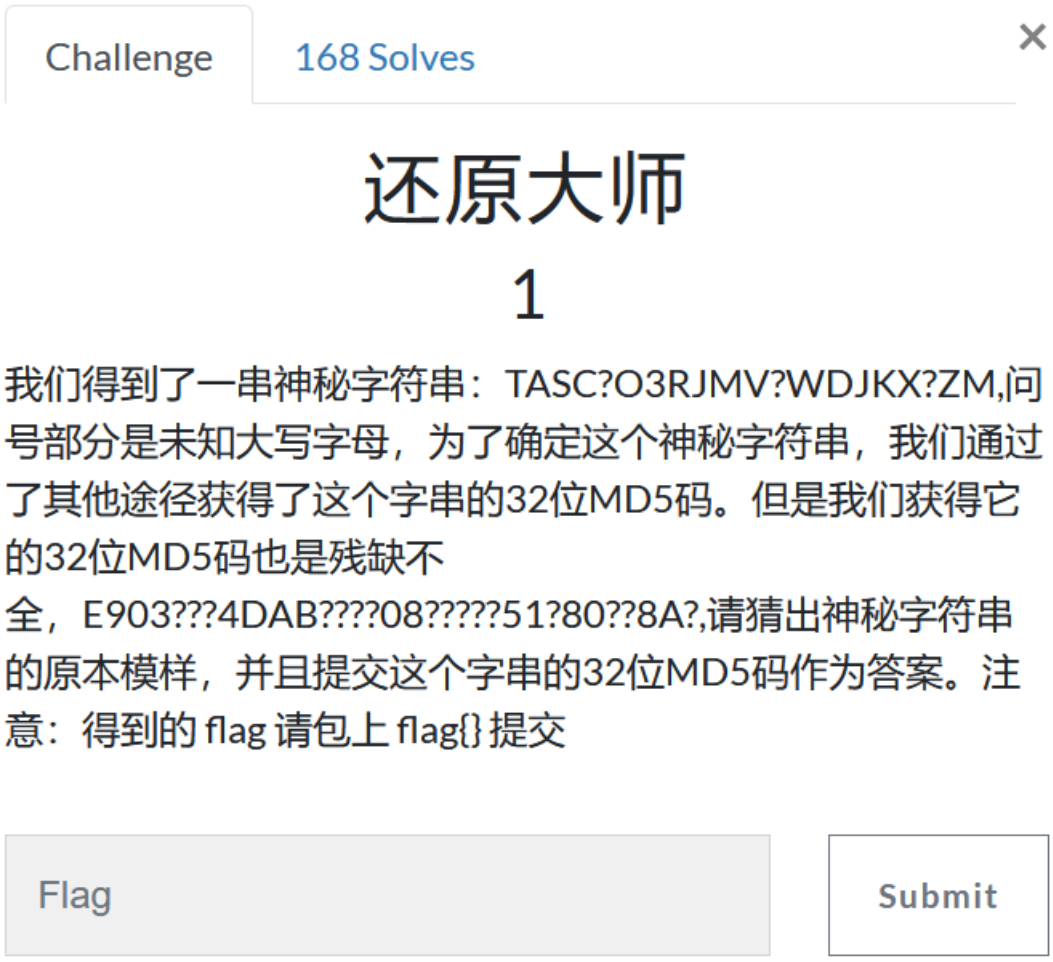
1.xor
2.hex2binary
3.len(bin(miwen))==len(bin(mingwen))

写脚本对其进行逐bit 异或

```
a = '0000101000000011000101110000001001010110000000010001010100010001000010100001010000001110000010100001111
000110000000011100000101000011110001100000000111000001010000111100011000000010100000011000001100100001101000
111110001000000001110000001100000001100011000'
b = '0110110001101111011101100110010101101100011011110111011001100101011011000110111101110110011001010110110
001101111011101100110010101101100011011110111011001100101011011000110111101110110011001010110110001101111011
101100110010101101100011011110111011001100101'
for i in range(len(a)):
    if(a[i] == b[i]):
        print '0'
    else:
        print '1'
```

0x0C 还原大师

题目：



可以看到 原文中三个 问号， 以生成的md5中的 E903为参照 进行爆破

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
import hashlib

#print hashlib.md5(s).hexdigest().upper()
k = 'TASC?O3RJMV?WDJKX?ZM'                      #要还原的明文
for i in range(26):
 temp1 = k.replace('?',str(chr(65+i)),1)
 for j in range(26):
  temp2 = temp1.replace('?',chr(65+j),1)
  for n in range(26):
   temp3 = temp2.replace('?',chr(65+n),1)
   s = hashlib.md5(temp3.encode('utf8')).hexdigest().upper()#注意大小写
   if s[:4] == 'E903':    #检查元素
    print (s)       #输出密文
```

flag{E9032994DABAC08080091151380478A2}

0x0D rsaroll

题目：

```
{920139713,19}

704796792
752211152
274704164
18414022
368270835
483295235
263072905
459788476
483295235
459788476
663551792
475206804
459788476
428313374
475206804
459788476
425392137
704796792
458265677
341524652
483295235
534149509
425392137
428313374
425392137
341524652
458265677
263072905
483295235
828509797
341524652
425392137
475206804
428313374
483295235
475206804
459788476
306220148
```

发现对每个 c 对应的原文m 对其进行ascii 编码 为字符

因此脚本如下：

```
import gmpy2

n = 920139713
e = 19
p = 18443
q = 49891

d = gmpy2.invert(e,(p-1)*(q-1))
c = [704796792,
752211152,
```

```
274704164,
18414022,
368270835,
483295235,
263072905,
459788476,
483295235,
459788476,
663551792,
475206804,
459788476,
428313374,
475206804,
459788476,
425392137,
704796792,
458265677,
341524652,
483295235,
534149509,
425392137,
428313374,
425392137,
341524652,
458265677,
263072905,
483295235,
828509797,
341524652,
425392137,
475206804,
428313374,
483295235,
475206804,
459788476,
306220148]

y = ''.join(chr(pow(i,d,n)) for i in c  )
print(y)
```

8476,
663551792,
475206804,
459788476,
428313374,
475206804,
459788476,
425392137,
704796792,
458265677,
341524652,
483295235,
534149509,
425392137,
428313374,
425392137,
341524652,
458265677,
263072905,
483295235,
828509797,
341524652,
425392137,
475206804,
428313374,
483295235,
475206804,
459788476,
306220148]

```
y = ‘’.join(chr(pow(i,d,n)) for i in c )
print(y)
```