[buuctf] crypto全解——前84道（不建议直接抄flag）

原创

buuctf crypto

23.信息化时代的步伐

24.RSA1

25.凯撒？替换？呵呵!

26.old-fashion

27.[BJDCTF 2nd]灵能精通-y1ng

28.权限获得第一步

29.萌萌哒的八戒

30.RSA3

31.RSA2

32.[BJDCTF 2nd]Y1nglish-y1ng

33.世上无难事

34.异性相吸

35.RSA

36.还原大师

37.[GKCTF2020]汉字的秘密

38.robomunication

39.RSAroll

40.Unencode

41.Dangerous RSA

42.Cipher

43.[AFCTF2018]Morse

44.[HDCTF2019]basic rsa

45.达芬奇密码

46.ras2

47.[BJDCTF 2nd]rsa0

48.[GXYCTF2019]CheckIn

49.RSA5

50.传感器

51.[GUET-CTF2019]BabyRSA

52.密码学的心声

53.rot

54.这是什么

55.[BJDCTF 2020]这是base？？

56.[NCTF2019]Keyboard

57.[BJDCTF 2nd]rsa1

58.[NCTF2019]childRSA

59.[HDCTF2019]bbbbbbrsa

60. [MRCTF2020]vigenere

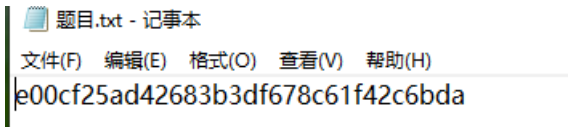
61.[BJDCTF2020]RSA

62.一张谍报

63.[MRCTF2020]古典密码知多少

1.MD5

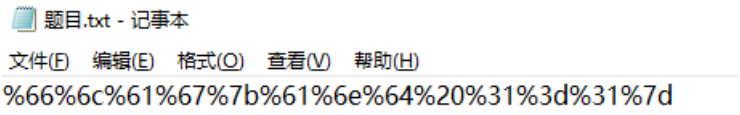
获得题目



题目名字就叫MD5，那我们直接把这个复制到md5在线解密即可得到flag

2.Url编码

获得题目



老规矩这种题目直接复制url解码里面解密即可得到flag

3.一眼就解密

看到题目

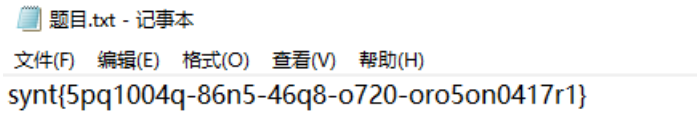
一眼就解密

1

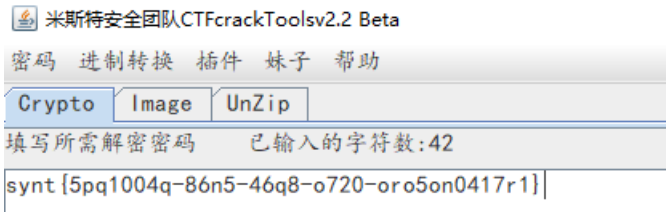
下面的字符串解密后便能获得flag:
ZmxhZ3tUSEVfRkxBR19PRI9USEITX1NUUklOR30= 注意:
得到的 flag 请包上 flag{} 提交

题目后面有=就先猜一手base64编码，直接复制base64解码解密即可得到flag

4.看我回旋踢



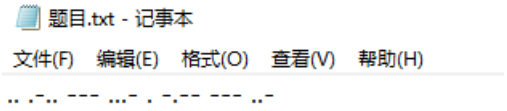
看题目觉得是凯撒密码



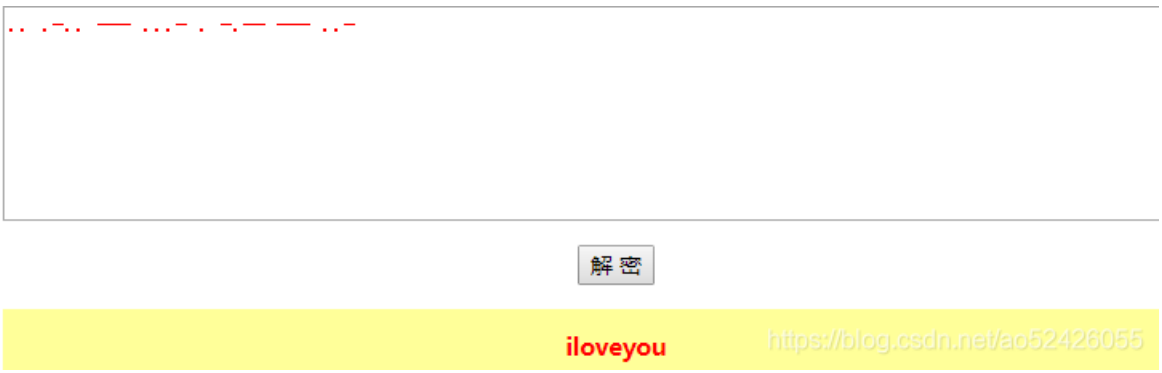
直接复制到CTFcrackToolsv里面解密

5.摩丝

查看题目



摩斯密码在线解密



解密完用flag{}包裹iloveyou提交即可

6.[BJDCTF 2nd]签到-y1ng

查看题目

[BJDCTF 2nd]签到-y1ng

1

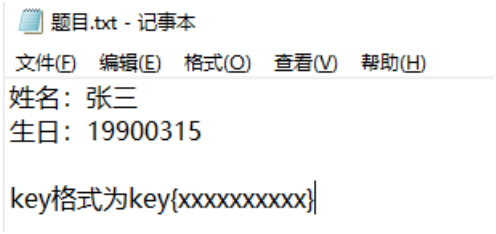
welcome to BJDCTF
1079822948
QkpEe1czbGMwbWVfVDBfQkpEQ1RGfQ==

见到=直接先试一下base64解密

注意：这里要把根据要求把BJD换成flag，然后提交即可。

7.password

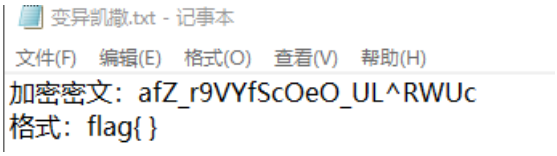
查看题目



这个说实话有点看运气
我是数了一下x有十个
然后就是zs+19900315
用flag包裹提交嗯然后就对了

8.变异凯撒

查看题目



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	32	20	(space)	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	SLE	48	30	0	80	50	P	112	70	p
17	11	CS1	49	31	1	81	51	Q	113	72	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SIB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

又因为明文flag对应afZ_,所以寻找明文和密文的规律

f-102 a-97 相差5

l-108 f-102 相差6

a-97 Z-90 相差7

g-103 _-95 相差8

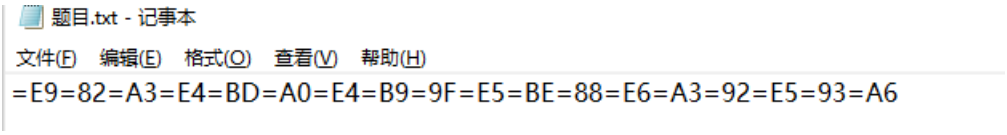
可以看出每个字符的偏移量为n+4

所以依次算出各密文字符对应的明文字符求得明文为

flag{Caesar_variation}

9.Quoted-printable

查看题目

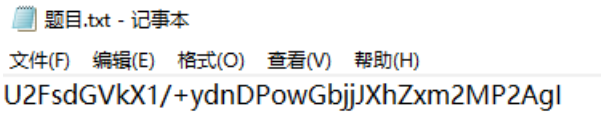


首先这个题目就是试一加密编程
试一我们直接用Quoted-printable解密即可

把得到的 那你也很棒哦 用flag包裹即可

10.Rabbit

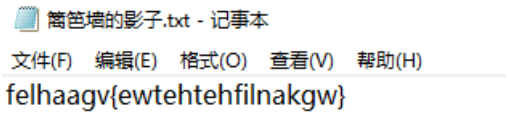
查看题目



题目叫Rabbit这是一个加密，我们直接Rabbit在线解密即可

11.篱笆墙的影子

查看题目



篱笆墙很明显联想到栅栏密码，用栅栏在线解密即可

12.RSA

查看题目

学习RSAtool2的使用：

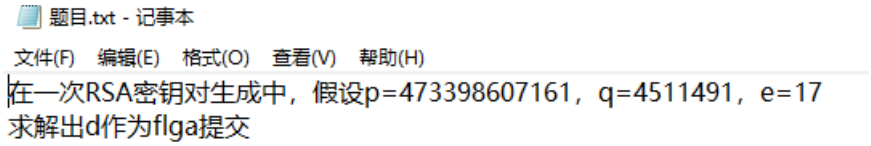
1.Number Base 设置为十进制

2.注意：Public Exponent这里要使用16进制的数，如果公钥e=17的话，就应该填入十六进制的11

3.给出p,q,e的话直接填入，再点击Calc.D,获得d

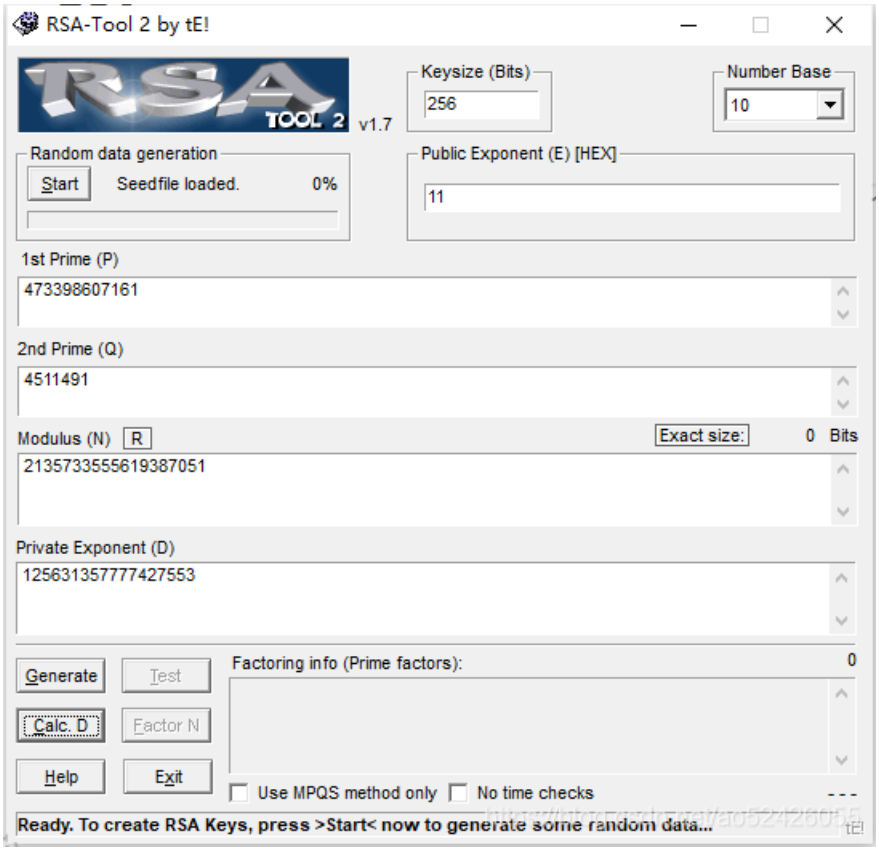
4.给出的是n和e的话，输入n和e，点击Factor N(分解)，得到p,q,再重复第3步就能得到d了

注意e填进去是16进制，需要将17转hex得到11再填进去



用RSA-Tool 2 by 1E！即可

RSA-Tool下载链接



13.丢失的MD5

查看题目

```
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
```

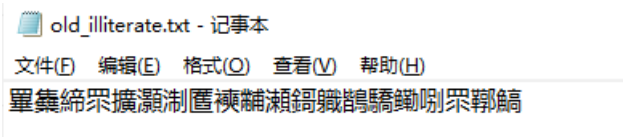

看代码用python2.x的版本运行即可获得flag

e9032994dabac08080091151380478a2

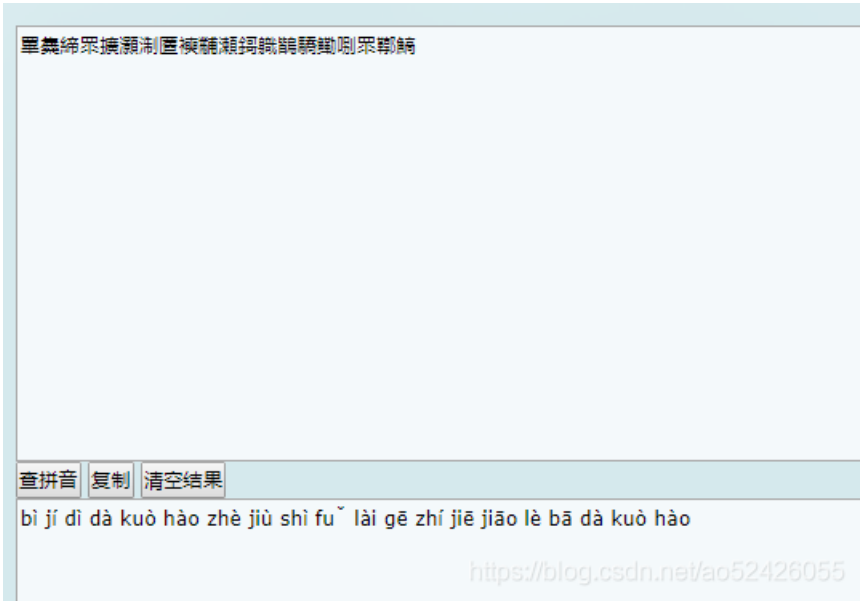
运行结果即是flag

14.[BJDCTF 2nd]老文盲了

查看题目



题目名字文盲，而且这txt里面的文字基本都不认识，那就用拼音解密去把拼音翻译出来



flag: BJD{淛匱襫黼瀨鍔膱鵲驕鰳唰}

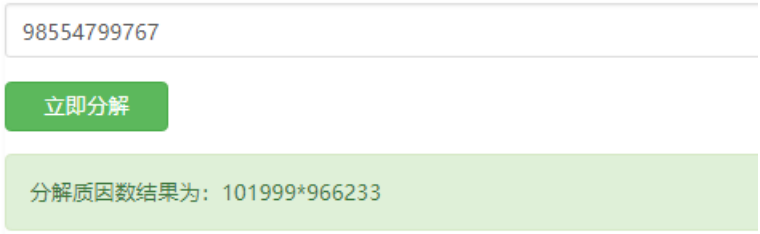
15.Alice与Bob

查看题目

密码学历史中，有两位知名的杰出人物，Alice和Bob。他们的爱情经过置换和轮加密也难以混淆，即使是没有身份认证也可以知根知底。就像在数学王国中的素数一样，孤傲又热情。下面是一个大整数:98554799767,请分解为两个素数，分解后，小的放前面，大的放后面，合成一个新的数字，进行md5的32位小写哈希，提交答案。 注意：得到的 flag 请包上 flag{} 提交

题目都说了分解素数，小前大后

素数分解

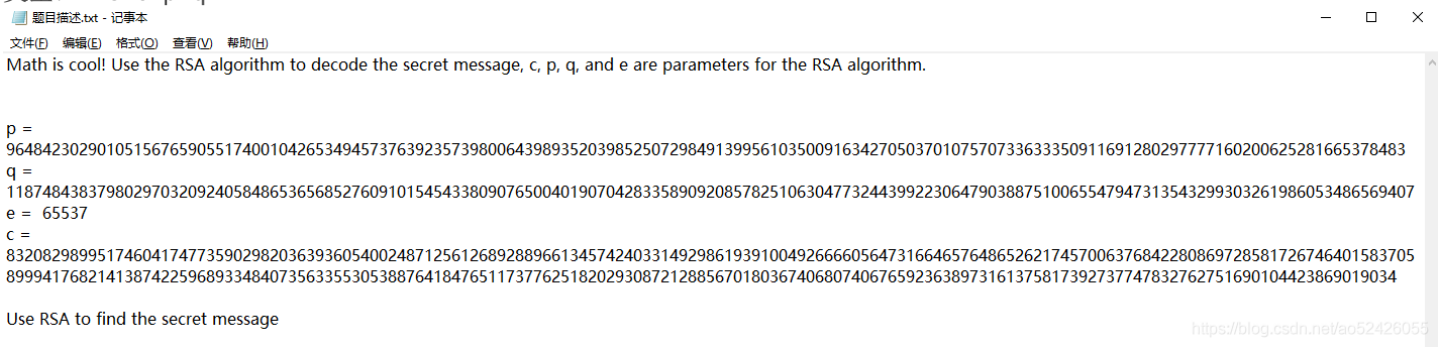


直接可以看出flag{101999966233}

16.rsarsa

查看题目

类型：n+e+c+p+q= m

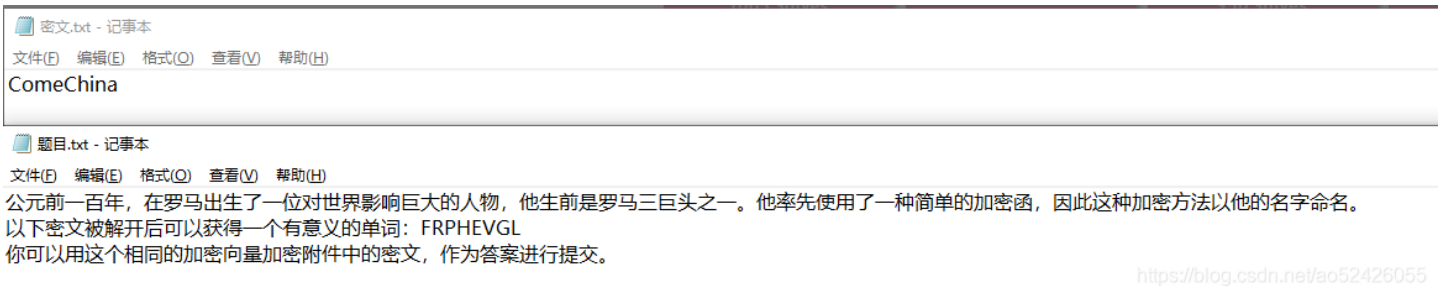



```
e = 65537
p = 964842302901051567659055174001042653494573763923573980064398935203985250729849139956103500916342705037010757
07336333509116912802977771602006252816653784 83
q =
118748438379802970320924058486536568527609101545433809076500401907042833589092085782510630477324439922306479038
75100655479473135432993032619860534865694 07
n = p*q
#密文
C = 832082989951746041747735902982036393605400248712561268928896613457424033149298619391004926660564731664657 64
865262174570063768422808697285817267464015837058999417682141387422596893348407356335530538876418476511737762518
02930872128856701803674068074067659236389731613758173927377478327627516901044238690190 34

d = 566320475711906605675203410288611948624114284168625070347625872299951386056498369602206199034563927521159432
993353851632162337446246238488742353033096363934467363472386277930227252609864669579747530041292106804014323774 4
498419514500980196739119661552448885362023292599238756327074629790911211745139852745397 7

#求明文
M = pow(C,d,n)     #快速求幂取模运算
print(M)
```

解出flag{557744663355446657776887998 8}

17.大帝的密码武器

下载的zip没有后缀，那就加一个zip为后缀
然后即可查看题目



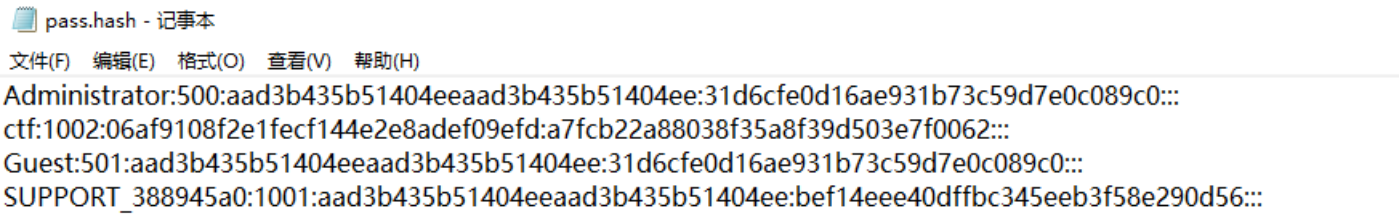
大帝的武器，基本可以猜是凯撒密码

```
str1 = 'FRPHEVGL'
str2 = str1.lower()                                 #转换为小写方便识别
num = 1                                             #偏移量
for i in range(26):
    print("{:<2d}".format(num),end = ' ')
    for temp in str2:
        if(ord(temp)+num > ord('z')):               #如果超出'z',需要重新映射会a~z这26个字母上
            print(chr(ord(temp)+num-26),end = '')
        else:
            print(chr(ord(temp)+num),end = '')
    num += 1
    print('')
    str = 'ComeChina'
    for temp in str:
        if (ord(temp) + 13 > ord('z')):
            print(chr(ord(temp) + 13 - 26), end='')
        else:
            print(chr(ord(temp) + 13), end='')
    print('')
```

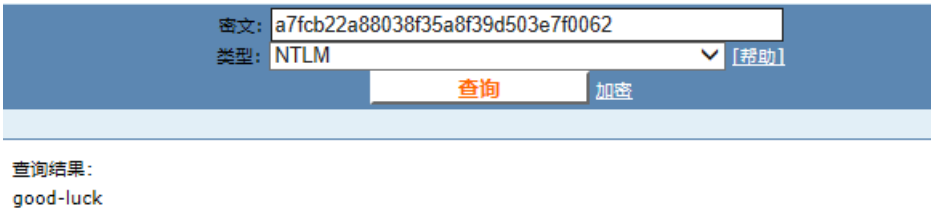
可以看到偏移量是13的时候，好像是我们想要的东西，然后将密文里面的ComeChina做偏移量为13的偏移：然后如果超出z，减26使其回到A-z范围内（别问我为什么，因为不减的结果P|zrPuv{n经过我的验证是不对的），最终得到PbzrPuvan，用花括号包起来就可以提交了flag{PbzrPuvan}

18.Windows系统密码

查看题目



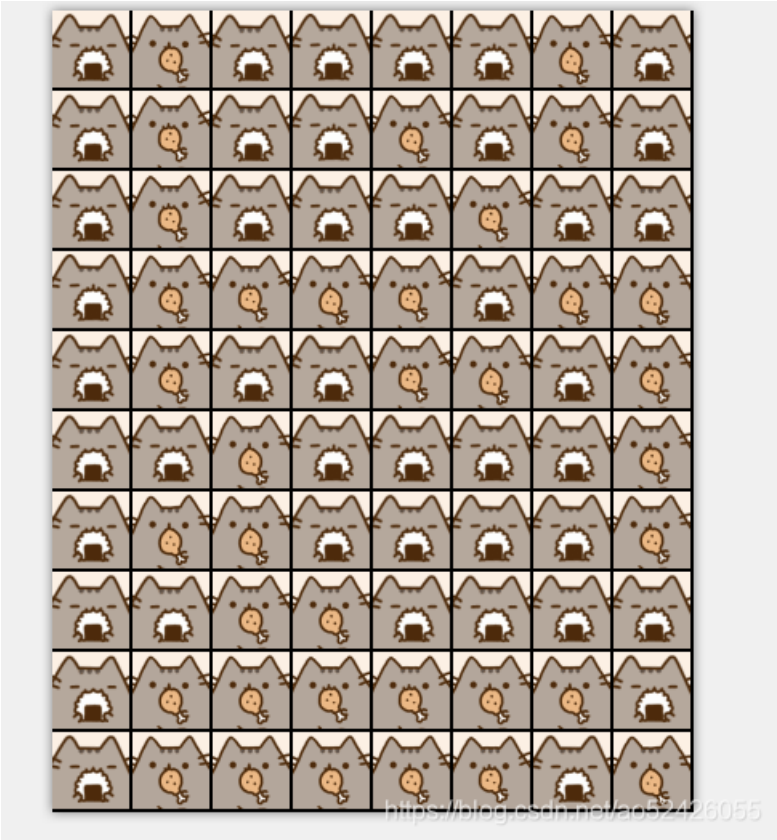
用md5解密来解一下ctf



第二段解出flag

19.[BJDCTF 2nd]cat_flag

查看题目



一只有鸡腿，一只没有鸡腿。很容易想到二进制数0，1。

将图片用二进制表示为：

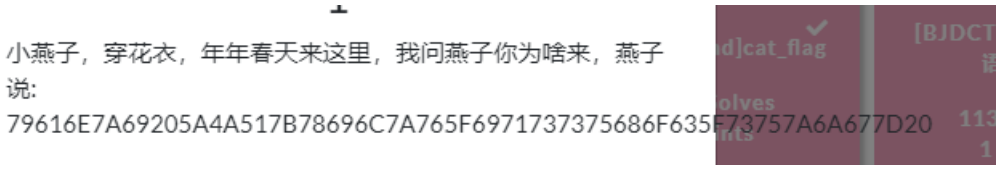
01000010
01001010
01000100
01111011
01001101
00100001
01100001
00110000
01111110
01111101

将二进制数转为16进制进制转换，再16进制转文本16转文。

得到flag为： BJD{M!a0~}

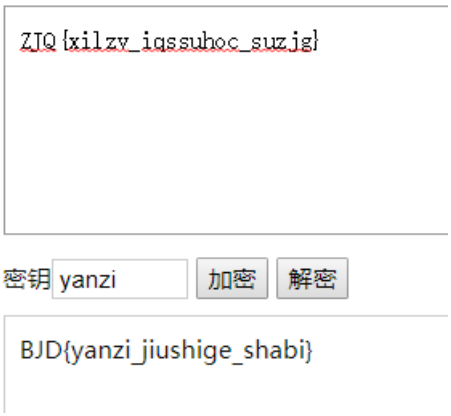
20.[BJDCTF 2nd]燕言燕语-y1ng

查看题目



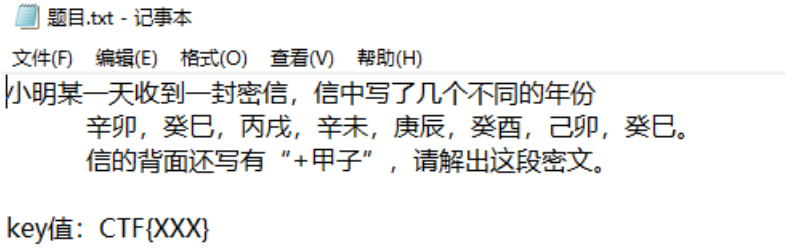
燕子说79616E7A69205A4A517B78696C7A765F6971737375686F635F73757A6A677D20
明显是16进制，16转文，转换一下。

明显是维吉尼亚密码加密，维吉尼亚密码在线解密



21.传统知识+古典密码

查看题目



加密方法：
①置换密码(又称易位密码)：明文的字母保持相同，但顺序被打乱了。
代表：栅栏加密

②代替密码：就是将明文的字符替换为密文中的另一种的字符，接收者只要对密文做反向替换就可以恢复出明文。

代表：恺撒加密

六十年甲子表

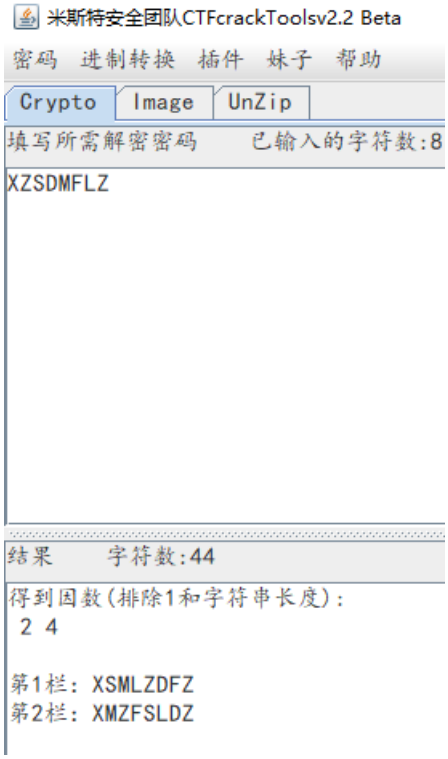
01 甲子	11甲戌	21甲申	31 甲午	41甲辰	51甲寅
02 乙丑	12 乙亥	22乙酉	32 乙未	42乙巳	52 乙卯
03 丙寅	13丙子	23 丙戌	33丙申	43丙午	53丙辰
04丁卯	14丁丑	24丁亥	34丁酉	44丁未	54丁巳
05戊辰	15戊寅	25戊子	35戊戌	45戊申	55 戊午
06己巳	16己卯	26己丑	36 己亥	46 己酉	56 己未
07庚午	17 庚辰	27庚寅	37 庚子	47 庚戌	57 庚申
08辛未	18 辛巳	28 辛卯	38 辛丑	48 辛亥	58 辛酉
09壬申	19壬午	29 壬辰	39 壬寅	49 壬子	59 壬戌
10癸酉	20癸未	30 癸巳	40 癸卯	50 癸丑	60 癸亥

1.由这张表就能知道，辛卯，癸巳，丙戌，辛未，庚辰，癸酉，己卯，癸巳。代表的数值了。

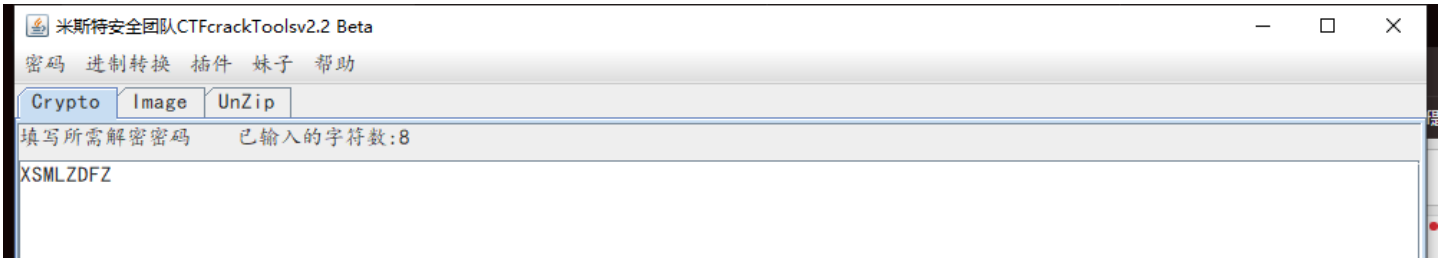
2.信的背面还写有“+甲子”，一甲子是60年，‘+甲子’ == ‘+60’

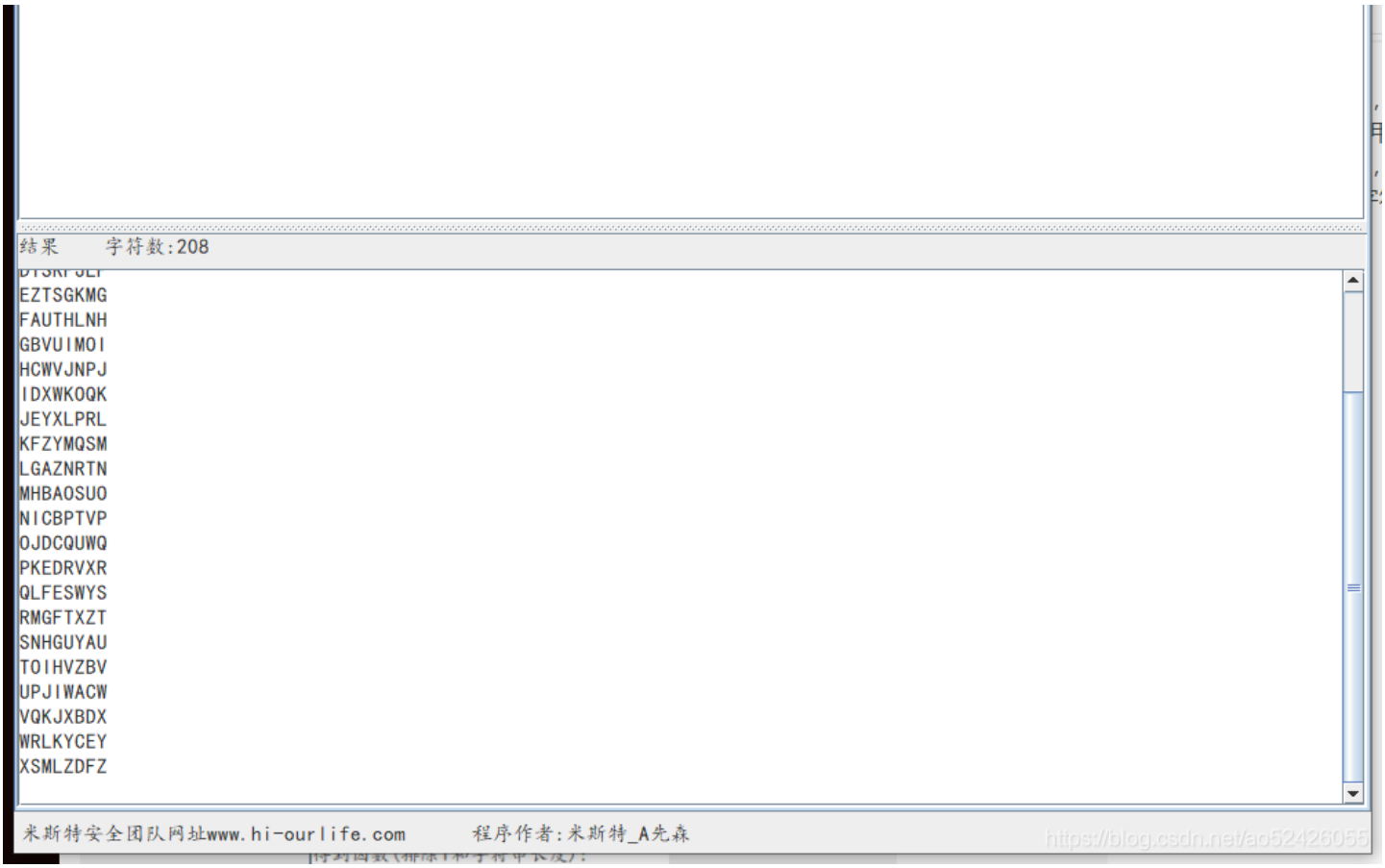
3.所以，辛卯，癸巳，丙戌，辛未，庚辰，癸酉，己卯，癸巳，再加上60，利用ASCLL码，就能得到对应的ACSLL字符了：XZSDMFLZ

4.把这个用栅栏解密



把这两个结果用凯撒解密





SNHGUYAU

这个比较通顺，即是flag

22.[GKCTF2020]小学生的密码学

查看题目

仿射密码的代码实现 破解代码

```
#include <iostream>
#include<math.h>
#include<string.h>
using namespace std;
```

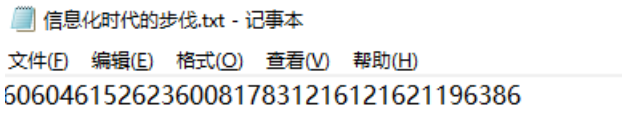
```
using namespace std;

//模的取逆
int dx, y, q;
void extend_Eulid(int aa, int bb)
{
 if (bb == 0) {
  dx = 1; y = 0; q = aa;
 }
 else {
  extend_Eulid(bb, aa % bb);
  int temp = dx;
  dx = y;
  y = temp - aa / bb * y;
 }
}
//

int main()
{
 int a, b, YN, i, l;
 char c[100];
 int x[100];
 char ex[100], y[100];

 cout << "请依次输入k =( a, b )的a, b值，其中 a,b ∈ Z/(26)，gcd( a,26) = 1 ： " << endl;
 cin >> a >> b;
 cout << "那么你的加密函数就是 ex = " << a << "*x + " << b << endl;
 cout << endl << "接下来输入你要加密的明文(小写字母)： " << endl;
 cin >> c;//明文
 l = strlen(c);

 for (i = 0; i < l; i++)
 {
  x[i] = c[i] - 'a';
  ex[i] = (a * x[i] + b) % 26;//数字
 }
 cout << "加密后的字母为：";

 for (i = 0; i < l; i++)
 {
  cout << char(ex[i] + 'a');//转字符
 }
 cout << endl << endl;
 cout << "是否要解密原文（输入1则确定，输入其他则取消）:";
 cin >> YN;

 while (YN == 1)
 {
  extend_Eulid(a, 26);//取逆
  dx = (dx + 26) % 26;
  cout << dx<<endl;
  for (i = 0; i < l; i++)
  {
   y[i] = (dx * int(x[i]) - dx * b) % 26;
   y[i] = (y[i] + 26) % 26;//+26取正
   cout << char(y[i]+ 'a' );
  }
  break;
 }
```

```
}
```

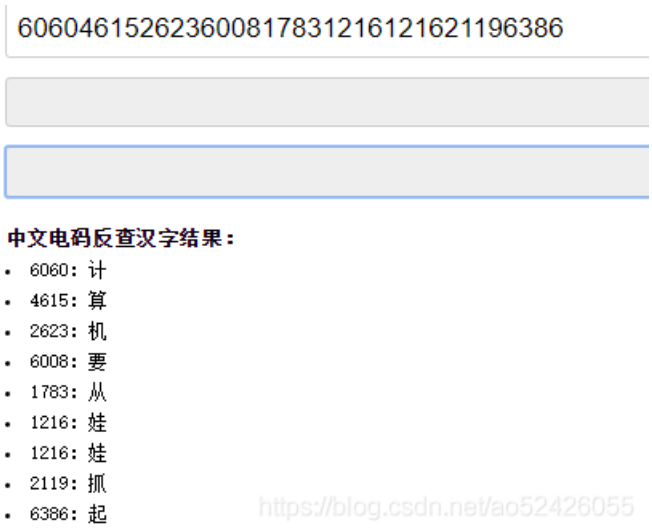
```
请依次输入k =( a, b )的a, b值，其中 a,b ∈ Z/(26)，gcd( a,26) = 1 :
11 6
那么你的加密函数就是 ex = 11*x + 6

接下来输入你要加密的明文(小写字母):
welcylk
加密后的字母为: oyxckxm

是否要解密原文（输入1则确定，输入其他则取消）:1
19
sorcery
--------------------------------
Process exited after 18.51 seconds with return value 0
请按任意键继续. . .
```


23.信息化时代的步伐

查看题目



中文电码在线查询



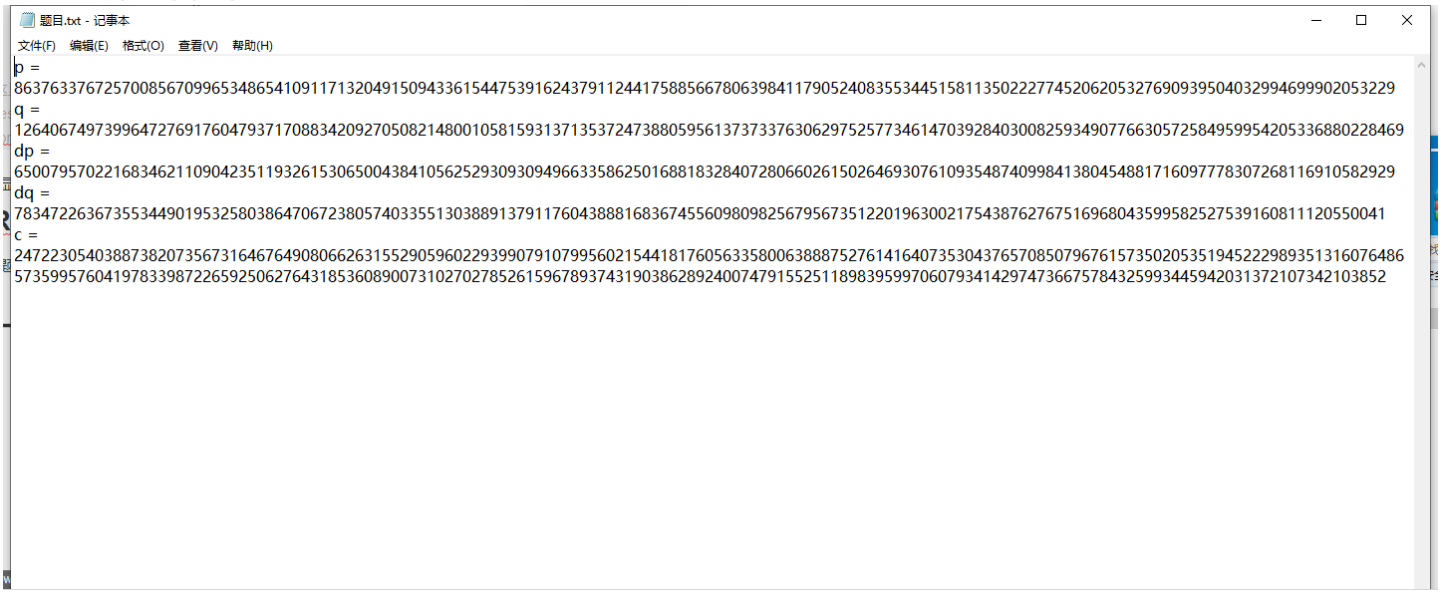
很明显了flag就是 计算机要从娃娃抓起

24.RSA1

查看题目

类型：dp+dq+p+q+c = m 已知dp dq泄露

使用脚本dp+dq+p+q+c = m



这个就用python来写即可

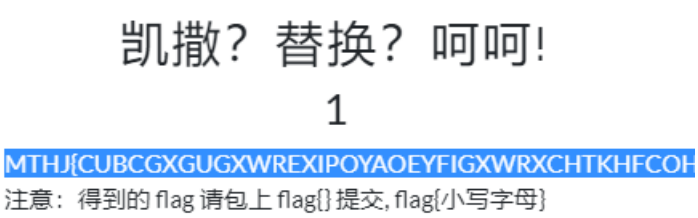
```
p = 863763376725700856709965348654109117132049150943361544753916243791124417588566780639841179052408355344515811350222774520620532769093950403299469990205322 9
q = 126406749739964727691760479371708834209270508214800105815931371353724738805956137373376306297525773461470392840300825934907766305725849599542053368802284 69
dp = 650079570221683462110904235119326153065004384105625293093094966335862501688183284072806602615026469307610935487409984138045488171609777830726811691058292 9
dq = 783472263673553449019532580386470672380574033551303889137911760438881683674556098098256795673512201963002175438762767516968043599582527539160811120550041
c = 247223054038873820735673164676490806626315529059602293990791079956021544181760563358006388875276141640735304376570850796761573502053519452229893513160764865735995760419783398722659250627643185360890073102702785261596789374319038628924007479155251189839599706079341429747366757843259934459420313721073421038 52

import gmpy2
I = gmpy2.invert(q,p)
mp = pow(c,dp,p)
mq = pow(c,dq,q)                #求幂取模运算

m = (((mp-mq)*I)%p)*q+mq        #求明文公式

print(hex(m))           #转为十六进制
```

```
D:\python38\python.exe D:/pycharm/venv/mima/RSA1.py
0x6e6f784354467b57333163306d335f37305f4368316e343730776e7d
```

运行的出0x6e6f784354467b57333163306d335f37305f4368316e343730776e7d
很明显是十六进制，我们直接十六转文

即可得到flag

25.凯撒？替换？呵呵!



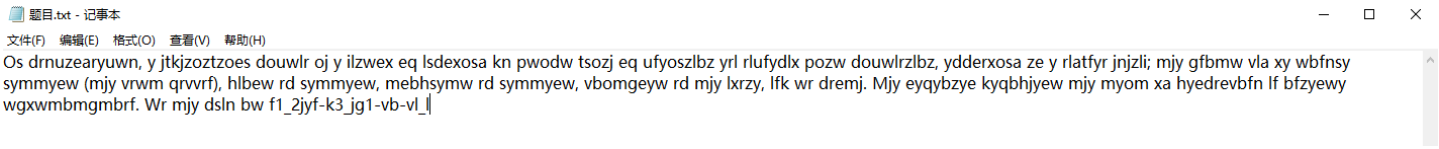
MTHJ{CUBCGXGUGXWREXIPOYAOEYFIGXWRXCHTKHFCOHCFDUCGTXZOHIXOEOWMEHZO}

强行爆破

FLAG{ SUBSTITUTION CIPHER DECRYPTION IS ALWAYS EASY JUST LIKE A PIECE OF CAKE}得到结果即为flag，大写还是小写忘了，这是大小写转换的在线网页

26.old-fashion

查看题目



跟上一道题一样直接强行爆破


```
So the flag is nl_2hen-d3_hul-mi-ma_a
```

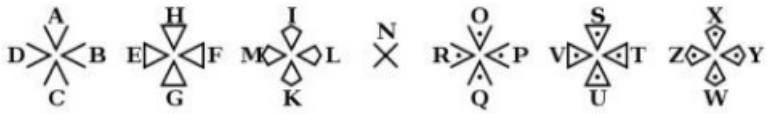
27.[BJDCTF 2nd]灵能精通-y1ng

老规矩，没后缀跟着前面加就是

得到一张图片

看着像是猪圈，就在网上找了找。

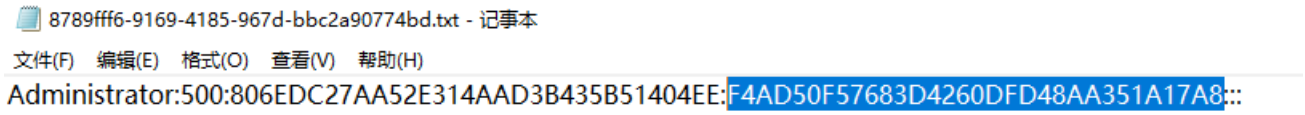
这是猪圈加密的变形圣堂武士密码



翻译即可得到flag

28.权限获得第一步

查看题目



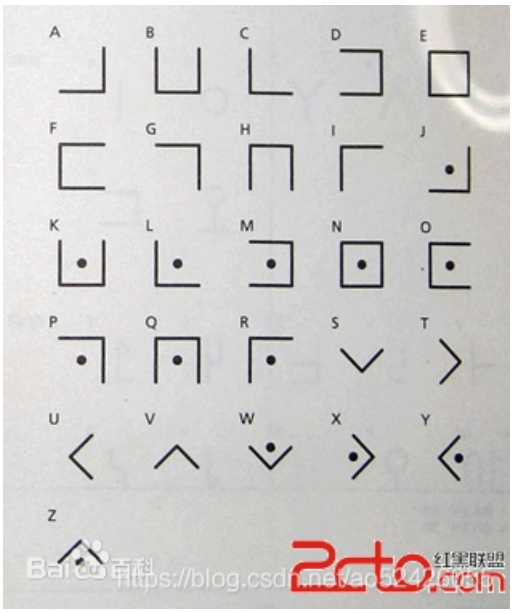
直接md5在线解密

即可得到flag

29.萌萌哒的八戒

查看题目

猪圈密码

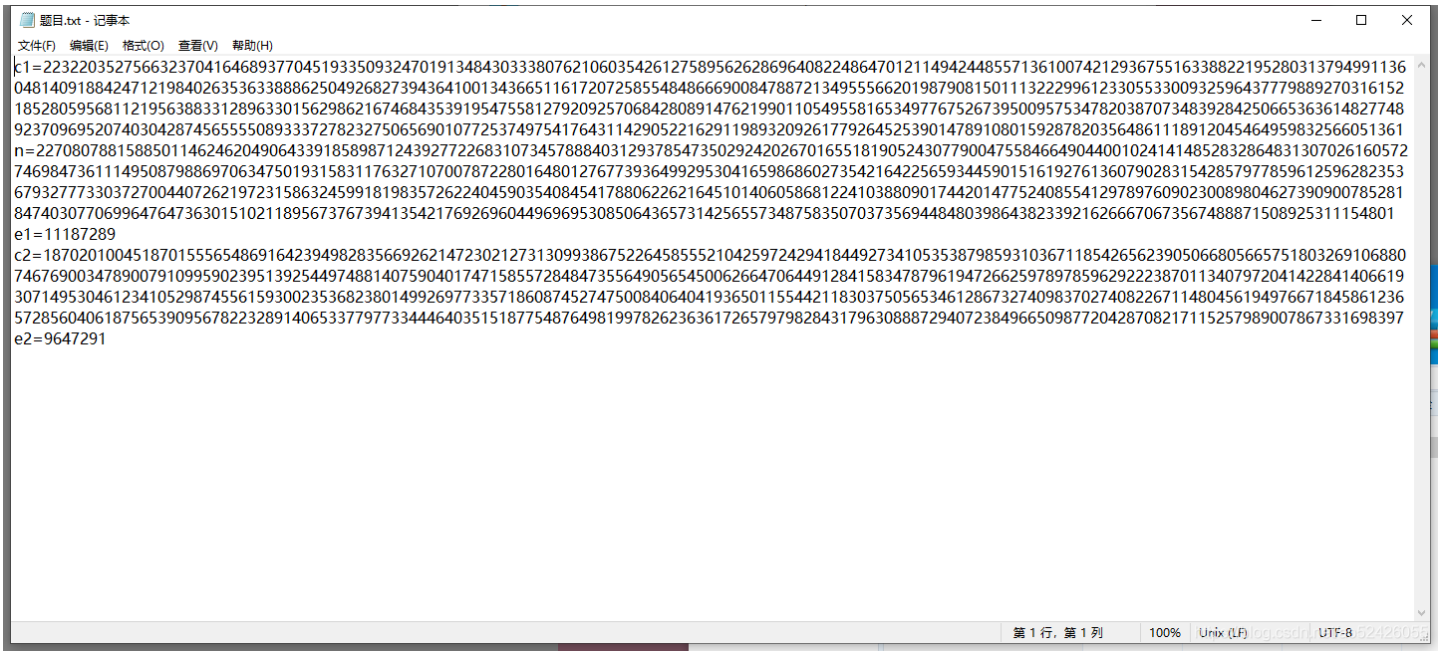


对着解密即可

30.RSA3

查看题目

类型：共模n攻击



```python
from gmpy2 import invert
# 欧几里得算法
def egcd(a, b):
  if a == 0:
    return (b, 0, 1)
  else:
    g, y, x = egcd(b % a, a)
    return (g, x - (b // a) * y, y)

def main():
  n = 22708078815885011462462049064339185898712439277226831073457888403129378547350292420267016551819052430779004755846649044001024141485283286483130702616057274698473611149508798869706347501931583117632710700787228016480127677393649929530416598686027354216422565934459015161927613607902831542857977859612596282353679327773303727004407262197231586324599181983572622404590354084541788062262164510140605868122410388090174420147752408554129789760902300898046273909007852818474030770699647647363015102118956737673941354217692696044969695308506436573142565573487583507037356944848039864382339216266670673567488871508925311154801
  c1 = 22322035275663237041646893770451933509324701913484303338076210603542612758956262869640822486470121149424485571361007421293675516338822195280313794991136048140918842471219840263536338886250492682739436410013436651161720725855484866690084788721349555662019879081501113222996123305533009325964377798892703161521852805956811219563883312896330156298621674684353919547558127920925706842808914762199011054955816534977675267395009575347820387073483928425066536361482774892370969520740304287456555508933372782327506569010772537497541764311429052216291198932092617792645253901478910801592878203564861118912045464959832566051361
  c2 = 18702010045187015556548691642394982835669262147230212731309938675226458555210425972429418449273410535387985931036711854265623905066805665751803269106880746769003478900791099590239513925449748814075904017471585572848473556490565450062664706449128415834787961947266259789785962922238701134079720414228414066193071495304612341052987455615930023536823801499269773357186087452747500840640419365011554421183037505653461286732740983702740822671148045619497667184586123657285604061875653909567822328914065337797733444640351518775487649819978262363617265797982843179630888729407238496650987720428708217115257989007867331698397
  e1 = 11187289
  e2 = 9647291
  s = egcd(e1, e2)
  s1 = s[1]
  s2 = s[2]
  # 求模反元素
  if s1<0:
    s1 = - s1
    c1 = invert(c1, n)
  elif s2<0:
    s2 = - s2
    c2 = invert(c2, n)

  m = pow(c1,s1,n)*pow(c2,s2,n) % n
  print(m)

if __name__ == '__main__':
  main()
```

运行得到

```
PSA3
D:\python38\python.exe D:/pycharm/venv/mima/PSA3.py
13040000448281994721293643679650728694052589818887496746545784530927147228703238333780127910 1
```

十进制转十六进制

十六进制转文本即可得到flag链接前面有

```
1 666c61677b343964393130373761316162636231346631613964353436633830626539656 67d
```

16进制转字符 | 字符转16进制 | 测试用例 | 清空结果 | 复制结果


```
1 flag{49d91077a1abcb14f1a9d546c80be9ef}
```


31.RSA2

查看题目

类型：dp+n+e+c = m dp泄露

RSA各题型脚本\dp+n+e+c = m

题目.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
e = 65537
n =
24825400785152624117772152669890180298583276617622160961225887737162058006043310153832803030521991869764361981420093067961210988553380133534844502375167047
84370730555447242806847332980515991676603036451831461614974853586336814921296688024020657977899055504895476451187872666019294297241331677684653096659061 13
dp =
90507449805234690464302513287951833069192517457305400462187725331868267505542197094355201669552856036483444630319693920705664292714809329037444021050365 7

c =
14042367097625269680753367358620940057566428210068411978420352712452118899640382659743688376604187906749428095741020195893573736038080184545382929399743341
41888387257517962617026220285872115603533628471910603065785105113809651621334726987130635926210289591670727814825626736830905905212142180711602876651807 51
```

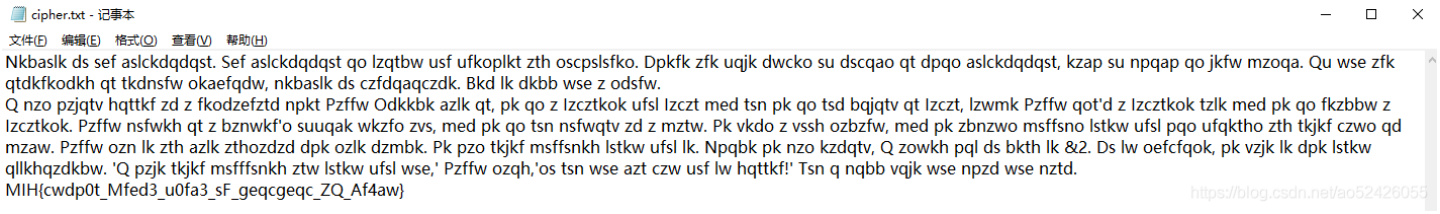
```python
import gmpy2 as gp

e = 65537
n = 24825400785152624117772152669890180298583276617622160961225887737162058006043310153832803030521991869764361981420093067961210988553380133534844502375167047843707305554472428068473329805159916766030364518314616149748535863368149212966880240206579778990555048954764511878726660192942972413316776846530966590611 3
dp = 90507449805234690464302513287951833069192517457305400462187725331868267505542197094355201669552856036483444630319693920705664292714809329037444021050365 7
c = 14042367097625269680753367358620940057566428210068411978420352712452118899640382659743688376604187906749428095741020195893573736038080184545382929399743341418883872575179626170262202858721156035336284719106030657851051138096516213347269871306359262102895916707278148256267368309059052121421807116028766518075 1

for i in range(1, e):  # 在范围(1,e)之间进行遍历
    if (dp * e - 1) % i == 0:
        if n % (((dp * e - 1) // i) + 1) == 0:  # 存在p，使得n能被p整除
            p = ((dp * e - 1) // i) + 1
            q = n // (((dp * e - 1) // i) + 1)
            phi = (q - 1) * (p - 1)  # 欧拉定理
            d = gp.invert(e, phi)  # 求模逆
            m = pow(c, d, n)  # 快速求幂取模运算

print(m)  # 10进制明文
print('------------')
print(hex(m)[2:])  # 16进制明文
print('------------')
print(bytes.fromhex(hex(m)[2:]))  # 16进制转文本
```

运行得到flag

```
RSA2
D:\python38\python.exe D:/pycharm/venv/mima/RSA2.py
367043495811078506691190575146963123133875122571015868069261652193574724658068848404048830993291
------------
666c61677b776f775f6c65616b696e675f64705f627265616b735f7273613f5f3938393234373433353032 7d
------------
b'flag{wow_leaking_dp_breaks_rsa?_98924743502}'

进程完成，退出码 0
```


32.[BJDCTF 2nd]Y1nglish-y1ng

查看题目



老规矩直接爆破

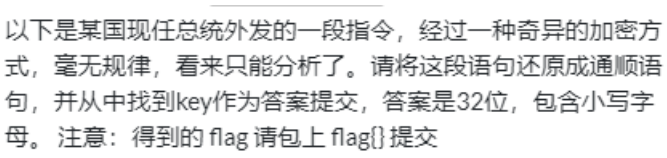
BJD{pyth0n_Brut3_f0rc3_oR_quipquip_AI_Cr4cy}

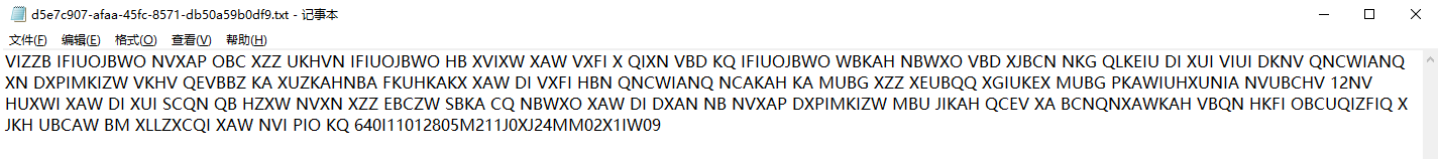
把最后一个字母改成k

BJD{pyth0n_Brut3_f0rc3_oR_quipquip_AI_Cr4ck}

33.世上无难事

查看题目





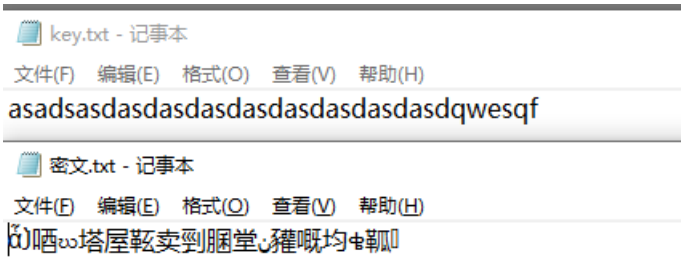
老规矩直接爆破

KEY IS 640E11012805F211B0AB24FF02A1ED09

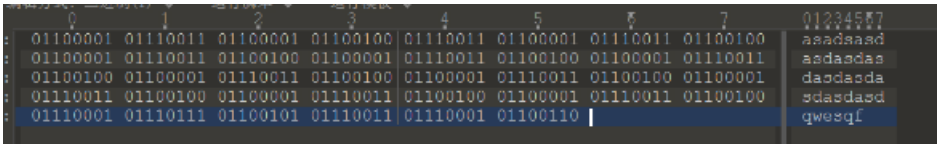
flag{640E11012805F211B0AB24FF02A1ED09}

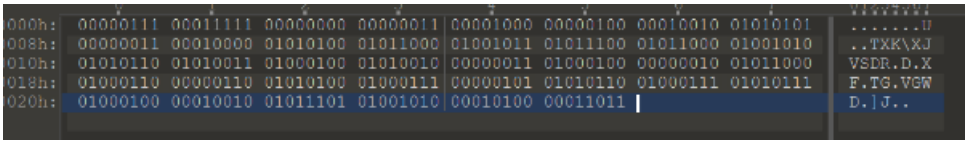
34.异性相吸

查看题目



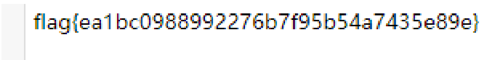
把这两个用010打开





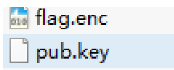
```python
a = '0110000101110011011000010110010001110011011000010111001101100100011000010111001101100100011000010111001101100100011000010111001101100100011000010111001101100100011000010111001101100100011000010111001101100100011000010111001101100100011100010111011101100101011100110111000101100110'
b = '0000011100011111000000000000000110000100000000100000100100101010100000001100010000010101000101100001001011010111000101100001001010010101100101001101000100010100100000001101000100000000100101100001000110000001100101010001000111000001010101011001000111010101110100010000010010010111010100101000010100000011011'
c = ''

for i in range(len(a)):
    if(a[i] == b[i]):
        c+='0'
    else:
        c+='1'
print(c)
```

运行得到
0110011001101100011000010110011101111011011001010110000100110001011000100110001100110000001110010011100000011100000011100100111001001100100011001000110111001101100110001000110111011001100011100100110101011000100011010100110100011000010011011100110100001100110011010101100101001110000011100101100101011111101
二转十六 十六转文



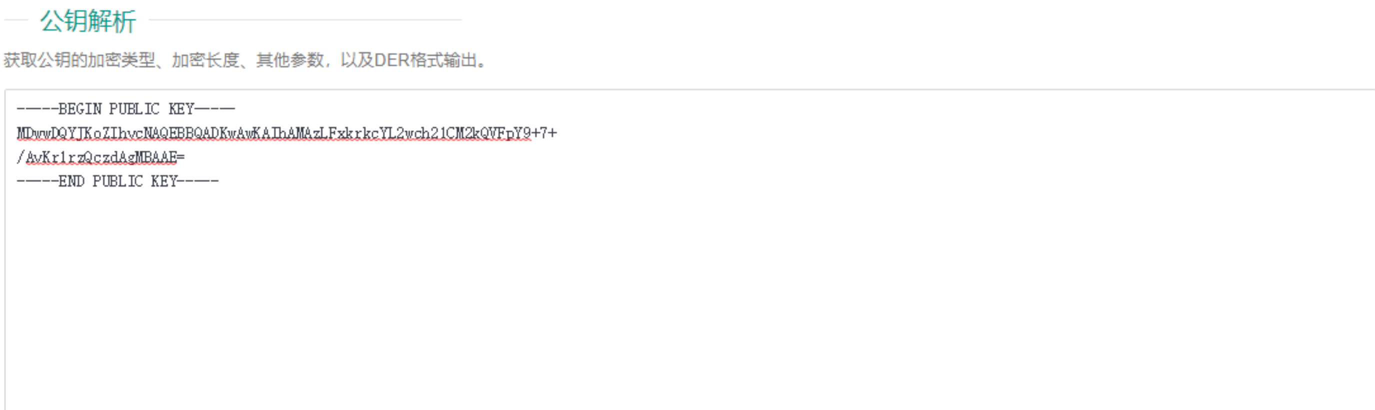
35.RSA

查看题目
类型：n+e+c+p+q= m + 公钥提取



把两个文件后缀都改成txt（我是把第一个文件改成了flagenc.txt）

明显pub.txt是一个公钥解析



解 析

详细信息

密钥类型	RSA
密钥强度	256
PN(e)	65537
PN(n)	869344822960481191906660620034948005889056560172030256172166540583783221035 17
DER格式	303c300d06092a864886f70d0101010500032b003028022100c0332c5c64ae47182f6c1c876d42336910545a58f7eefefc0bcaaf5af341ccdd0203010001

再用分解N

ences | Report results | Factor tables | Status | Downloads

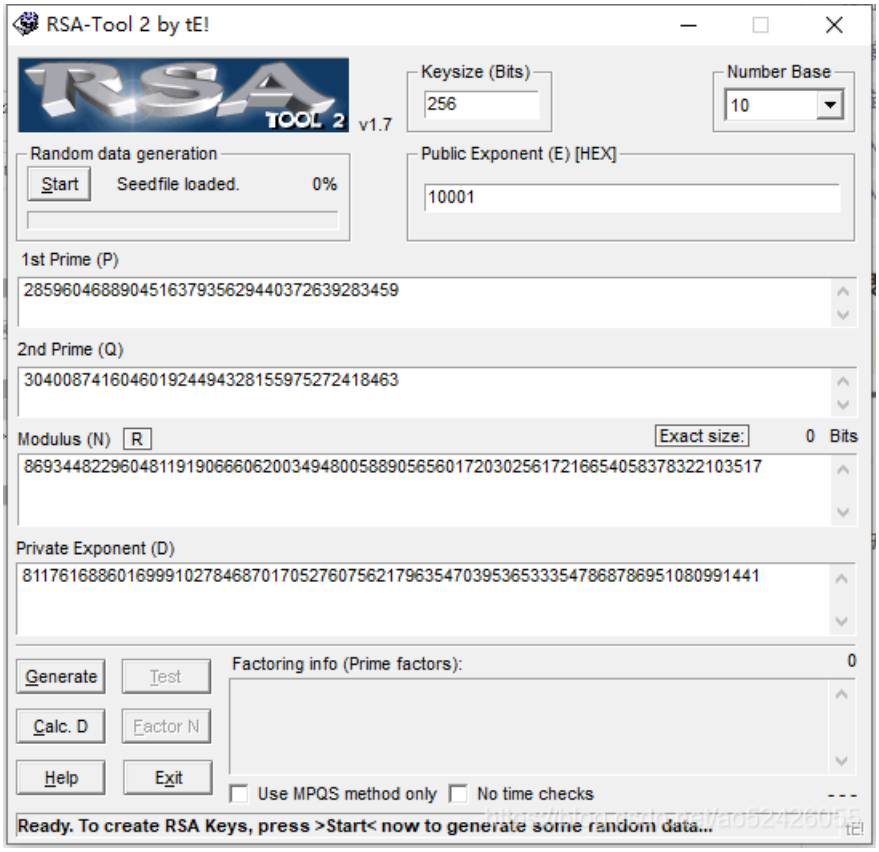
869344822960481191906660620034948005889056560172030256172166540583783221035 17 Factorize! (?)

Result:

= 285960468890451637935629440372639283459<39> · 304008741604601924494328155975272418463<39>

More information

得到q，p



明显q，p是十进制的得到D

写脚本

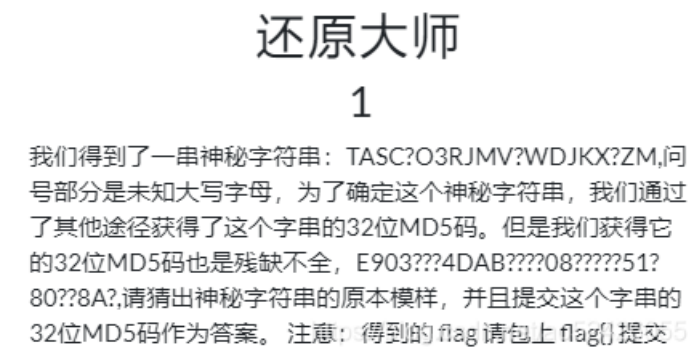
```
import rsa

e= 65537
n= 869344822960481191906660620034948005889056560172030256172166540583783221035 17
p= 285960468890451637935629440372639283459
q= 304008741604601924494328155975272418463
d= 81176168860169991027846870170527607562179635470395365333547868786951080991441

key = rsa.PrivateKey(n,e,d,q,p)          #在pkcs标准中,pkcs#1规定,私钥包含(n,e,d,p,q)

with open("D:\\ctfbisai\\buumima\\0eaf8d6c-3fe5-4549-9e81-94ac42535e7b\\flagenc.txt","rb") as f:  #以二进制读模式
, 读取密文
    f = f.read()
    print(rsa.decrypt(f,key))            # f:公钥加密结果  key:私钥
```

```
D:\python38\python.exe D:/pycharm/venv/mima/RSA4.py
b'flag{decrypt_256}\n'
```

运行就可以得到flag

36.还原大师

查看题目



这个就是md5爆破

python爆破脚本：

```
# -*- 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)       #輸出密文
```

运行得到flag{E9032994DABAC08080091151380478A2}

37.[GKCTF2020]汉字的秘密

下载后得到word，查看得到这些字符

王壮 夫工 王中 王夫 由由井 井人 夫中 夫夫 井王 土土 夫由 土夫 井中 士夫 王工 王人 土由 由口夫

很明显是当铺密码

上脚本

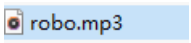
```
dh = '田口由中人工大土士王夫井羊壮'
ds = '00123455567899'

cip = '王壮 夫工 王中 王夫 由由井 井人 夫中 夫夫 井王 土土 夫由 土夫 井中 士夫 王工 王人 土由 由口夫'
s = ''
for i in cip:
 if i in dh:
  s += ds[dh.index(i)]
 else:
  s += ' '
#print(s)

ll = s.split(" ")
t = ''
for i in range(0,len(ll)):
 t += chr(int(ll[i])+i+1)
print('t=', t, '\t\tt.lower()=', t.lower())
```

运行得到flag

```
t.lower()= flag{you_are_good}
```

38.robomunication

查看题目



一个音频文件，直接就是用Audacity听

发现是摩斯密码，

听完整理一下就是... . .-... .-... — .-- ... .- - ... ... - ... . -.- . -.-- ... - ... ... -... — --- .-. -... . . .-.

摩斯密码解密即可得到

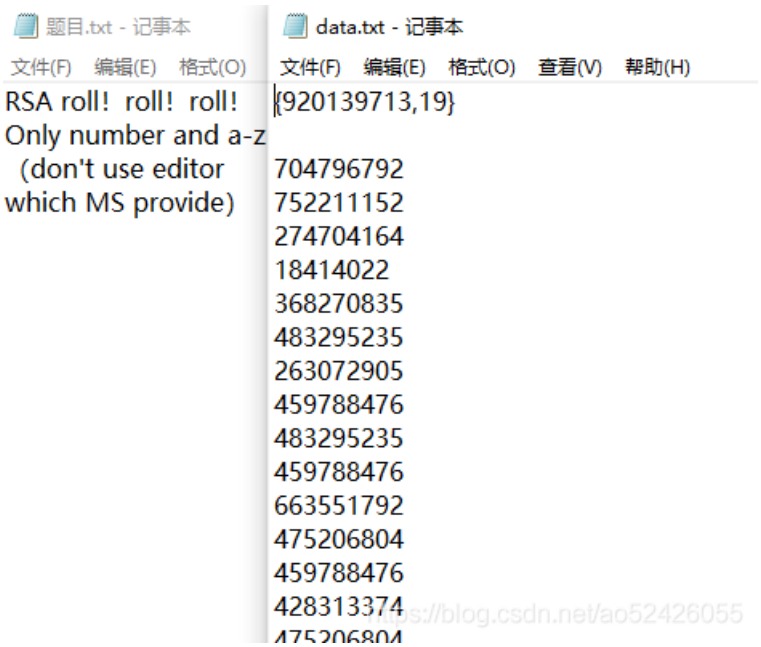
本明显了boopbeep是flag

小写提交不行，转换大写提交。

39.RSAroll

查看题目

类型：n+e+c+p+q= m + n分解



在线分解质因数分解920139713

然后直接上代码

```
import gmpy2
N,p,q,e=920139713,18443,49891,19
d=gmpy2.invert(e,(p-1)*(q-1))
result=[]

with open("D:\\pycharm\\venv\\mima\\RSAroll.txt","r") as f:
    for line in f.readlines():
        line=line.strip('\n')#去掉列表中每一个元素的换行符
        result.append(chr(pow(int(line),d,N)))

for i in result:
    print(i,end='')
```

```
open("D:\\pycharm\\venv\\mima\\RSAroll.txt"
```

这个是重新创的一个txt文本把

```
704796792
752211152
274704164
18414022
368270835
483295235
263072905
459788476
483295235
459788476
663551792
475206804
459788476
428313374
475206804
```

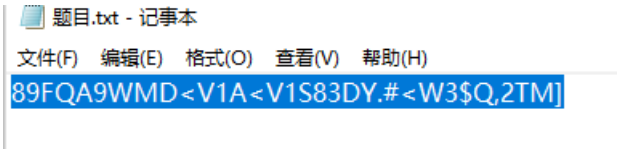
这些要分解的放在里面

脚本跑起来即可得到flag

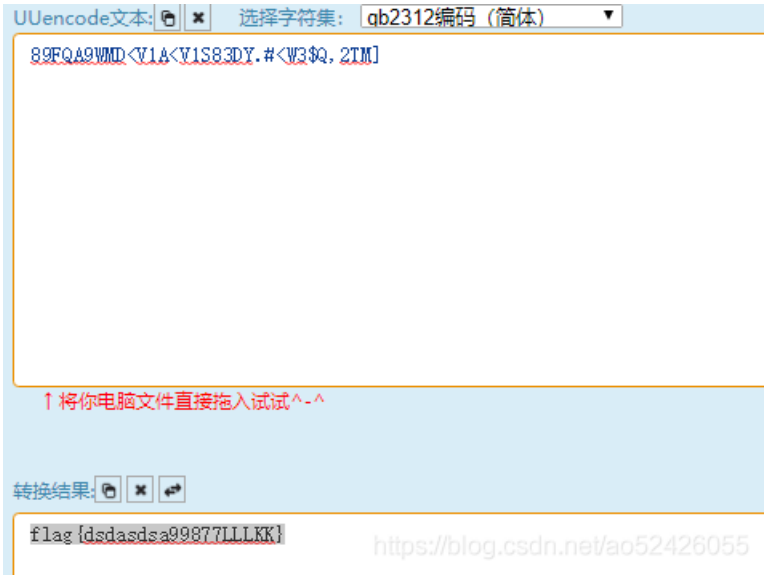
```
D:\python38\python.exe D:/pycharm/venv/mima/RSAroll.py
flag{13212je2ue28fy71w8u87y31r78eu1e2}
```

40.Unencode

查看题目



UUencode在线解密



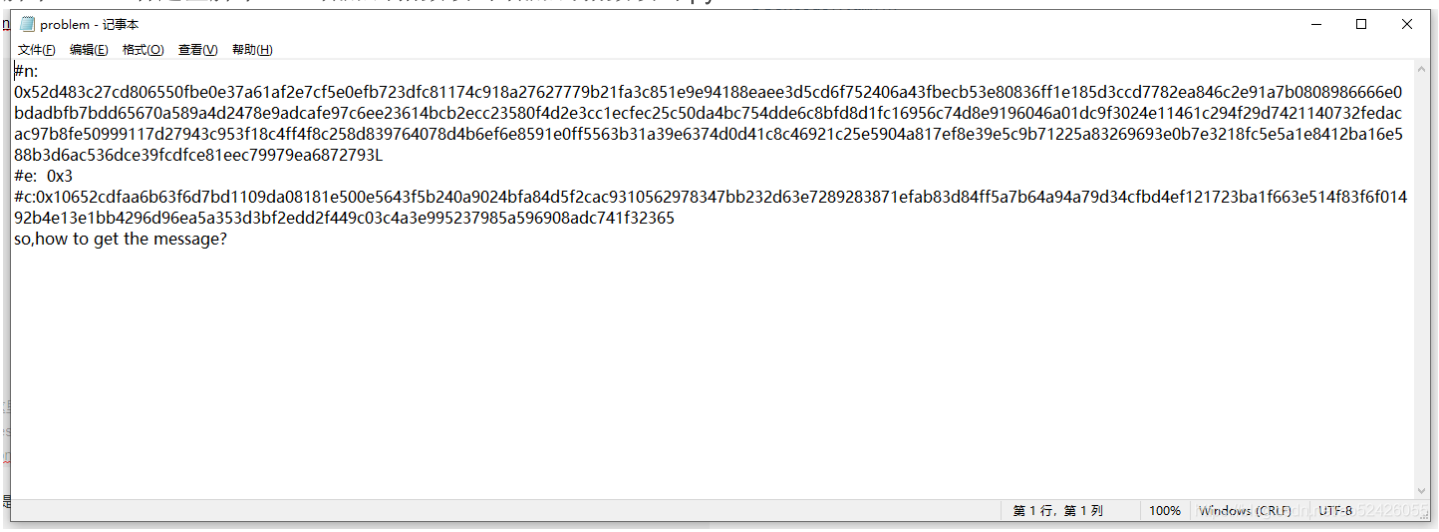
直接提交即可

41.Dangerous RSA

查看题目

类型：低加密指数攻击 e很小 n很大又不好分解

脚本：RSA各题型脚本\e=2-低加密指数攻击\低加密指数攻击.py



低加密指数攻击：

所谓低加密指数指的就是e非常小的情况下，通常为3。

这种题目通常有两种类型，一种直接爆破，另外一种是低指数广播攻击。

先介绍比较简单的情况。假设e=3, e很小，但是n很大。

回顾RSA加密公式： C=M^e % n (C密文，M明文)

```
'''
当M^e < n 时，
 C = M^e ，所以对C开方就能得到M
'''
from gmpy2 import iroot
import libnum
n = 0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c851e9e94188eaee3d5cd6f752406a43fbec
b53e80836ff1e185d3ccd7782ea846c2e91a7b0808986666e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ecc23580f4d2
e3cc1ecfec25c50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f3024e11461c294f29d7421140732fedacac97b8fe50999117d
27943c953f18c4ff4f8c258d839764078d4b6ef6e8591e0ff5563b31a39e6374d0d41c8c46921c25e5904a817ef8e39e5c9b71225a832696
93e0b7e3218fc5e5a1e8412ba16e588b3d6ac536dce39fcdfce81eec79979ea6872793

c = 0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978347bb232d63e7289283871efab83d84ff
5a7b64a94a79d34cfbd4ef121723ba1f663e514f83f6f01492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e995237985a596908ad
c741f32365

k = 0
while 1:
    res=iroot(c+k*n,3)
    if(res[1]==True):
        print(libnum.n2s(int(res[0])))
        break
    k=k+1


'''

第二种写法
当M^e ＞ n 时，此时用爆破的方法
 假设我们   M^e / n 商 k 余数为c，
 所以M^e  = k*n + C，对K进行爆破，只要k满足 k*n + C能够开方就可以
'''
'''
import gmpy2
from libnum import*
n = 0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c851e9e94188eaee3d5cd6f752406a43fbec
b53e80836ff1e185d3ccd7782ea846c2e91a7b0808986666e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ecc23580f4d2
e3cc1ecfec25c50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f3024e11461c294f29d7421140732fedacac97b8fe50999117d
27943c953f18c4ff4f8c258d839764078d4b6ef6e8591e0ff5563b31a39e6374d0d41c8c46921c25e5904a817ef8e39e5c9b71225a832696
93e0b7e3218fc5e5a1e8412ba16e588b3d6ac536dce39fcdfce81eec79979ea6872793
c = 0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978347bb232d63e7289283871efab83d84ff
5a7b64a94a79d34cfbd4ef121723ba1f663e514f83f6f01492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e995237985a596908ad
c741f32365

i = 0
while 1:
    if(gmpy2.iroot(c+i*n,3)[1]==1):     #开根号
        print(gmpy2.iroot(c+i*n,3))
        break
    i=i+1

'''
```

运行得到flag

```
D:\python38\python.exe "D:/pycharm/venv/mima/Dangerous RSA.py"
b'flag{25df8caf006ee5db94d48144c33b2c3b}'
```

42.Cipher

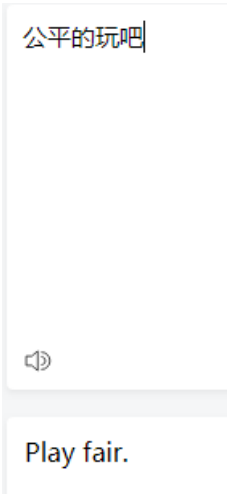
查看题目

Cipher

1

还能提示什么呢？公平的玩吧（密钥自己找）
Dncnoqqfliqrpgeklwmppu 注意：得到的 flag 请包上 flag{} 提交, flag{小写字母}

这道题我也一点思路没有，看了大佬的wp知道了



playfair也是一个加密方式，嗯然后找了一个在线解密playfair在线解密

The Playfair cipher is a digraph substitution cipher. It employs a table where one letter of the alphabet is omitted, and the letters are arranged in a 5x5 gri
Below is an unkeyed grid.

A	B	C	D	E
F	G	H	I	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

To encode a message, one breaks it into two-letter chunks. Repeated letters in the same chunk are usually separated by an X. The message, number of letters in the message, it was padded with a spare X. Next, you take your letter pairs and look at their positions in the grid.

"HE" forms two corners of a rectangle. The other letters in the rectangle are C and K. You start with the H and slide over to underneath the E a are "KC". "LX" becomes "NV" in the same way.

"LO" are in the same row. In this instance, you just slide the characters one position to the right, resulting in "MP". The same happens for "ON", resulting we scroll around back to the left side and get A.

"ND" are in a rectangle form and beomes "OC". "AL" are both in the same column, so we just move down one spot. "AL" is changed into "FQ". "LX" is an

The resulting message is now "KC NV MP PO AB OC FQ NV" or "KCNVMPPOABOCFQNV" if you remove the spaces.

This encoder will do all of the lookups for you, but you still need to do a few things yourself.

1. Manually break apart double letters with X (or any other) characters. Some people break apart all doubles, others break all doubles that happen in t
2. Manually make the message length even by adding an X or whatever letter you want. If you don't, the encoder will automatically add an X for you.

All non-letters are ignored and not encoded. The one letter that you select to share a square in the cipher is translated. Numbers, spaces, and punctuatic down and right one square ("LL" becomes "RR") where as traditional Playfair ciphers will automatically insert an X for you.

This particular cipher was used by the future U.S. President, John F. Kennedy, Sr. He sent a message about a boat going down.

Decrypt ▾

Translate the letter J ▾ into I ▾

☑ Encode double letters (down and right one spot)

Alphabet Key: playfair - Show Keymaker

Tableau Used:

P	L	A	Y	F
I	R	B	C	D
E	G	H	K	M
N	O	Q	S	T
U	V	W	X	Z

Your message:

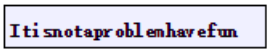
Dncnoqqfliqrpgeklwmppu

Add Spaces - Adds a space after every other letter (only A-Z count) so you can see the letter pairs.
Only Letters - Removes all non-letters from the text.

This is your encoded or decoded text:

Itisnotaproblemhavefun

再把得到的

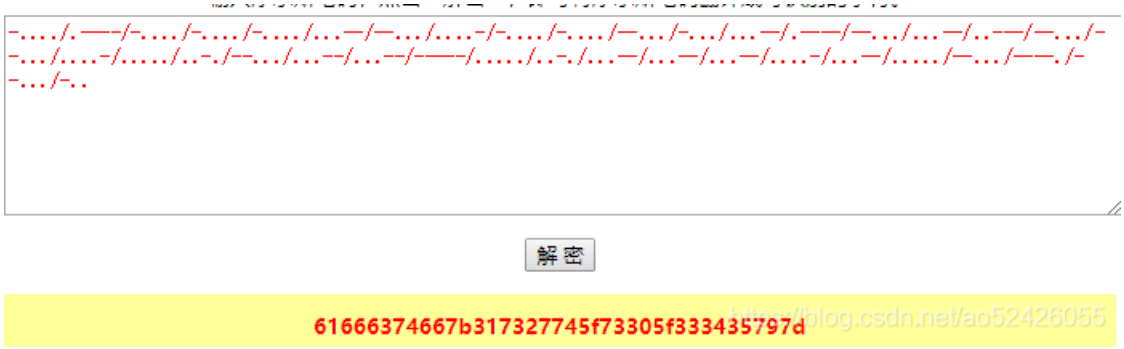


这个转换为小写即可提交。

43.[AFCTF2018]Morse

查看题目

很明显的一个摩斯密码，摩斯密码在线解密。



得到这些东西，我是先提交了一下发现错误

然后直接根据 十六进制转换文本

得到afctf{1s’t_s0_345y}

题目上说的很明白要flag{}提交

所以最后的flag就是 flag{1s’t_s0_345y}

44.[HDCTF2019]basic rsa

下载得到一个py

```python
import gmpy2
from Crypto.Util.number import *
from binascii import a2b_hex,b2a_hex

flag = "******************"

p = 26224880018227704065019205543990658047 9
q = 26285499423932282854792559548751991555 1

e = 65533
n = p*q


c = pow(int(b2a_hex(flag),16),e,n)
print c


# 27565231154623519221597938803435789010285480123476977081867877
```

类型：n+e+c+p+q= m
注释中就是该脚本生成的c
使用脚本RSA各题型脚本\n+e+c+p+q= m
直接套用脚本

```python
import random
from binascii import a2b_hex,b2a_hex
p = 26224880018227704065019205543990658047 9
q = 26285499423932282854792559548751991555 1
n = p * q
def multiplicative_inversr(a,b):
    x = 0
    y = 1
    lx = 1
    ly = 0
    oa = a
    ob = b
    while b != 0:
        q = a // b
        (a, b) = (b, a % b)
        (x, lx) = ((lx - (q * x)), x)
        (y, ly) = ((ly - (q * y)), y)
    if lx < 0:
        lx += ob
    if ly < 0:
        ly += oa
    return lx
def gcd(a,b):
    while b != 0:
        a, b = b, a % b
    return a
def generate_keypair(p,q):
    n = p * q
    phi = (p - 1) * (q -1)
    e = 65533
    g = gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = gcd(e, phi)
    d = multiplicative_inversr(e, phi)
    return ((e,n),(d,n))
def encrypt(pk, plaintext):
    key, n = pk[0]
    print(b2a_hex(plaintext.encode()))
    cipher = pow(int(b2a_hex(plaintext.encode()),16), key , n)
    return cipher
def decrypt(pk, cipher):
    key, n = pk[1]
    cipher = pow(cipher, key ,n)
    cipher = a2b_hex(hex(cipher).split('0x')[1])
    return cipher
pk = generate_keypair(p,q)
cipher = 27565231154623519221597938803435789010285480123476977081867877272451638645710
plaintext = decrypt(pk, cipher)
print(plaintext)
```

运行得到flag 提交即可

```
D:\python38\python.exe "D:/pycharm/venv/mima/basic rsa.py"
b'flag{B4by_Rs4}'
```

解题思路

首先这道题题目是达芬奇密码，百度之后发现这是一部电影，当时也没想的去看一下电影的简介什么的，后面加buuctf关键字，也没有找到相应的wp。果断google，找到大佬的wp，发现在电影简介中会提到——斐波那契数列。

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309
1
对比蒙娜丽莎中的数字列，发现数值一样，但是进行了位移。

之后对比，题目中给到的两个数列的长度都是32，并且flag也是32位，可以推测，神秘数列是通过flag位移后得出的，而位移的规则是斐波那契数列的位移。

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309

1 233 3 2584 1346269 144 5 196418 21 1597 610 377 10946 89 514229 987 8 55 6765 2178309 121393 317811 46368 4181 1 832040 2 28657 75025 34 13 17711

369688538821167255473421769522 86
1
2
3
4
5
6
7
规则如下：

第零位1还是1，没有位移。

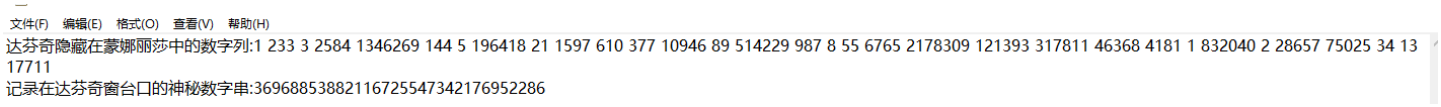
第一位233是斐波那契数列的第十二位（以0开始算），因此下面神秘数字串的第一位的6是原本flag的第十二位。

第二位3是斐波那契数列的第三位，因此下面神秘数字串的第二位的9是原本flag的第三位。

以此类推...，写出如下脚本。

45.达芬奇密码

查看题目



当时第一想法这都什么东西。果断google，找到大佬的wp，发现在电影简介中会提到——斐波那契数列。

```
达芬奇隐藏在蒙娜丽莎中的数字列:1 233 3 2584 1346269 144 5 196418 21 1597 610 377 10946 89 514229 987 8 55 6765 2178309 121393 317811 46368 4181 1 832040 2 28657 75025 34 13 17711
记录在达芬奇窗台口的神秘数字串:369688538821167255473421769522 86
```

对比蒙娜丽莎中的数字列，发现数值一样，但是进行了位移。

之后对比，题目中给到的两个数列的长度都是32，并且flag也是32位，可以推测，神秘数列是通过flag位移后得出的，而位移的规则是斐波那契数列的位移。

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 3
17811 514229 832040 1346269 2178309

1 233 3 2584 1346269 144 5 196418 21 1597 610 377 10946 89 514229 987 8 55 6765 2178309 121393 317811 46368 4181
 1 832040 2 28657 75025 34 13 17711




3696885388211672554734217695228 6
```

第零位1还是1，没有位移。

第一位233是斐波那契数列的第十二位（以0开始算），因此下面神秘数字串的第一位的6是原本flag的第十二位。

第二位3是斐波那契数列的第三位，因此下面神秘数字串的第二位的9是原本flag的第三位。

```
fb = '1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 19
6418 317811 514229 832040 1346269 2178309'

t = '1 233 3 2584 1346269 144 5 196418 21 1597 610 377 10946 89 514229 987 8 55 6765 2178309 121393 317811 46368
 4181 1 832040 2 28657 75025 34 13 17711 '

m = '3696885388211672554734217695228 6'
s = 'a' * 32
s = list(s)
fb = fb.split(' ')
t = t.split(' ')


for i in range(32):
    s[fb.index(t[i])] = m[i]
for i in range(32):
    print(s[i], end='')
```

用大佬的代码运行得到
7a995588256861228614165223347687
输出结果中还存在a，是因为斐波那契数列中存在两个1，而在index()找位置的时候，是从前往后找的，因此两次的1会覆盖掉。
所以要将m中t的第二次出现1的位置上的数替换给a，然后复原被覆盖的值.
flag包裹提交即可
然后我有动了动这腐朽的手找到另一个脚本

```
#主要思路是遍历fakefibbo，然后找出对应哪一位，如fb中第二个数对应rb中第五个
#然后cipher对应fakebibbo,因此cipher中第二个数对应flag第五个数
#因此有reslut[4]=cipher[1],依次类推
realfibbo = '1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 12
1393 196418 317811 514229 832040 1346269 2178309'
fakefibbo = '1 233 3 2584 1346269 144 5 196418 21 1597 610 377 10946 89 514229 987 8 55 6765 2178309 121393 3178
11 46368 4181 1 832040 2 28657 75025 34 13 17711'
cipher = '3696885388211672554734217695228 6'
realfibbo = realfibbo.split(' ')
fakefibbo = fakefibbo.split(' ')
result = ['a']*32
for i in range(len(cipher)):
#这里要考虑到第二个1(fb[24])寻找的时候，会找到1123中第一个数，也就是index=0，而我们希望他找到第二个数，也就是index=1
    if(i == 24):
        index = 1
    else:
        index = realfibbo.index(fakefibbo[i])
    result[index] = cipher[i]
for i in result:
    print(i,end='')
```

```
D:\python38\python.exe D:/pycharm/venv/mima/feibonaqie.py
379955882568612286141652233476 87
```

这个flag就可以直接提交

46.ras2

查看题目

```
N = 10199180977755325347027675139926474013115768232925267350179215450700615843443200914199536724196252570595004625340018888465826249653470643879151507188586089755273665689956691573129722581725063987364337631010399217064690655724283289391490205358108750251278730332274778042021088485216658671763655905815254497947 1
e = 46731919563265721307105180410302518676676135509737992912625092976849075262192092549323082367518264378630543338219025744820916471913696072050291990620486581719410354385121760761374229374847695148230596005409978383369740305816082770283909611956355972181848077519920922059268376958811713365106925235218265173085

import hashlib
flag = "flag{" + hashlib.md5(hex(d)).hexdigest() + "}"
```

公开的轮子

求d 这里要将破解脚本和rsa-wiener-attack的py文件放在同一个目录下

```
import  RSAwienerHacker
n = 10199180977755325347027675139926474013115768232925267350179215450700615843443200914199536724196252570595004625340018888465826249653470643879151507188586089755273665689956691573129722581725063987364337631010399217064690655724283289391490205358108750251278730332274778042021088485216658671763655905815254497947 1
e = 46731919563265721307105180410302518676676135509737992912625092976849075262192092549323082367518264378630543338219025744820916471913696072050291990620486581719410354385121760761374229374847695148230596005409978383369740305816082770283909611956355972181848077519920922059268376958811713365106925235218265173085

d =  RSAwienerHacker.hack_RSA(e,n)
if d:
    print(d)
```

这个python要用python2去运行，如果在py3中进行，会报错提示你要先将d进行编码，然后一直出错

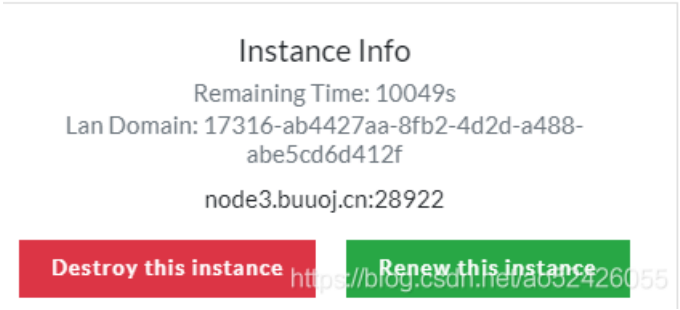
```
import hashlib
d = 89207589954145871528294265585800256573573287458397476937395918202813
flag = "flag{" + hashlib.md5(hex(d)).hexdigest() + "}"
print flag
```

再把d进行md5哈希

得到 flag{47bf28da384590448e0b0d23909a25a4}

47.[BJDCTF 2nd]rsa0

查看题目



这个很明显需要nc一下

打开kali

这个直接写一个简单python的脚本就好了

```
import gmpy2

a = 178647949799715707856623795451626188047662627873436920427027582698838898805865328476277394977103773791338089
606314497833209886460481925702159120449384 9524
b = -136211411695969353402663153409850392241363447180749338895339494651406777458233643836815789381841004372723457
984031789555336466519007467751314134348462 38
p = (a+b)//2
q = (a-b)//2

e = 10047043
c = 694380029171269225981005630334981173130598467618102765154725834433153304862497872348838178335812078755557002
361926207572212363213457583281162285880603699151107567259937930154682702710165114527669211671611470129167640259 2
556420601432217625249225791131983766380297718085093016971344969740035278435213058503850 9
n = p*q
phi = (p-1)*(q-1)

d = gmpy2.invert(e,phi)
m = pow(c,d,n)

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

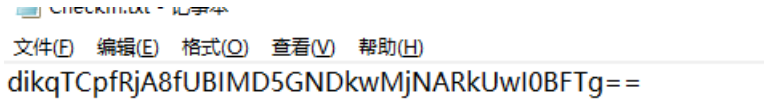
运行就可得到flag

```
1433763655511768098193373431925864895093024246686511175466230276537479388167205729379830241179341760026 6
0x666c61677b63396538326139612d646236392d343535622d383438392d303365643436366265323263 7d0a
b'flag{c9e82a9a-db69-455b-8489-03ed466be22c}\n'
```

这个是动态的，想直接拿我的flag提交不可能的，别问我怎么知道的。

48.[GXYCTF2019]CheckIn

查看题目



刚开始我以为是base64，结果解密出来一堆乱码
然后又试了一下凯撒也不行
那就只有试一下ROT-N了
为rot-47加密：flag{Y0u_kNow_much_about_Rot}

49.RSA5

看题目

```
m = xxxxxxxx
e = 65537
========== n c ==========
n = 204749188940517785333052623456018809280882844711218237540497253540724771558737788480550738433458206978866410
868426124865412501839659660015913420315629535617933323416413343028479961084174663606881398665051796895165893056 3
690213721018562465085490678003720441220630994919908000557692277577372243886376211775042932758579209344742398000 2
401200613302943834212820909269713876683465817369158585822294675056978970612202885426436071950214538262921077409 0
761604174366998361388011626213148456087968702068347041167077631698473872233078289085709449844169730194275297900 2
908976626494907803866952346524383767526385806285473908363420 7
c = 974463908243330865728978769213595400782053398596897741316275722596415018912929508637393850919224969271766388
710025195039896961956062895570062146947736340342927974992616678893372744261954172873490878805483241196345881721 1
```

```
64078651156067119957816422768524442025688079462656755605982104174001635345874022133045402344010045961111720151990412034477755851802769069309069018738541854130183692204758761427121279982002993939745343695671900015296790637464880337375511536424796890996526681200633086841036320395847725935744757993013352804650575068136129295591306569213300156333650910795946800820067494143364885842896291126137320

n = 20918819960648891349438263046954902210959146407860980742165930253781318759285692492511475263234242002509419079545644051755251311392635763412553499744506421566074721268822337321637265942226790343839856182100575539845358877493718334237585821263388181126545189723429262149630651289446553402190531135520836104217160268349688525168375213462570213612845898989694324269410202496871688649978370284661017399056903931840656757330859626183773396574056413017367606446540199973155630466239453637232936904063706551160650295031273385619470740593510267285957905801566362502262757750629162937373721291789527659531499435235261620309759
c = 15819636201971185538694880505120469332582151856714070824521803121848292387556864177196229718923770810072104155432038682511434979353089791861087415144087855679134833968978174587265438830935676003252045961566493059303525752740394254708363550026911458644357553338211339692669515451580527459382525743013276968223471150536140524230288355325092206413787608006933515426338607022257726389305010215714159073481282696812241783002482726897053089112822086854596682005070571834206629591139560775847817379832547887030482756989214270298842825574683343996778499623421961408644039891621177382062461836658149387831229099300828020318551
n = 25033254625906757272369609119214202033162128625171246436639570615263949157363273213121556825878737923265290579551873824374870957467163989542063489416636713654642486717219231225074115269684119428086352535471683359486248203644461465935500517901513233739152882943010177276545128308412934555830087776128355125932914846459470221102007666912211992310538890654396487111705385730502843589727289829692152177134753098649781412247065660637826282055169991824099110916576856188876975621376606634258927784025787142263367152947108720757222446686415627479703666031871635656314282727051189190889008763055811680040315277078928068816491
c = 41853085294168740058312307810140924071984513859556773996685018339026234783956692794048839907251843327091524433725837010761987866352917393567708572867021071567300200043589556225110614106610589826220551997368208082038414467963052843946517144309186903894869205608346723161581464531837894121409390290293247560353580817544266451600332629243302486752161082709801570497054886202634851294809528147640028652800191851276624493183242793832777664162581422751439235321687984130110282715430852490290489974522125031117423023020654010514580665853953604684474606586729528516435471938227752183878536234536380254923891222045075559088621
n = 21206968097314131007183427944486801953583151151443627943113736996776787181111063957960698092696800555044199156765677935373149598221184792286812213294617749834607696302116136745662816658117055427803315230042700695125718401646810484873064775005221089174056824724922160855810527236751389605017579545235876864998419873065217294820244730785120525126565815560229001887622837549118168081685183371092395128598125004730268910276024806808565802081366898904032509920453785997056150497645234925528883879419642189109649009132381586673390027614766605038951015853086721168018787523459264932165046816881682774229243688581614306480751
c = 45210380110447584418911284684672330884938857085058898570851991115477809059713612615028904189345412667446814139347266233735036171221269486731162297044070772794111326383235717314177585522797374257108897459347630208411177062576422283836627755956088704294885989213855147268065451781491660927974836558061071225985667774051847708653159223310717547006829190360750579943293198966370747701790461142621377023839700574373038608003195569415846655847559975194024503916762912657678402448234845286831341747154295677828556777943594026714067990668653186246762723840100345910163719129720942247038812180253656976141445761825834355061 3

n = 22822039733049388110936778173014765663663303811791283234361230649775805923902173438553927805407463106104699773994158375704033093471761387799852168337898526980521753614307899669015931387819927421875316304591521901592823814417756447695701045846773508629371397013053684553042185725059996791532391626429712416994990889693732805181947970071429309599614973772736556299404246424791660679253884940021728846906344198854779191951739719342908761330661910477119933428550774242910420952496929605686154799487839923424336353747442153571678064520763149793294360787821751703543288696726923909670396821551053048035619499706391118145067
c = 15406498580761780108625891878008526815145372096234083936681442225155097299264808624358826686906535594853622687379268969468433072388149786607395396424104318820879443743112358706546753935215756078345959375299650718555759698887852318017597503074317356745122514481807843745626429797861463012940172797612589031686718185390345389295851075279278516147076602270178540690147808314172798987497259330037810328523464851895621851859027823681655934104713689539848047163088666896473665500158179046196538210778897730209572708430067658411755959866033531700460551556380993982706171848970460224304996455600503982223448904878212849412357

n = 21574139855341432908474064784318462018475296809327285532337706940126942575349507668289214078026102682252713757030815530931088232140637915184822898467801973298211395079747637802602903096008849208119598429255405839670856700
```

```
0848765317877441480914852329276375776405689784571404635852204097622600656222714808541872252335877037561388406257
1817152787666528247863762622492749604671939619566909748536797952491587510784222965803675062197197387621599659588
7780618746107068907129094818194956125414431077694333485977512165018624584603172050794498783848972312789722341680
2436021278671237227993686791944711422345000479751187704426369
c = 2036685615071030512458306537529766181979524223837648526495118533699608374460459341898333628518549119742601855
9503144465212328846149187902109602820369413668320344169298706956351302600186143572211798555990969267090734756359
4578265880806540396777223906955491026286843168637367593400342814725694366078337030937104035993569672959361347287
8941430271868468567729830583289197167029822221428488481177684999661758830530148308542854726733707099876741254022
2591150819684225313435590126386112150065024029674670296759422440165022016878053714165448921501914212228430811628
4129004257364769474080721001708734051264841350424152506027932

n = 2536022741266661249010216113117458481924093180319644848122430525058384143958100852853593081416733838198376499
1296575637231916547647970573758269411168219302370541684789125112505021148506809643081950237623703181025696585998
8044695691322012183660424636496897073045557400768745943787342548267386564625462143150176113656264450210023925571
9459614057092766319907316021981042875285280556500504861598376122796004152594863061549475140054089075900837477589
5311548612486548672063382055913506344094252803140295195855763083350377511201071560427811432552899377108123353524
7118481765852273252404963430792898948219539473312462979849137
c = 1989277252465145234102759561948273435624343567159239817268037998150275969578408790066908991998770567589994565
8648623800090272599154590123082189645021800958076861518397325439521139995652026377132368232502108620033400051346
1277576986238861426217934232257492402865116665560917878516839780175069833100735243982872797376800917873335475382
3992060776108098824363954757081836378867324958278301547568210998471529316313732443986283857446010879371417260367
2477766831356411304446881998674779501188163600664488032943639694828698984739492200699684462748922883550002652913
5182293229450408190641333503145363786945237047933961690651 79

n = 2272685524463235602915969175345182216333151923754763993877951775149649871317458893556657616732957649479021936
0727877166074136496129927296296996970048082870488804456564986667129388136556137013346228118981936899510687589585
2865171513230482931502570368474754240443781091681794122878893405963947552577049380061626776565815093754711025462
6135574825186904800360052003465626452193180865103852413418573292957038470591856398206568414576642796250226152248
1994191989820110575981906998431553107525542001187655703534683231777988419268338249547641335718393312295800044734
5347616927994034694979540628978562990312574547359458674911 91
c = 6040119795175856407541082360023532204614723858688636724822712717572759793960246341800308149739809871234313049
6297329347975697810530006861856663748339784032905250725987740017313502447445907727957010651295618981165764999841
8592066127112366535613271919366547423559688423910803060588277786885612237822268114057051918032128697694715404227
2622411303981011302586225630859892731724640574658125478287115198406253847367979883768000812605395482952698689604
4477719478947595442185921480652637868335673233200662100621025061500895729605305665864693122952557361871523165300
2060703256603530955927780377673953603292313313228236100600 06
n = 2329733379144305329736300078683533609525229081846195005454265832748450740659463278571276745995891794309552259
4228205423428207345128899745800927319147257669773812669542782839237744305180098276578841929496345963997512244219
3767017876160462353971393818948374355626625910607684769973335387480652940331416105022523252928018168122689341713
6193439995154862726779140108970393738901258658108022331306015945623885708074069952866641130302993480701121495398
4169785844714159627792016926490955282697877141614638806397689306795328344778478692084754216753425842557818899467
9451026467763426551676553842248605040860831478412522327609 41
c = 5418120301208378713115889465579964257871814114515046096090960159737859076829258516920361577853903925954198406
8437573036875578483023022002292959169024302057378436018067007382347566985757086124249284804408687391200758886816
7206220652915656642127661110780291741899362502969062719681383032636987424977619239603300605876865967515719079797
1159105786535627878990193101399459049580248824178337363048947654334894762345753567552751472565773870228733489069
0014963494074710451385015411810699113707264330862028466310828305224575094522899538780343212884215225154929269894
7407663643895853432650029352092018372834457054271102816934

n = 2887366790471568272298723429349320030697694789871125506412511593366696867874259885872243142621891446290352159
6341771131695619382266194233561677824357379805303885993804266436810606263022097900266975250431575654686915049693
0914678648205127670707132677089938998990111561067661789067003361117128033621130396135486729370533978756631447940
1808701773194908779489490373768238391617326742140340814096771307102600187473348729500750106887104464917061570989
1451856792232315526696220161842742664778581287321318748202431466508948902745314372299799561625186955234673012098
2109197458798822685126569317143267823352110895768973105914 91
c = 9919880463786836684987957979091527477471444996392375244075527841865509160181666543016317634963512437510324198
7024163228413774894170295723884744500758014629968252446575302861074281863541728367165028176090705909297692619323
2427535328993930253644031062869834924487206400570064452022372767095078792429600429688303297894120088336265399335
1638545860207179022472492671256630427228461852668118035317021428675954874947015197745916918197725121122236369 38
```

```
274153398302346225591392469280624938744901662986582331640236601765784416691984668349785184238805828385621990053556742710360386995506619342550138525532209790153140210388 3869

n = 223246859475396537224999324694096075330654191573478139619580756890476904652664043841994836839085947873124455281596355278339044758018903814556538072655012173287578713527312930003034382053158167926639175790666748423077438452617710323639285688446698957680925156583287562292458370252617442606148607469979315035487885099838680383497202253057309855762936752690737090223507008365100540676417537132129999543070225244958855833617073785137421625663390101343549078637332059218450389182244639037898418814008140745872617202838797601220709014665171182654228634203769215367348455021002514608724991222366868321895496980207371766830 19

c = 149152705020329498988282924856039518480497727774712614310395721916462418752844104783735126358044068647476738046400554026462791012648312993066834409581454759211506105784347013149807506042039511100861902719903701992570123666016656306824568397578776280435952016470169169091648259102613858270555824686949616275978087843713796082300004398822730300387641050312137016330371160335943076453933759786686250845152815828510325181005874187968787521838416028250617270661335947765721542073481604939333959375548921858879660706026189790523345326867141161063104734045948793747951193345036946221379573893301900147180315760779173853846 7

n = 276467464237590201110078286532640279992578476456661299077890260545943936488002361170467691127626417788656208924434231001896193275858113848835154249187527495596275536377850373596398011252132561630084319425937279319318981997275527686267756184798330291012496925737160307066957025109822835557408510470226724857434324646477728823142151761147322574972402841640169140186890445572189203002622346528406324060672733752693010084098601931808223667358772882057833143261022637565037867361223213483200319500121449058695562040174305936560528679394936331634995802422247634043388070225101362171877790849179961716027370365649910367242 99

c = 219915241289572605360437712848549203931058081267001282221258567755068857219711931093613159611291908146746471364648870878939906608949616128382050864010188854576674889118986542702355619801111746033237212809111974882865852693568495792630434563163194764958886962193442198665168611876541805092478812512512789193462671299047392773862892400394384575124331135655943513831009934023397457082184699737734388823763306805326430395849935770213817533387235486307008892410920611669932693018165569417445885810825749609388627231235840912644654685819620931663346297596334834498661789016450371769203650109994771872404185770230172934013971

n = 205454874058169287317389883744750126868279337097897843918557068351362702709334012030193291369376508783861171877765306393425721232371880539786226972825214739179782828304321611532212161941698796695419988406913830254872208508720754360643084999249585179797279544029656121960814043416515173263640415192501250364248226343542687738954656989208834392229965812263585958739939766046998306139323207205541300116712979444335150471805654844951910038875998912890379820102163578310783281590289532220569181893658407115886710933330131174540343136228550827958131223385624462230412111922770892250783246821080338430239035501728919596735 51

c = 142274391881910294612504766927905396546191998848731942911441455797537630868890802814081715720557980405978380764130557738572475853013851497296220906223057610740614240260348437562607734519088309409763601977137786633953151196513665056741236388918315961618844926375247532866324531105998833799604735926328883743630558884804457293775942446658687028051242433680706472989451584055240475687959069879704633333644546512044508758762174390662427962176796347723788029591097144005161837183232672738247365401685459464444375862992141104247381599573883507859993485351715535693730882515527123912883652952676656913577196160116136287721 75

n = 273597277115842772348971577240558527940192168452297989386558142694600463843535681385985677553925596534609494455787912004079679814221893925184476246127025167239954677406727534829100396255196464874205321542462025699934544839880527859277704966828155831287177397993134309780687870111405603004150669047695425400659255527534257952962523119432135790466851212153951488070404696997489841209567508258531545826759101673492464629435766692429390841834550890211271107523204799877530360317536396405504858976931856210488365975497495556172569477975427960672635858886247919881599927683923495214201721059388737195064541841735591256798 7

c = 378852978424825502708167454087701637280784822277688792045348887824713793057829679743764792249451048376765115049293335609328896594374157026894386198702427661071271740913994640951396304311446393314608843000423774716342280295925029660257064936301615158136400679589422659958470807258269699674051888760678546077585102981428035938576309107890230195722648462042851360463058513151116701576319059122588420277284045656364315950780571100411390141750375118105082363820780353311142951091161616085139175475443476481956805485082381090115982129784979000564610212935403573535012447683878666154208904550965691034867674284495700885745 7

n = 275459376037517372487852208917357964689733297380762091440799214499672925723494245390105022875640301168312612681973846505110430687389114291697306401359478008859871715392672146119076875705870019338292086551008280456513916180896032884565703345005331786952384076847022512526715793710186516750543686062825246733699830346823305783087698864563358187338272372945704768536735526853616891442615528957582665223930041160178493973462591192210638216632809358
```

```
20440671825601452417487330105280889520007917979115568067161590058277418371493228631232457972494285014767469893647892888681433965857496916110704944758070268626897045014782837
c = 14069112970608895732417039977542732665796601893762401500878786871680645798754783315693511261740059725171342404186571066972546332813667711135661176659424619936101038903439144294886379322591635766682645179888058617577572409307484708171144488708410543462972008179994594087473935638026612679389759756811490524127195628741262871304427908481214992471182859308828778119005750928935764927967212343526503410515793717201360360437981322576798056276657140363337007147322248483468089639923024090377060945889641702395211935894700708397904045972529908185837178691402298117122950057105404763567433789066422670457236338740116492598421
n = 25746162075697911560263181791216433062574178572424600336856278176112733054431463253903433128232709054141607100891177804285813783247735063753406524678030561284491481221681954564804141454666928657549670266775659862814924386584148785453647316864935942772919140563506305666207816897601862713092809234429096584753263707828899780979223118181009293655563146526792388913462557306433664296663314699064286651274388293997030028678002699478558692620367142565500755201931259870119451922735317322766417280084068558715986789365853247824386687468105166601520182442530080924700665556872771389372987479519295762310362513162706025134512
c = 17344284860275489477491525819922855326792275128719709401292545608122859829827462088390044612234967551682879954301458425842831995513832410355328065562098763660326163262033200347338773439095709944202252494552172589503915965931524326523663289777583152664722241920800537867331030623906674081852296232306336271542832728410803631170229642717524942332390842467035143631504401140727083270732464237443915263865880580308776111219718961746378842924644142127243573824972533819479079381023103585862099063382129757560124074676150622288706094110075567706403442920696472627797607697962873026112240527498308535903232663939028587036724
n = 23288486934117120315036919418588136227028485494137930196323715336208849327833965693894670567217971727921243839129969128783853015760155446770590696037582684845937132790047363216362087277861336964760890214059732779383020349204803205725870225429985939570141508220041286857810048164696707018663758416807708910671477407366098883430811861933014973409390179948577712579749352299440310543689035651465399867908428885541237776143404376333442949397063249223702355051571790555151203866821867908531733788784978667478707672984539512431549558672467752712004519300318999208102076732501412589104904734983789895358753664077486894529499
c = 10738254418114076548071448844964046468141621740603214384986354189105236977071001429271560636428075970459890958274941762528116445171161040004083335787613468974984694005261939275039468350481608119343235066945244611328563898255176258665632910900721401994497581643482776888270463046000120945223916289657619187632466233315383553395660029525515837702519842695094404064323543021101106358603246772432973578594737205175904213817105416585484247299058380089998489323254909276640051030008358551301417122042310345229289149614180695630039654068238166836756456942781309206405399310353763599431114301070881485186723970649257720389902 4
n = 19591441383958529435598729113936346657001352578357909347657257239777540424811749817783061233235817916560689138344041497732749011519736303038986277394036718790971374656832741054547056417771501234494768509780369075443550907847298246275717420562375114406055733620258777905222169702036494045086017381084272496162770259955811174440490126514747876661317750649488774992348005044389081101686016446219264069971370646319546429782904810063020324704138495608761532563310699753322444871060383693044481932265801505819646998535192083036872551683405766123968487907648980900712118052346174533513978009131757167547595857552370586353973
c = 38349170988872029319819687046591193416244322947593619195539375510534996074403332340181891419702463022993857425482785898960332828949812003532706371272134831721825298904959034256491167559016311016658763017998656127177503600890851791427506646034541936420530163847145158558683687235089222717671902855211377856880756228329248292483627744764562328268858010469693845195493854282595915667168908446046962587836393908541530393294807262051471992471836215351724508259790471324954396038408065012549971670511424271573817998907253237655588038080301094680486822520287202413574786147046100891208103671924143520341774846885023640228874
n = 19254242571588430171308191757871261075358521158624745702744057556054652332495961196795369630484782930292003238730267396462491733557715379956969694238267908985251699834707734400775311452868924330866502429576951934279223234676654749272932769107390976321208605516299532560054081301829440688796904635446986081691156842271268059970762004259219036753174909942343204432795076377432107630203621754552804124408792358220071862369443201584155711893388877350138023238624566616551246804054720492816226651467017802504094070614892556444425915920269485861799532473383304622064493223627552558344088839860178294589481899206318863310603
c = 67905535399129720580456199122549310531239882518768225078019751078476522642966328422040048056303934193859978334672405107621126566346864382643010901324501403581117829508193995868708747731286772028996450609781976209524447912935999886767181181973819668788469668046345866137431099461076000947426411575020492087552743448643753662358968451941151910017029142336742493856682031548650744420202240800387911846576127391675529089811299152554611419106402299132972437006463256990385618923617789400776669078263024744389535889398373582282424348718185109878727127025678089109440512194763108872991739831765232049776510179013267917188 9
```

```
n = 26809700251171279102974962949184411136459372267620535198421449833298448092580497485301953796619185339316064387798092220298630428207556482805739803420279056191194360049651767412572609187680508073074653291350998253938793269214230457117194434853888765303403385824786231859450351212449404870776320297419712486574804794325602760347306432927281716160368830187944940128907971027838510079519466846176106565164730963988892400240063089397720414921398936399927948235195085202171264728816184532651138221862240969655185596628285814057082448321749567943946273776184657698104465062749244327092588237927996419620170254423837876806659
c = 38621355660843401376986472712387941204199127152899052854850745121069261898665287042463221942460167752426501104314674830977406789498506928806795254613941681940403968845475604486278463088283349609082256858057285902980064667130174890152813215371291330117925487987744132228591454497451972730731100233035053485786751646661247476975357785866007583059289140355186724605739783968832917253017718704222902868586203614077906577106193352813742301940731147358183240589908970925174700278803200209449537961468654467296907324930970348255638602462281473101576781004296981375254861746497491571442559535194026607702167240985864542734 6
```

给了一个e，和多组的n，c。这些nc还都是一个明文m
通过对不同的n进行gcd()算法，求出最大公约数，（即p）
求出P了，就能求出q，进而求出d， 解出明文m
python3代码

```
from gmpy2 import *

n0 = 20474918894051778533305262345601880928088284471121823754049725354072477155873778848055073843345820697886641086842612486541250183965966001591342031562953561793332341641334302847996108417466360688139866505179689516589305636902137210185624650854906780037204412206309949199080005576922775773722438863762117750429327585792093447423980002401200613302943834212820909269713876683465817369158585822294675056978970612202885426436071950214538262921077409076160417436699836138801162621314845608796870206834704116707763169847387223307828908570944984416973019427529790029089766264949078038669523465243837675263858062854739083634207
c0 = 97446390824333086572897876921359540078205339859689774131627572259641501891292950863739385091922496927176638871002519503989696195606289557006214694773634034292797499261667889337274426195417287349087880548324119634588172116407865115606711995781642276852444202568807946265675560598210417400163534587402213304540234401004596111172015199041203447775585180276906930906901873854185413018369220475876142712127998200299393974534369567190001529679063746488033737551153642479689099652668120063308684103632039584772593574475799301335280465057506813612929559130656921330015633365091079594680082006749414336488584289629112613732 0
n1 = 20918819960648891349438263046954902210959146407860980742165930253781318759285692492511475263234242002509419079545644051755251311392635763412553499744506421566074721268822337321637265942226790343839856182100575539845358877493718334237585821263388181126545189723429262149630651289446553402190531135520836104217160268349688525168375213462570213612845898989694324269410202496871688649978370284661017399056903931840656757330859626183773396574056413017367606446540199973155630466239453637232936904063706551160650295031273385619470740593510267285957905801566362502262757750629162937373721291789527659531499435235261620309759
c1 = 15819636201971185538694880505120469332582151856714070824521803121848292387556864177196229718923770810072104155432038682511434979353089791861087415144087855679134383396897817458726543883093567600325204596156649305930352575274039425470836355002691145864435755333821133969266951545158052745938252574301327696822347115053614052423028835532509220641378760800693351542633860702225772638930501021571415907348128269681224178300248272689705308911282208685459668200507057183420662959113956077584781737983254788703048275698921427029884282557468334399677849962342196140864403989162117738206246183665814938783122909930082802031855
n2 = 25033254625906757272369609119214202033162128625171246436639570615263949157363273213121556825878737923265290579551873824374870957467163989542063489416636713654642486717219231225074115269684119428086352535471683359486248203644461465935500517901513233739152882943010177276545128308412934555830087776128355125932914846459470221102007666912211992310538890654396487111705385730502843589727289296921521771347530986497814122470656606378262820551699918240991109165768561888769756213766066342589277840257871422633671529471087207572224466864156274797036660318716356563142827270511891908890087630558116800403152770789280688164 91
c2 = 41853085294168740058312307810140924071984513859556773996685018339026234783956692794048839907251843327091524433725837010761987866352917393567708572867021071567300200043589556225110614106610589826220551997368208082038414467963052843946517144309186903894869205608346723161581464531837894121403902902932475603535808175442664516003326292433024867521610827098015704970548862026348512948095281476400286528001918512766244931832427938327776641625814227514392353216879841301102827154308524902904899745221250311174230230206540105145806658539536046844746065867295285
```

```
164354719382277521838785362345363802549238912220450755590886 2

n3 = 212069680973141310071834279444868019535831511514436279431137369967767871811110639579606980926968005550441991567656779353731495982211847922868122132946177498346076963021161367456628166581170554278033152300427006951257184016468104848730647750052210891740568247249221608558105272367513896050175795452358768649984198730652172948202447307851205251265658155602290018876228375491181680816851833710923951285981250047302689102760248068085658020813668989040325099204537859970561504976452349255288838794196421891096490091323815866733900276147666050389510158530867211680187875234592649321650468168816827742292436885816143064807513c3 = 452103801104475844189112846846723308849388575085058898570851991115477809059713612615028904189345412667446814139347266233735036171221269486731162297044070772794111326383235717314177585522797374257108897459347630208411177062576422283836627755956088704294885989213855147268065451781491660927974836558061071225985667774051847708653159223310717547006829190360750579943293198966370747701790461142621377023839700574373038608003195569415846656847559975194024503916762912657684024482348452868313417471542956778285567779435940267140679906686531862467627238401003459101637191297209422470388121802536569761414457618258343550613n4 = 228220397330493881109367781730147656636633038117912832343612306497758059239021734385539278054074631061046997739941583757040330934717613877998521683378985269805217536143078996690159313878199274218753163045915219015928238144177564476957010458467735086293713970130536845530421857250599679153239162642971241699499088969373280518194797007142930959961497377273655629940424642479166067925388494002172884690634419885477919195173971934290876133066191047711993342855077424291042095249692960568615479948783992342433635374744215357167806452076314979329436078782175170354328869672692390967039682155105304803561949970639111814506 7c4 = 154064985807617801086258918780085268151453720962340839366814422251550972992648086243588266869065355948536226873792689694684330723881497866073953964241043188208794437431123587065467539352157560783459593752996507185557596988878523180175975030743173567451225144818078437456264297978614630129401727976125890316867181853903453892958510752792785161470766022701785406901478083141727989874972593300378103285234648518956218518590278236816559341047136895398480471630886668964736655001581790461965382107788977302095727084300676584117559598660335317004605515563809939827061718489704602243049964556005039822234489048782128494123571 8n5 = 215741398553414329084740647843184620184752968093272855323377069401269425753495076682892140780261026822527137577030815530931088232140637915184822898467801973298211395079747637802602903096008849208119598429255405839670856708487653178774414809148523292763757764056897845714046358522040976226006562227148085418722523358770375613884062571817152787666528247863762622492749604671939619566909748536797952491587510784222965803675062197197387621599659588778061874610706890712909481819495612541443107769433348597751216501862458460317205079449878384897231278972234168024360212786712372279936867919447114223450004797511877044263691c5 = 203668561507103051245830653752976618197952422383764852649511853369960837446045934189833362851854911974260185950314446521232884614918790210960282036941366832034416929870695635130260018614357221179855599096926709073475635945782658808065403967772239069554910262868431686373675934003428147256943660783370309371040359935696729593613472878941430271868468567729830583289197167029822221428488481177684999661758830530148308542854726733707099876741254022591150819684225313435590126386112150065024029674670296759422440165022016878053714165448921501914212228430811628412900425736476947408072100170873405126484135042415250602793 2

n6 = 253602274126666124901021611311745848192409318031964484812243052505838414395810085285359308141673383819837649912965756372319165476479705737582694111682193023705416847891251125050211485068096430819502376237031810256965859804469569132201218366042463649689707304555740076874594378734254826738656462546214315017611365626445021002392557194596140570927663199073160219810428752852805565005048615983761227960041525948630615494751400540890759008374775895311548612486548672063382055913506344094252803140295195855763083350377511201071560427811432552899377108123353524711848176585227325240496343079289894821953947331246297984913 7c6 = 198927725246514523410275956194827343562434356715923981726803799815027596957840879006690899199877056758999456586486238000902725991545901230821896450218009580768615183973254395211399956520263771323682325021086200334000513461277576986238861426217934232257492402865116665560917878516839780175069833100735243982872797376800917873335475382399206077610809882436395475708183637886732495827830154756821099847152931631373244398628385744601087937141726036724777668313564113044468819986747795011881636006644880329436396948286989847394922006996844627489228835500026529135182293229450408190641333503145363786945237047933961690651791 8n7 = 227268552446323560291596917534518221633315192375476399387795177514964987131745889355665761673295764947902193607278771660741364961299272962969969700480828704888044565649866712938813655613701334622811898193689951068758958528651715132304829315025703684747542404437810916817941228788934059639475525770493800616267765658150937547110254626135574825186904800360052003465626452193180865103852413418573292957038470591856398206568414576642796250226152248199419198982011057598190699843155310752554200118765570353468323177798841926833824954764133571839331229580004471 2
```

```
345347616927994034694979540628978562990312574547359458674911191
c7 = 6040119795175856407541082360023532204614723858688636724822712717572759793960246341800308149739809871234313049629732934797569781053000686185666374833978403290525072598774001731350244744590772795701065129561898116576499984185920661271123665356132719193665474235596884239108030605882777868856122378222681140570519180321286976947154042272622411303981011302586225630859892731724640574658125478287115198406253847367979883768000812605395482952698689604477719478947595442185921480652637868335673233200662100621025061500895729605305665864693122952557361871523165300206070325660353095592778037767395360329231331322823610060006

n8 = 2329733379144305329736300078683533609525229081846195005454265832748450740659463278571276745995891794309552259422820542342820734512889974580092731914725766977381266954278283923774430518009827657884192949634596399751224421937670178761604623539713938189483743556266259106076847699733353874806529403314161050225232529280181681226893417136193439995154862726779140108970393738901258658108022331306015945623885708074069952866641130302993480701121495398416978584471415962779201692649095528269787714161463880639768930679532834477847869208475421675342584255781889946794510264677634265516765538422486050408608314784125223276094l
c8 = 5418120301208378713115889465579964257871814114515046096090960159737859076829258516920361577853903925954198406843757303687557848302302200229295916902430205737843601806700738234756698575708612424928480440868739120075888681672062206529156566421276611107802917418993625029690627196813830326369874249777619239603300605876865967515719079797115910578653562787899019310139945904958024882417833736304894765433489476234575356755275147256577387022873348906900149634940747104513850154118106991137072643308620284663108283052245750945228995387803432128842152251549292698947407663643895853432650029352092018372834457054271102816934

n9 = 2887366790471568272298723429349320030697694789871125506412511593366696867874259885872243142621891446290352159634177113169561938226619423356167782435737980530388599380426643681060626302209790026697525043157565468691504969309146786482051276707071326770899389989901115610676617890670033611171280336211303961354867293705339787566314479401808701773194908779489490373768238391617326742140340814096771307102600187473348729500750106887104464917061570989145185679223231552669622016184274266477858128732131874820243146650894890274531437229979956162518695523467301209821091974587988226851265693171432678233521108957689731059149l
c9 = 9919880463786836684987957979091527477471444996392375244075527841865509160181666543016317634963512437510324198702416322841377489417029572388474450075801462996825244657530286107428186354172836716502817609070590929769261932324275353289939302536440310628698349244872064005700644520223727670950787924296004296883032978941200883362653993351638545860207179022472492671256630427228461852668118035317021428675954874947015197745916918197725121122236369382741533983023462255913924692806249387449016629865823316402366017657844166919846683497851842388058283856219900535567427103603869955066193425501385255322097901531402103883869

n10 = 223246859475396537224999324694096075330654191573478139619580756890476904652664043841994836839085947873124455281596355278339044758018903814556538072655012173287578713527312930003034382053158167926639175790666748423077438452617710323639285688446698957680925156583287562292458370252617442606148607469979315035487885099838680383497202253057309855762936752690737090223507008365100540676417537132129999543070225244958855833617073785137421625663390101343549078637332059218450389182244639037898418814008140745872617202838797601220709014665171182654228634203769215367348455021002514608724991222366868321895496980207371766830l9
c10 = 1491527050203294989882829248560395184804977277747126143103957219164624187528441047837351263580440686474767380464005540264627910126483129930668344095814547592115061057843470131498075060420395111008619027199037019925701236660166563068245683975787762804359520164701691690916482591026138582705558246869496162759780878437137960823000043988227303003876410503121370163303711603359430764539337597866862508451528158285103251810058741879687875218384160282506172706613359477657215420734816049393339593755489218588796607060261897905233453268671411161063104734045948793747951193345036946221379573893301900147180315760779173853846l7
```

n11 = 2764674642375902011100782865326402799925784764566612990778902605459439364880023611704676911276264177886562089244342310018961932758581138488351542491875274955962755363778503735963980112521325616300843194259372793193189819997275527686267756184798330291012496925737160307066957025109822835557408510470226724857434324646477728823142151761147322574972402841640169140186890445572189203002622346528406324060672733752693010084098601931808223667358772882057833143261022637565037867361223213483200319500121449058695562040174305936560528679394936331634995802422247634043388070225101362171877790849179961716027370365649910367242991
c11 = 2199152412895726053604377128485492039310580812670012822212585677550688572197119310936131596112919081467464713646488708789399066089496161283820508640101888545766748891189865427023556198011117460332372128091119748828658526935684957926304345631631947649588869621934421986651686118765418050924788125125127891934626712990473927738628924039438457512433113565594351383100993402339745708218469973773438882376330680532643039584993577021381753338723548630700889241092061166993269301816556941744588581082574960938862723123584091264465468581962093166334629759633483449866178901645037176920365010999477187240418577023017293401397l
```
```

```
n12 = 2054548740581692873173898837447501268682793370978978439185570683513627027093340120301932913693765087838611
7187776530639342572123237188053978622697282521473917978282830432161153221216194169879669541998840691383025487220
8508720754360643084999249585179797279544029656121960814043416515173263640415192501250364248226343542687738954656
9892088343922299658122635859587399397660469983061393232072055413001167129794443351504718056548449519100388759989
1289037982010216357831078328159028953222056918189365840711588671093333013117454034313622855082795813122338562446
2230412111922770892250783246821080338430239035501728919596735 51
c12 = 1422743918819102946125047669279053965461919988848731942911441455797537630868890802814081715720557980405978
3807641305577385724758530138514972962209062230576107406142402603484375626077345190883094097636019771377866339531
5119651366505674123638891831596161884492637524753286632453110599883379960473592632888374363055888480445729377594
2446658687028051242433680706472989451584055240475687959069879704633333644546512044508758762174390662427962177963
4772378802959109714400516183718323267273824736540168545946444437586299214110424738159957388350785999348535171553
5693730882515527123912883652952676656913577196160116136287721 75
n13 = 2735972771158427723489715772405585279401921684522979893865581426946004638435356813859856775539255965346094
9444557879120040796798142218939251844762461270251672399546774067275348291003962551964648742053215424620256999345
4483988052785927770496682815583128717739799313430978068787011140560300415066904769542540065925552753425795296252
3119432135790466851212153951488070404696997489841209567508258531545826759101673492464629435766692429390841834550
8902112711075232047998775303603175363964055048589769318562104883659754974955561725694779754279606726358588862479
1988159992768392349521420172105938873719506454184173559125679 87
c13 = 3788529784248255027081674540877016372807848222776887920453488878247137930578296797437647922494510483767651
1504929333560932889659437415702689438619870242766107127174091399464095139630431144639331460884300042377471634228
0295925029660257064936301615158136400679589422659958470807258269699674051888760678546077585102981428035938576309
1078902301957226484620428513604630585131511167015763190591225884202772840456563643159507805711004113901417503751
1810508236382078035331114295109116161608513917547544347648195680548508238109011598212978497900056461021293540357
3535012447683878666154208904550965691034867674284495700885 7457
n14 = 2754593760375173724878522089173579646897332973807620914407992144996729257234942453901050228756403011683126
1268197384650511043068738911429169730640135947800885987171539267214611907687570587001933829208655100828045651391
6180896032884565703345005331786952384076847022512526715793710186516750543686062825246733699830346823305783087698
8645633581873382723729457047685367355268536168914426155289575826652239300411601784939734625911922106382166328093
5820440671825601452417487330105280889520007917979115568067161590058277418371493228631232457972494285014767469893
6478928886814339658574969161107049447580702686268970450147828 37
c14 = 1406911297060889573241703997754273266579660189376240150087878687168064579875478331569351126174005972517134
2404186571066972546332813667711135661176659424619936101038903439144294886379322591635766682645179888058617577572
4093074847081711444887084105434629720081799945940874739356380266126793897597568114905241271956287412628713044279
0848121499247118285930882877811900575092893576492796721234352650341051579371720136036043798132257679805627665714
0363332700714732224848346808963992302409037706094588964170239521193589470070839790404597252990818583717869140229
8117122950057105404763567433789066422670457236338740116492598 42
c15 = 2574616207569791156026318179121643306257417857242460033685627817611273305443146325390343312823270905414160
7100891177804285813783247735063753406524678030561284491481221681954564804141454666928657549670266775659862814924
3865841487854536473168649359427729191405635063056662078168976018627130928092344290965847532637078288997809792231
1818100929365556314652679238891346255730643366429696633146990642866512743882939970300286780026994785586926203671
4256550075520193125987011945192273531732276641728008406855871598678936585324782438668746810516660152018244253008
0924700665556872771389372987479519295762310362513162706025134 51
c15 = 1734428486027548947749152581992285532679227512871970940129254560812285982982746208839004461223496755168287
9954301458425842831995513832410355328065562098763660326163262033200347338773439095709944202252494552172589503915
9659315243265236632897775831526647222419208005378673310306239066740818522962323063362715428327284108036311702296
4271752494233239084246703514363150440114072708327073246423744391526386588058030877611121971896174637884292464414
2127243573824972533819479079381023103585862099063382129757560124074676150622288706094110075567706403442920696472
6277976076979628730261122405274983085359032326639390285870367 24
n16 = 2328848693411712031503691941858813622702848549413793019632371533620884932783396569389467056721797172792124
3839129969128783853015760155446770590696037582684845937132790047363216362087277861336964760890214059732779383020
3492048032057258702254299859395701415082200412868578100481646967070186637584168077089106714774073660988834308118
6193301497340939017994857771257974935229944031054368903565146539986790842888554123777614340437633344294939706324
9223702355051571790555151203866821867908531733788784978667478707672984539512431549558672467752712004519300318999
2081020767325014125891049047349837898953587536640774868945294 99
c16 = 1073825441811407654807144884496404646814162174060321438498635418910523697707100142927156063642807597045989
```

```
095827494176252811644517116104004083335787613468974984694005261939275039468350481608119343235066945244611328563898255176258665632910900721401994497581643482776888270463046000120945223916289657619187632466233315383553395660029525515837702519842695094404064323543021101106358603246772432973578594737205175904213817105416585484247299058380089998489323254909276640051030008358551301417122042310345229289149614180695630039654068238166836756456942781309206405399310353763599431114301070881485186723970649257720389902 4

n17 = 19591441383958529435598729113936346657001352578357909347657257239777540424811749817783061233235817916560689138344041497732749011519736303038986277394036718790971374656832741054547056417771501234494768509780369075443550907847298246275717420562375114406055733620258777905222169702036494045086017381084272496162770259955811174440490126514747876661317750649488774992348005044389081101686016446219264069971370646319546429782904810063020324704138495608761532563310699753322444871060383693044481932265801505819646998535192083036872551683405766123968487907648980900712118052346174533513978009131757167547595857552370586353973
c17 = 38349170988872029319819687046591193416244322947593619195539375510534996074403332340181891419702463022993857425482785898960332828949812003532706371272134831721825298904959034256491167559016311016658763017998656127177503600890851791427506646034541936420530163847145158558683687235089222717671902855211377856880756228329248924836277447645623282688580104696938451954938542825959156671689084460469625878363939085415303932948072620514719924718362153517245082597904713249543960384080650125499716705114242715738179989072532376555880380803010946804868225202872024135747861470461008912081036719241435203417748468850236402288 7

n18 = 19254242571588430171308191757871261075358521158624745702744057556054652332495961196795369630484782930292003238730267396462491733557715379956969694238267908985251699834707734400775311452868924330866502429576951934279223234676654749272932769107390976321208605516299532560054081301829440688796904635446986086191156842271268059970762004259219036753174909942343204432795076377432107630203621754552804124408792358220071862369443201584155711893388877350138023238624566616551246804054720492816226651467017802504094070614892556444425915920269485861799532473383304622064493223627552558344088839860178294589481899206318863310603
c18 = 67905535339912972058045619912254931053123988251876822507801975107847652264296632842204004805630393419385997833467240510762112656634686438264301090132450140358111782950819399586870874773128677202899645060978197620952444791293599988676718118197381966878846966804634586613743109946107600094742641157502049208755274344864375366235896845194115191001702914233674249385668203154865074442020224080038791184657612739167552908981129915255461141910640229913297243700646325699038561892361778940077666907826302474438953588939837358228242434871818510987872712702567808910944051219476310887299173983176523204977651017901326791718890 5

n19 = 26809700251171279102974962949184411136459372267620535198421449833298448092580497485301953796619185339316064387798092220298630428207556482805739803420279056191194360049651767412572609187680508073074653291350998253938793269214230457117194434853888765303403385824786231859450351212449404870776320297419712486574804794325602760347306432927281716160368830187944940128907971027838510079519466846176106565164730963988892400240063089397720414921398936399927948235195085202171264728816184532651138221862240969655185596628285814057082448321749567943946273776184657698104465062749244327092588237927996419620170254423837876806659
c19 = 38621355660843401376986472712387941204199127152899052854850745121069261898665287042463221942460167752426501104314674830977406789498506928806795254613941681940403968845475604486278463088283349609082256858057285902980646671301748901528132153712913301179254879877441322285914544974519727307311002330350534857867516466612474769753577858660075830592891403551867246057397839688329172530177187042229028685862036140779065771061933528137423019407311473581824058990897092517470027880320020944953796146865446729690732493097034825563860246228147310157678100429698137525486174649749157144255953519402660770216724098586454273 46

n=[n0,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18,n19]
c=[c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,c19]

for i in range(len(n)):
    for j in range(len(n)):
        if(i!=j):
            if(gcd(n[i],n[j])!=1):   #对不同的n进行 欧几德得 算法，以求出最大公约数
                print(i,j)
                print("p =",gcd(n[i],n[j]))
```

这里求出p

```
4 17
p = 13258580638379860030542695730761256760422356262676419021133313624664372381104614933785296682872905247672555236113243737052154870766497712316527930505297186801275550916040864110054874404662151687798186418
17 4
p = 13258580638379860030542695730761256760422356262676419021133313624664372381104614933785296682872905247672555236113243737052154870766497712316527930505297186801275550916040864110054874404662151687798186418
```

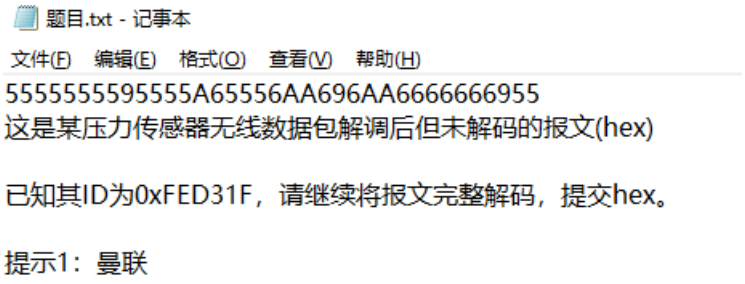
有了p就直接求q，得到m再转换成字符

```
from gmpy2 import*
import libnum
n4 = 22822039733049388110936778173014765663663303811791283234361230649775805923902173438553927805407463106104699
77399415837570403309347176138779985216833789852698052175361430789966901593138781992742187531630459152190159282381
44417756447695701045846773508629371397013053684553042185725059996791532391626429712416994990889693732805181947970
07142930959961497377273655629940424642479166067925388494002172884690634419885477919195173971934290876133066191 0
47711993342855077424291042095249692960568615479948783992342433635374744215357167806452076314979329436078782175170
03543288696726923909670396821551053048035619499706391118145067
c4 = 15406498580761780108625891878008526815145372096234083936681442225155097299264808624358826686906535594853622
68737926896946843307238814978660739539642410431882087944374311235870654675393521575607834595937529965071855575960
98887852318017597503074317356745122514481807843745626429797861463012940172797612589031686718185390345389295851070
52792785161470766022701785406901478083141727989874972593300378103285234648518956218518590278236816559341047136890
53984804716308866689647366550015817904619653821077889773020957270843006765841175595986603353170046055155638099390
82706171848970460224304996455600503982223448904878212849412357

p = mpz(13258580638379860030542695730761256760422356262676419021133313624664372381104614933785296682872905247672
55523611324373705215487076649771231652793050529718680127555091604086411005487440466215168779818641800764975240930
20140455803630182021627496863882524515075577255925957554410191859031106846660161847246483249 9)

q = n4//p        #“//”  整除

phi = (p-1)*(q-1)

e = 65537
d = invert(e,phi)

m = pow(c4,d,n4)

print(m)    # "n2s" (数值转字符串)
print(hex(m))
print(bytes.fromhex(hex(m)[2:]))
```

```
13040004482825176402070107903979416267670062118522537076883968693524598900675425175282673277
0x666c61677b6162646362653566643934653233623366465343239323233616239633266646 67d
b'flag{abdcbe5fd94e23b3de429223ab9c2fdf}'
```

直接提交flag即可

50.传感器



说实话我也是一点头绪没有 直接找大佬的wp
大佬说这是曼彻斯特编码
然后我找了一下这个编码
编码原理
发现啥也不是，抄脚本吧

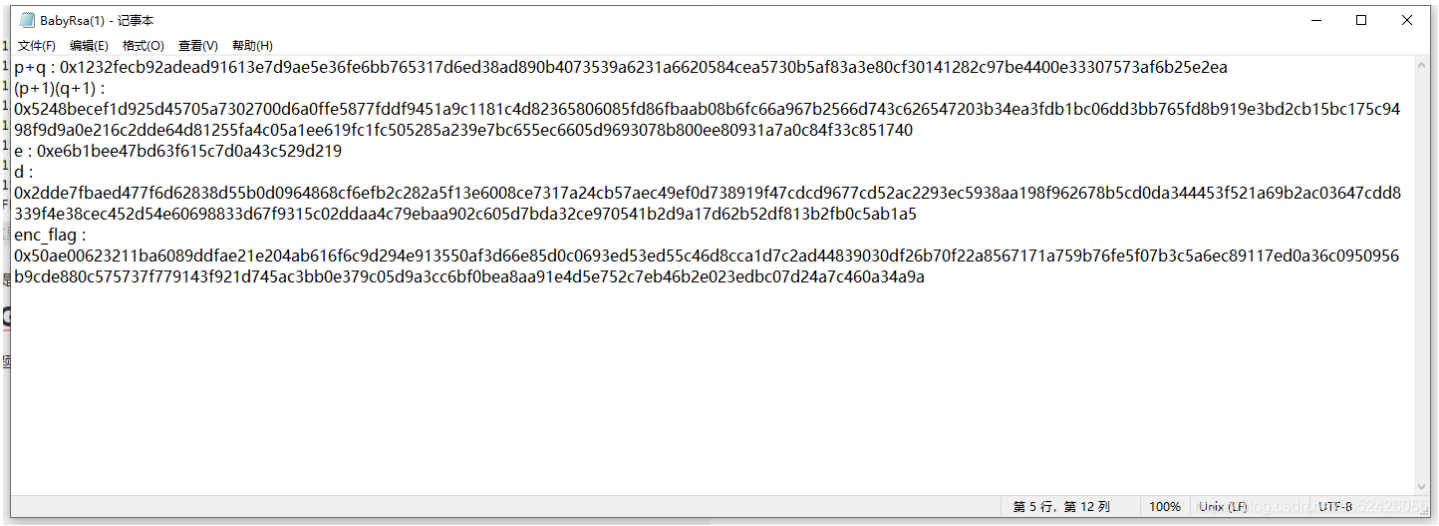
```
cipher='5555555595555A65556AA696AA6666666955'
def iee(cipher):
    tmp=''
    for i in range(len(cipher)):
        a=bin(eval('0x'+cipher[i]))[2:].zfill(4)
        tmp=tmp+a[1]+a[3]
        print(tmp)
    plain=[hex(int(tmp[i:i+8][::-1],2))[2:] for i in range(0,len(tmp),8)]
    print(''.join(plain).upper())

iee(cipher)
```

```
11111111111111101111111110010111111100000100110000010101
11111111111111101111111110010111111100000100110000010101010
11111111111111101111111110010111111100000100110000010101010
11111111111111101111111110010111111100000100110000010101010
11111111111111101111111110010111111100000100110000010101010
11111111111111101111111110010111111100000100110000010101010
11111111111111101111111110010111111100000100110000010101010
11111111111111101111111110010111111100000100110000010101010
FFFFFED31F645055F9
```

白嫖提交

51.[GUET-CTF2019]BabyRSA

查看题目



观察题目给的条件，给了p+q，（p+1)(q+1)，e，d,以及密文C.

RSA的解密公式：M=C^d mod n

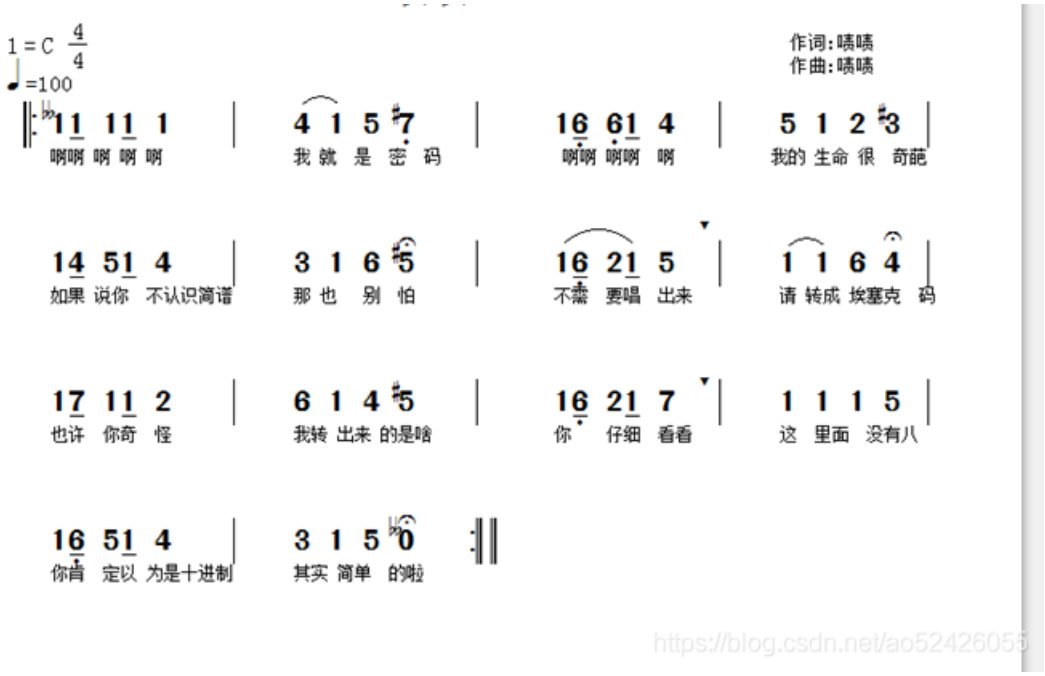
所以我们只要求出n即可。(n = p*q)*

n = (p+1)(q+1) - (p+q) - 1*

求M的值，已知C，d，n后 用函数pow(),即可求出

```
import libnum
a = 0x1232fecb92adead91613e7d9ae5e36fe6bb765317d6ed38ad890b4073539a6231a6620584cea5730b5af83a3e80cf30141282c97be4400e33307573af6b25e2ea
b = 0x5248becef1d925d45705a7302700d6a0ffe5877fddf9451a9c1181c4d82365806085fd86fbaab08b6fc66a967b2566d743c626547203b34ea3fdb1bc06dd3bb765fd8b919e3bd2cb15bc175c9498f9d9a0e216c2dde64d81255fa4c05a1ee619fc1fc505285a239e7bc655ec6605d9693078b800ee80931a7a0c84f33c851740
e = 0xe6b1bee47bd63f615c7d0a43c529d219
d = 0x2dde7fbaed477f6d62838d55b0d0964868cf6efb2c282a5f13e6008ce7317a24cb57aec49ef0d738919f47cdcd9677cd52ac2293ec5938aa198f962678b5cd0da344453f521a69b2ac03647cdd8339f4e38cec452d54e60698833d67f9315c02ddaa4c79ebaa902c605d7bda32ce970541b2d9a17d62b52df813b2fb0c5ab1a5
c= 0x50ae00623211ba6089ddfae21e204ab616f6c9d294e913550af3d66e85d0c0693ed53ed55c46d8cca1d7c2ad44839030df26b70f22a8567171a759b76fe5f07b3c5a6ec89117ed0a36c0950956b9cde880c575737f779143f921d745ac3bb0e379c05d9a3cc6bf0bea8aa91e4d5e752c7eb46b2e023edbc07d24a7c460a34a9a

n = b-a-1

m = pow(c,d,n)

print(libnum.n2s(m))  #（n2s将数值转化为字符串）
```

52.密码学的心声



图片中的线索很明显ASCLL码 八进制
数字三个一组，转换就行了

```
s = '111 114 157 166 145 123 145 143 165 162 151 164 171 126 145 162 171 115 165 143 150'
tmp = [s.split(' ')[i] for i in range(len(s.split(' ')))]
cipher = ''
for i in tmp:
    cipher += chr(int(i,8))
flag = "flag{"+cipher+"}"
print(flag)
```

运行即可

```
D:\python38\python.exe D:/pycharm/venv/mima/8zhuanwen.py
flag{ILoveSecurityVeryMuch}
```

53.rot

看题目

破解下面的密文:

83 89 78 84 45 86 96 45 115 121 110 116 136 132 132 132 108 128 117 118 134 110 123 111 110 127 108 112 124 122 108 118 128 108 131 114 127 134 108 116 124 124 113 108 76 76 76 76 138 23 90 81 66 71 64 69 114 65 112 64 66 63 69 61 70 114 62 66 61 62 69 67 70 63 61 110 110 112 64 68 62 70 61 112 111 112

flag格式flag{}

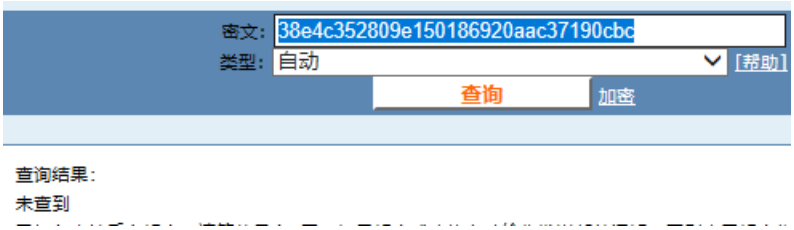
题目名字叫rot
ROT是一种古典密码，左边移某位，或者右边移某位，凯撒算是ROT的一种。
Rot13 加密，用python脚本将每个数字减去13后，转ASCLL字符，得到

```
a = [83,89,78,84,45,86,96,45,115,121,110,116,136,132,132,132,108,128,117,118,134,110,123,111,110,127,108,112,124
,122,108,118,128,108,131,114,127,134,108,116,124,124,113,108,76,76,76,76,138,23,90,81,66,71,64,69,114,65,112,64,
66,63,69,61,70,114,62,66,61,62,69,67,70,63,61,110,110,112,64,68,62,70,61,112,111,112]
b = []
for i in a:
    i = i-13
    b.append(i)

for j in b:
    print(chr(j),end='')
```

直接跑了个寂寞

```
FLAG IS flag{www_shiyanbar_com_is_very_good_????}
MD5:38e4c352809e150186920aac37190cbc
```

后面有四个问好，给了md5



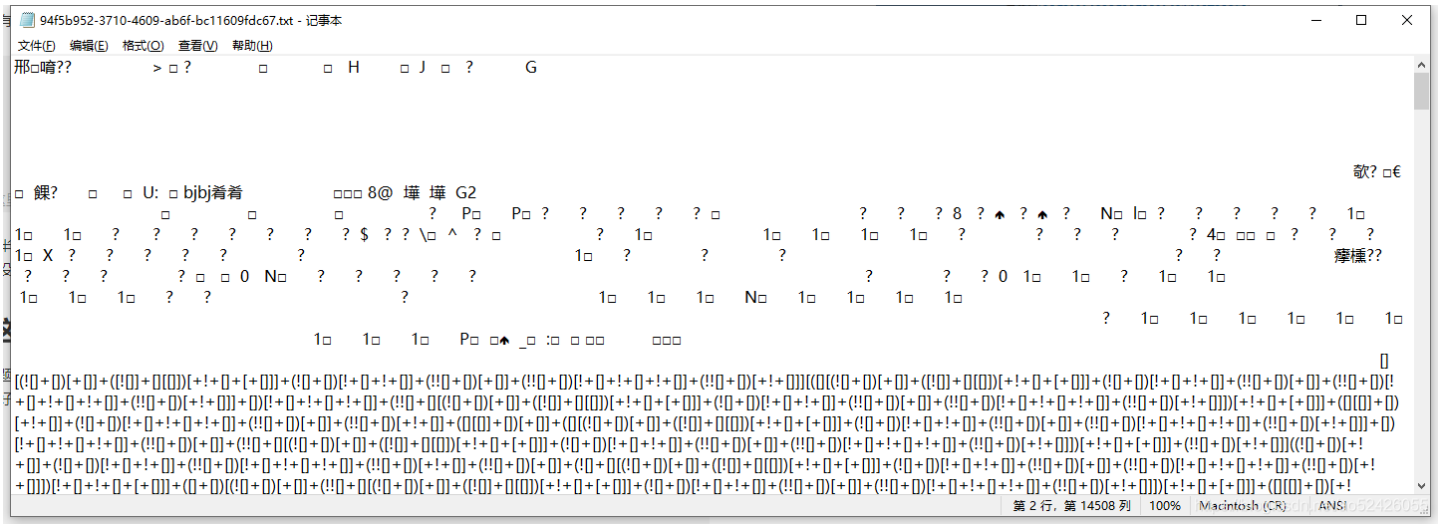
看了半天，还是找大佬的wp吧

我是没跑出来拿flag吧 flag{www_shiyanbar_com_is_very_good_@8Mu}

54.这是什么

查看题目

下载好了以后，把后缀改成txt



我也不知道这些都是什么东西

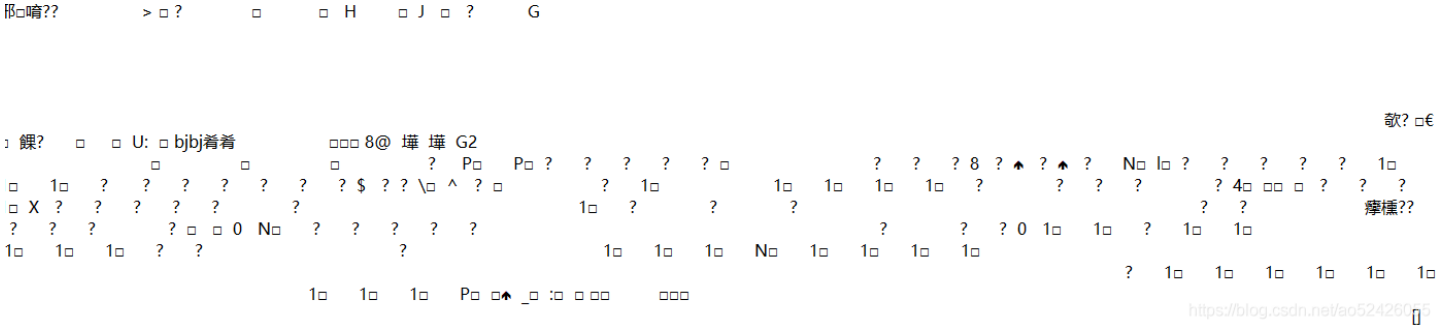
我在网上找了一下

JSfuck编码，解码工具

Run This 得到flag

flag{a0448fd730b62c13ca80200c4529daa2}

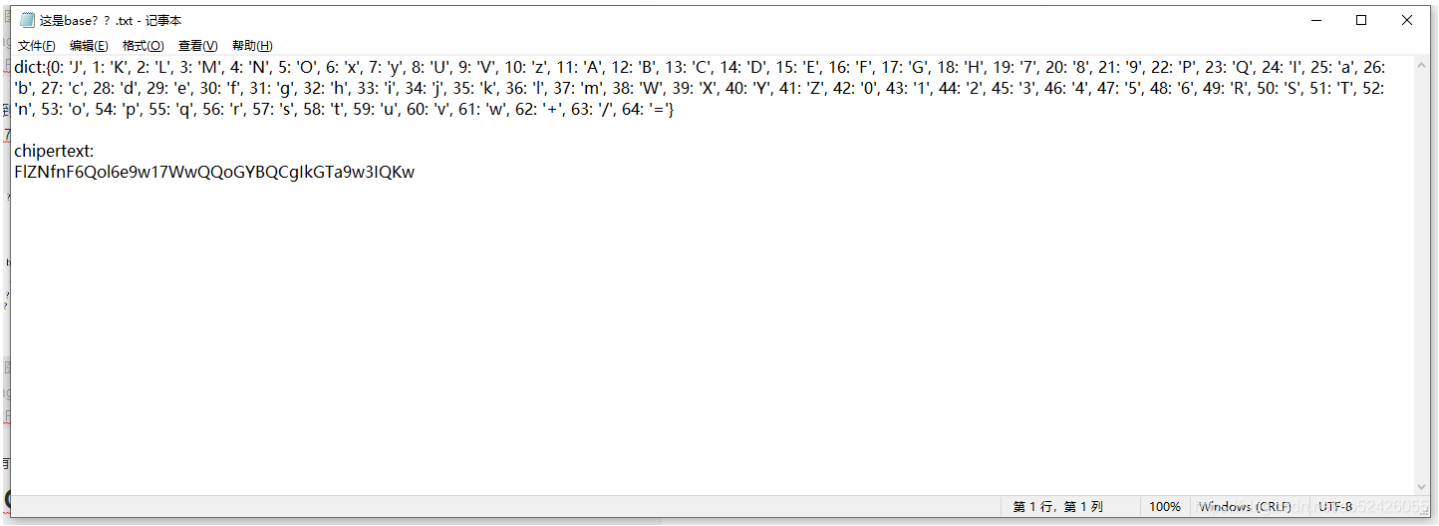
注意



右下角那还有一个 我一开始就是没看到还以为是我找错了

55.[BJDCTF 2020]这是base？？

查看题目



得到的 flag 请包上 flag{} 提交。来源：
https://github.com/BjdsecCA/BJDCTF2020

这里给了一个来源

给了一个关于base64的dict字典，查一下base64的标准字典

索引	对应字符	索引	对应字符	索引	对应字符	索引	对应字符
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

这应该是替换加密，那么直接脚本

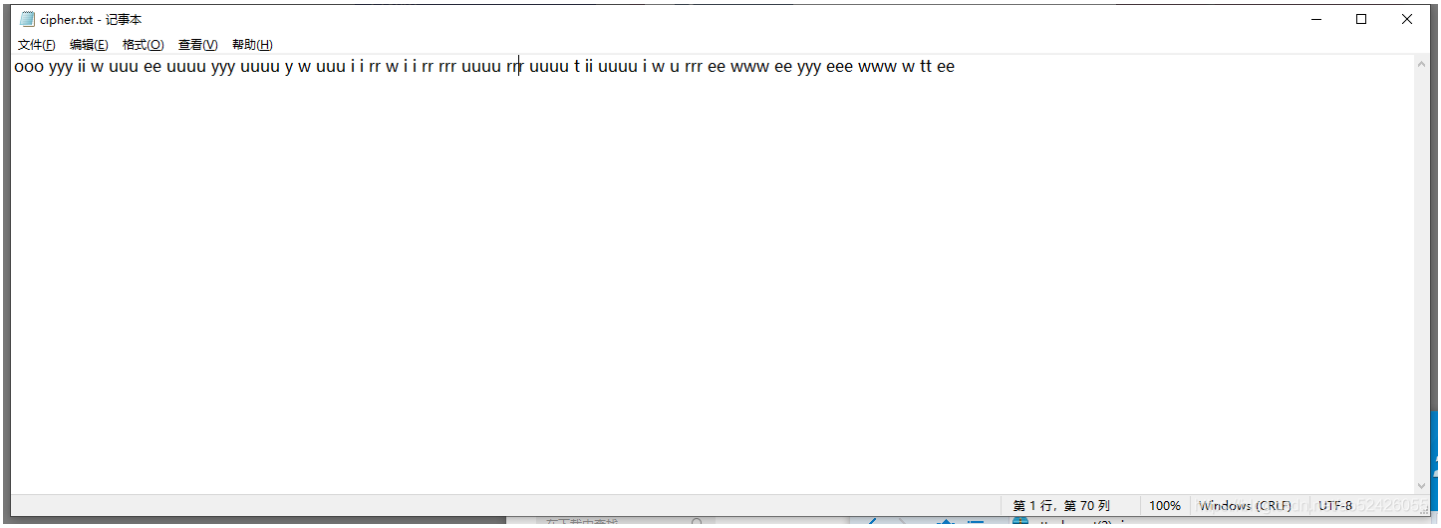
```python
import base64
dict={0: 'J', 1: 'K', 2: 'L', 3: 'M', 4: 'N', 5: 'O', 6: 'x', 7: 'y', 8: 'U', 9: 'V', 10: 'z', 11: 'A', 12: 'B',
 13: 'C', 14: 'D', 15: 'E', 16: 'F', 17: 'G', 18: 'H', 19: '7', 20: '8', 21: '9', 22: 'P', 23: 'Q', 24: 'I', 25:
 'a', 26: 'b', 27: 'c', 28: 'd', 29: 'e', 30: 'f', 31: 'g', 32: 'h',33: 'i', 34: 'j', 35: 'k', 36: 'l', 37: 'm',
 38: 'W', 39: 'X', 40: 'Y', 41: 'Z', 42: '0', 43: '1', 44: '2', 45: '3', 46: '4', 47: '5', 48: '6', 49: 'R', 50:
 'S', 51: 'T', 52: 'n', 53: 'o', 54: 'p', 55: 'q', 56: 'r', 57: 's', 58: 't', 59: 'u', 60: 'v', 61: 'w', 62: '+'
, 63: '/', 64: '='}
base64_list = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P','Q', 'R', 'S', 'T'
, 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f','g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', '
q', 'r', 's', 't', 'u', 'v','w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/']
cipher='FlZNfnF6Qol6e9w17WwQQoGYBQCgIkGTa9w3IQKw'
res=''
for i in range(len(cipher)):
    for j in range(64):
        if(dict[j]==cipher[i]):
            res+=base64_list[j]
flag=base64.b64decode(res)
print(flag)
```

```
D:\python38\python.exe D:/pycharm/venv/mima/tihuanjiami.py
b'BJD{D0_Y0u_kNoW_Th1s_b4se_map}'
```

运行BJD改成flag即可

56.[NCTF2019]Keyboard

查看题目



提示是键盘，然后发现这些字母是26键盘的第一行，上面有对应数字，并且位数在1到4位，说明是九键键盘，刚好和题目意思对上了，这种题写法就是比如o,对应9，就在九键键盘上9的位置，看o有多少位，3位的话，就是9那个位置字符串的第三个字符。

我懒得写脚本，就直接手写了主要是不多

```
youaresosmartthatthisisjustapieceofcake
```

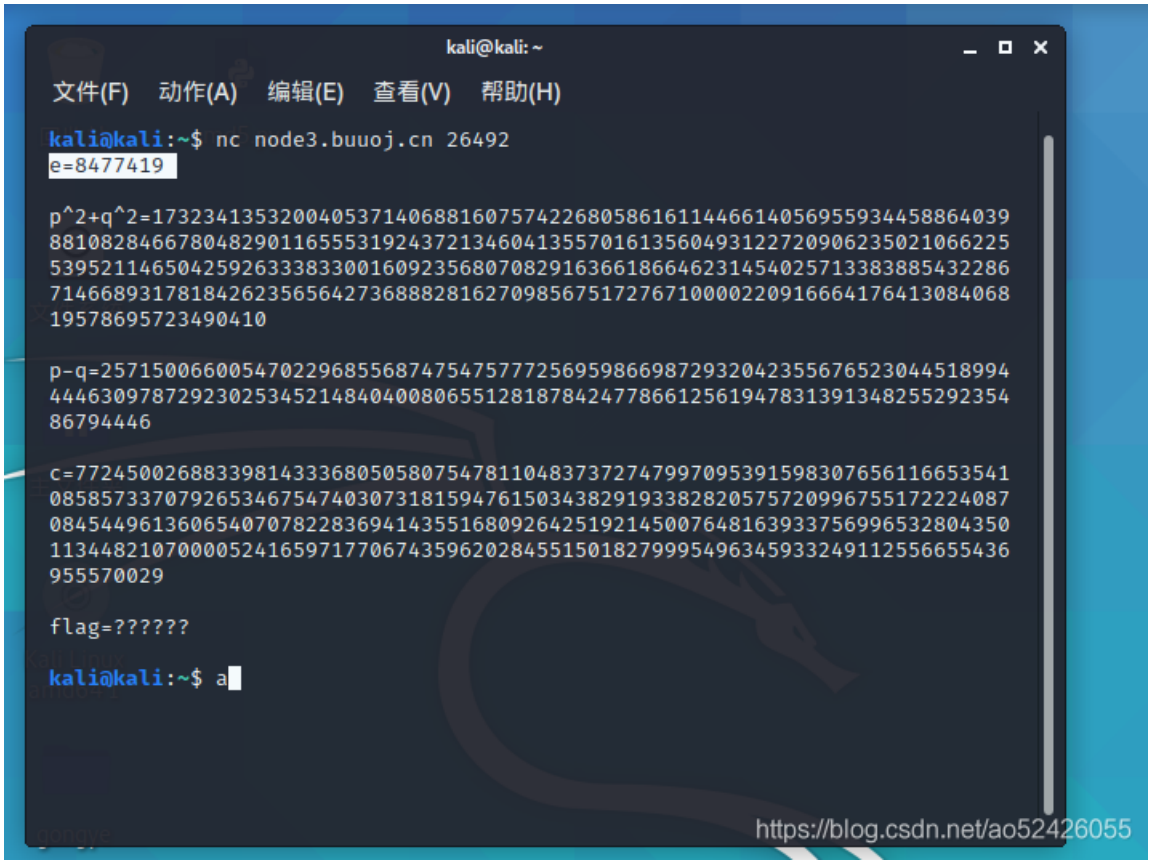
然后又在网上帮你们找了一个脚本

```
cipher="ooo yyy ii w uuu ee uuuu yyy uuuu y w uuu i i rr w i i rr rrr uuuu rrr uuuu t ii uuuu i w u rrr ee www e
e yyy eee www w tt ee"
base=" qwertyuiop"
a=[" "," ","abc","def","ghi","jkl","mno","pqrs","tuv","wxyz"]
for part in cipher.split(" "):
    s=base.index(part[0])
    count=len(part)
    print(a[s][count-1],end="")
```

跑出来和我手写的是一样的

57.[BJDCTF 2nd]rsa1

查看题目，kali nc链接


kali@kali: ~
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)
kali@kali:~$ nc node3.buuoj.cn 26492
e=8477419
p^2+q^2=173234135320040537140688160757422680586161144661405695593445886403988108284667804829011655531924372134604135570161356049312272090623502106622553952114650425926333833001609235680708291636618664623145402571338388543228671466893178184262356564273688828162709856751727671000022091666417641308406819578695723490410
p-q=257150066005470229685568747547577725695986698729320423556765230445189944446309787292302534521484040080655128187842477866125619478313913482552923548679443
c=772450026883398143336805058075478110483737274799709539159830765611665354108585733707926534675474030731815947615034382919338282057572099675517222408708454496136065407078228369414355168092642519214500764816393375699653280435011344821070000524165971770674359620284551501827999549634593324911255665543695557002
运算得到
aa=solve([p*p+q*q-nn,p-q+qjp],[p,q])
print(aa)
```

运算得到
```

```
[(-105033375278169563375831754572800771727698528611363430256008429712592445382422796983763719580920895628295349274745542274069695878184463909149800036636986 49,
-79318368677622540407274879818042999158099858738431387900331906668073450937791818254533466128772491620229836455961294487457133930353072560894507681769042 03),
(79318368677622540407274879818042999158099858738431387900331906668073450937791818254533466128772491620229836455961294487457133930353072560894507681769042 03,
105033375278169563375831754572800771727698528611363430256008429712592445382422796983763719580920895628295349274745542274069695878184463909149800036636986 49)]
```

取正

在套脚本求m

```
import gmpy2

q = 79318368677622540407274879818042999158099858738431387900331906668073450937791818254533466128772491620229836455961294487457133930353072560894507681769042 03
p = 10503337527816956337583175457280077172769852861136343025600842971259244538242279698376371958092089562829534927474554227406969587818446390914980003663698649
c=77245002688339814333680505807547811048373727479970953915983076561166535410858573370792653467547403073181594761503438291933828205757209967551722240870845449613606540707822836941435516809264251921450076481639337569965328043501134482107000052416597177067435962028455150182799954963459332491125566554369555700 29

e=8477419


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

d=gmpy2.invert(e,phin)

print(d)

m = pow(c,d,n)

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

```
917680104480318202648463700124935774593473991013351087505748944852439588072936023472661930678523605371831052142738598952388768770860668533085232230954151152944974329742195726524692684374405899424219141470150053238074273440052288677966800596045475781139800577134114633383471885806657253277269435962850759737914337636555117422867244651266930730447572000359923226860275116396608177965909155459540675956218620312842
0x666c61677b61326332646630342d353637352d343435392d613063662d343235316433633346566396 47d0a
b'flag{a2c2df04-5675-4459-a0cf-4251d3c4ef9d}\n'
```

得到flag提交即可

58.[NCTF2019]childRSA

查看题目

```python
from random import choice
from Crypto.Util.number import isPrime, sieve_base as primes
from flag import flag


def getPrime(bits):
    while True:
        n = 2
        while n.bit_length() < bits:
            n *= choice(primes)
        if isPrime(n + 1):
            return n + 1

e = 0x10001
m = int.from_bytes(flag.encode(), 'big')
p, q = [getPrime(2048) for _ in range(2)]
n = p * q
c = pow(m, e, n)

# n = 3284971819733758182300224371705765921850251900438699666088510059281
# c = 2630801835673985389538224010996889417516673128370292700216526899871
```

恩总结不会，愉快的找wp去

这一道RSA打开加密算法乍一看感觉没有问题，N特别大，除了p和q的生成算法啥都没给，

查了一下 Crypto.Util.number 中的sieve_base，发现这是前10000个素数的生成列表，我们再去查一下第10000个素数的值为104729

不过就算我们知道了这个值的大小似乎还是得不到结果，从这个p，q的生成算法中，我们可以知道其是由小于104729的素数随机组合生成的.

为小于p的任何数的倍数，即我们可以将这10000个素数乘起来就为p-1的倍数，于是尝试用费马小定理，即是a^(t*(p-1))-1为p的倍数，t*（p-1)通过上述算法得出即可

```
// python2
from Crypto.Util.number import sieve_base as primes
import gmpy2
n = 32849718197337581823002243717057659218502519004386996660885100592872201948834155543125924395614928962750579667346279456710633774501407292473006312537723894221717638059058796679686953564471994009285384798450493756900459225040360430847240975678450171551048783818642467506711424027848778367427338647282428667393241157151675410661015044633282064056800913282016363415202171926089293431012379261585078566301060173689328363696699811123592090204578098276704877408688525618732848817623879899628629300385790344366046641825507767709276622692835393219811283244303899850483748651722336996164724553364097066493953127153066970594638491950199605713033004684970381605908909693802373826516622872100822213645899846325022476318425889580091613323747640467299866189070780620292627043349618839126919699862580579994887507733838561768581933029077488033326056066378869170169389819542928899483936705521710423905128732013121538495096959944889076705471928490092476616709838980562233255542325528398956185421193665359897664110835645928646616337700617883946369110702443135980068553511927115723157704586595844927607636003501038871748639417378062348085980873502535098755568810971926925447913858894180171498580131088992227637341857123607600275137768132347158657063692388249513c = 26308018356739853895382240109968894175166731283702927002165268998773708335216338997058314157717147131083296551313334042509806229853341488461087009955203854253313827608275460592785607739091992591431080342664081962030557042784864074533380701014585315663218783130162376176094773010478159362434331787279303302718098735574605469803801873109982473258207444342330633191849040553550708886593340770753064322410889048135425025715982196600650740987076486540674090923181664281515197679745907830107684777248532278645343716263686014941081417914622724906314960249945105011301731247324601620886782967217339340393853616450077105125391982689986178342417223392217085276465471102737594719932347242482670320801063191869471318313514407997326350065187904154229557706351355052446027159972546737213451422978211055778164578782156428466626894026103053360431281644645515155471301826844754338802352846095293421718249819728205538534652212984831283642472071669494851823123552827380737798609829706225744376667082534026874483482483127491533474306552210039386256062116345785870668331513725792053302188276682550672663353937781055621860101624242216671635824311412793495965628876036344731733142759495348248970313655381407241457118743532311394697763283681852908564387282605279108t=pow(2,2048)e = 0x10001k=2for i in range(10000): k=pow(k,primes[i],n) if(k>t): if(i%15==0): if(gmpy2.gcd(k-1,n)!=1): print(gmpy2.gcd(k-1,n)) #178449493212694205742332078583256205058672290603652616240227340638730811945224947826121772642204629335108873832781921390308501763661154638696935732709724016546955977529088135995838497476350749621442719690722226913635772410880516639651363626821442456779009699333452616953193799328647446968707045304702547915799734431818800374360377292309248361548868909066895474518333089446581763425755389837072166970684877011663234978631869703859541876049132713490090720408351108387971577438951727337962368478059295446047962510687695047494480605473377173021467764495541590394732685140829152761532035790187269724703444386838656193674253139 break
p=gmpy2.gcd(k-1,n)
q=n//p
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
flag=hex(m)[2:].decode('hex')
print(flag)
#NCTF{Th3r3_ar3_1ns3cure_RSA_m0duli_7hat_at_f1rst_gl4nce_appe4r_t0_be_s3cur3}
```

59.[HDCTF2019]bbbbbbrsa

查看题目

enc.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
p = 177077389675257695042507998165006460849
n =
374218295098877962748971622493673294009886471456133253673379680633413727260 61
c =
==gMzYDNzIjMxUTNyIzNzIjMyYTM4MDM0gTMwEjNzgTM2UTN4cjNwIjN2QzM5ADMwIDNyMTO4UzM2cTM5kDN2MTOyUTO5YDM0czM3MjM
```



```
from base64 import b64encode as b32encode
from gmpy2 import invert,gcd,iroot
from Crypto.Util.number import *
from binascii import a2b_hex,b2a_hex
import random

flag = "******************************"

nbit = 128

p = getPrime(nbit)
q = getPrime(nbit)
n = p*q

print p
print n

phi = (p-1)*(q-1)

e = random.randint(50000,70000)

while True:
        if gcd(e,phi) == 1:
                break;
        else:
                e -= 1;

c = pow(int(b2a_hex(flag),16),e,n)

print b32encode(str(c))[::-1]

# 23737406995293649917635893242000934662067855618361018403816222372255122346 32
```

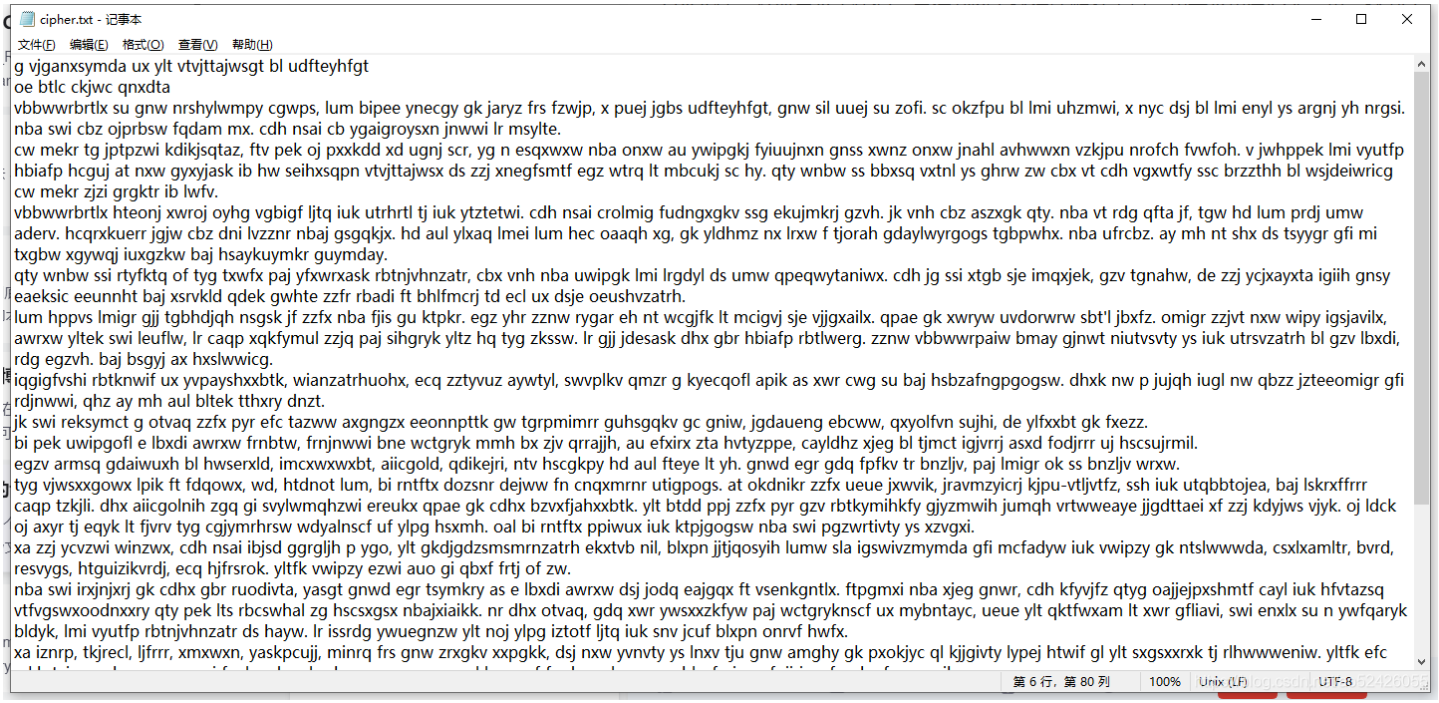
```python
#!/usr/bin/env python
# -*- coding:utf-8 -*-
from base64 import b64encode as b32encode
from base64 import b64decode
from gmpy2 import invert, gcd, iroot
from Crypto.Util.number import *

p = 177077389675257695042507998165006460849
n = 37421829509887796274897162249367329400988647145613325367337968063341372726061
c64 = '==gMzYDNzIjMxUTNyIzNzIjMyYTM4MDM0gTMwEjNzgTM2UTN4cjNwIjN2QzM5ADMwIDNyMTO4UzM2cTM5kDN2MTOyUTO5YDM0czM3MjM'
c = int ( b64decode ( str ( c64)[::-1] ) )
print ( c )
q = n // p
phi = (p - 1) * (q - 1)
for e in range ( 50000, 70000 ):
    if  gcd ( e, phi ) == 1:
        d = invert ( e, phi )
        m = pow ( c, d, n )
        flag=str(long_to_bytes(m))
        if 'flag' in flag or 'CTF' in flag or ("{" in flag and '}'in flag):
            print(flag)
```

```
'F\x8bk{h\x8c)\xe6*9\xf2\x03\xafy\xf9\xdb\x08\xc4T\x85\x96\x967<\x00q\x0f\xbe\xca\x0f}\xa2'
'*W0\xaey\xcd{{\xdcxp\x08\x16\xecs}\xd8\x7f\xdc3\x85{\xde_B\xc5\xdc3\x95i\xde\xa8'
'\x1f\x0b\x95\xc5\xa1\xc4Fi{\xe5f\xbf\xb46\xe8\x98\x05\xea\xc0\x1d\xab]\xc5\x87\xc5\x96}\xf81g"\xfa'
'flag{rs4_1s_s1mpl3!#}'
```

在运行结果里面手动找一下 即可找到flag

60. [MRCTF2020]vigenere

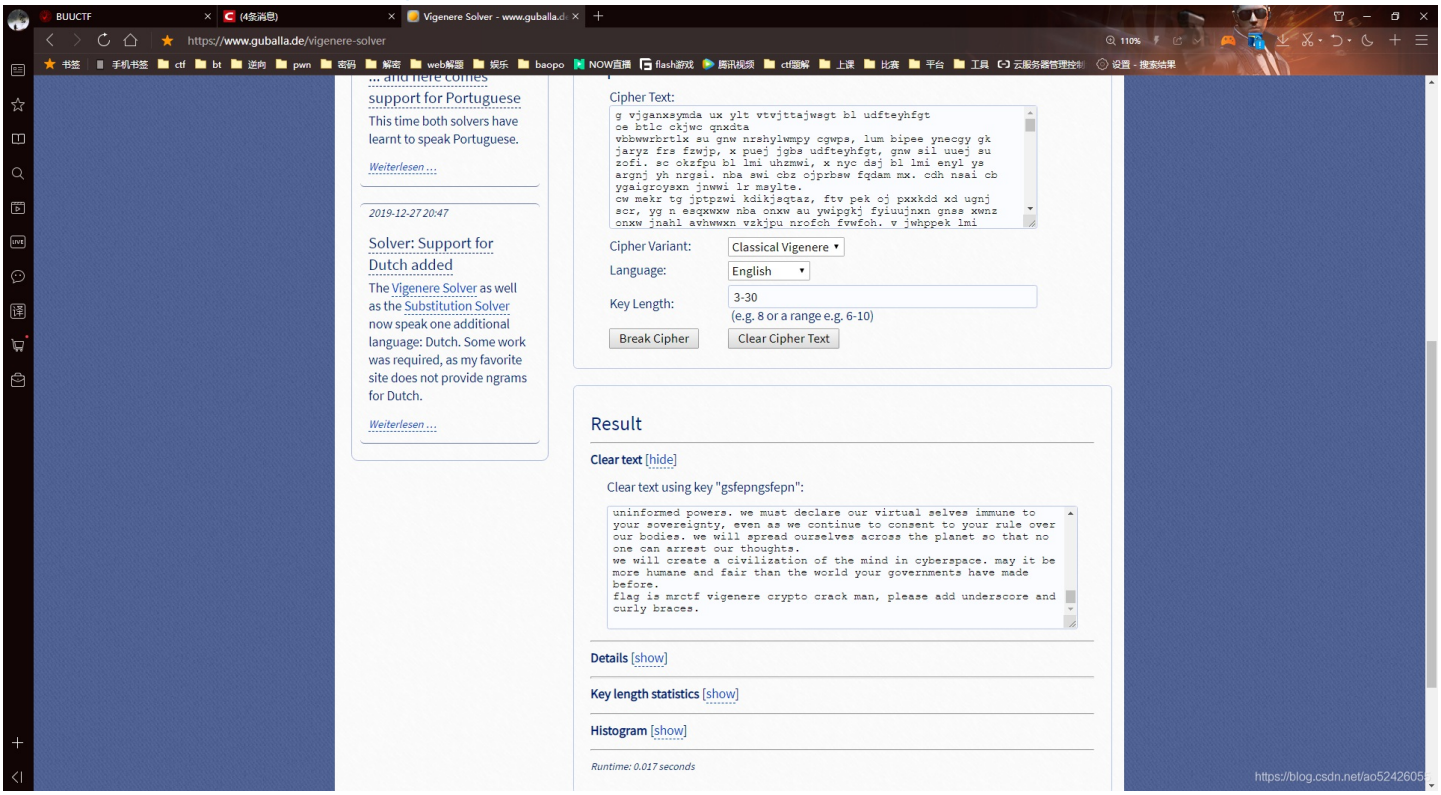
看题目




```python
#!/bin/python3
from ctf import source_text, key_string

getdiff = lambda char: ord(char)-ord('a')
getchar = lambda num: chr(ord('a')+num)
```

```python
def vigenere(src: chr, key: chr) -> chr:
    assert(src.isalpha() and key.isalpha())
    return(getchar((getdiff(src) + getdiff(key) + 1) % 26))

src = source_text.lower()
count = 0
assert(len(key_string) > 5 and len(key_string) < 10)
for i in src:
    if(i.isalpha()):
        print(vigenere(i, key_string[count % len(key_string)]), end='')
        count+=1
    else:
        print(i, end='')
```


弄了半天也没弄出来，去找大佬的wp结果就是一个在线解密



flag{vigenere_crypto_crack_man}

咱也不知道那个py是干嘛的，咱也不敢问。

暴打出题人哈哈哈

61.[BJDCTF2020]RSA

查看题目

```
from Crypto.Util.number import getPrime,bytes_to_long

flag=open("flag","rb").read()

p=getPrime(1024)
q=getPrime(1024)
assert(e<100000)
n=p*q
m=bytes_to_long(flag)
c=pow(m,e,n)
print c,n
print pow(294,e,n)

p=getPrime(1024)
n=p*q
m=bytes_to_long("BJD"*32)
c=pow(m,e,n)
print c,n

'''
output:
12641635617803746150332232646354596292707861480200207537199141183624438303757120570096741248020236666965755798009656547738616399025300123043766255518596149348930444599820675230046423373053051631932557230849083426859490183732303751744004874183062594856870318614289991675980063548316499486908923209627563871554875612702079100567018698992935818206109087568166097392314105717555482926141030505639571708876213167112187962584484065321545727594135175369233925922507794999607323536976824183162923385005669930403448853465141405846835919842908469787547341752365471892495204307644586161393228776042015534147913888338316244169120  13508774104460209743306714034546704137247627344981133461801953479736017021401725818808462898375994767375627749494839671944543822403059978073813122441407612530658168942987820256786583006947001711749230193542370570950705530167921702835627122401475251039000775017381633900222474727396823708695063136246115652622259769634591309421761269548260984426148824641285010730983215377509255011298737827621611158032976420011662547854515610597955628898073569684158225678333474543920326532893446849808112837476684390030976472053905069855522297850688026960701186543428139843783907624317274796926248829543413464754127208843070331063037381631268825806469518166370387352035475775677163615730759454343913563615970881967332407709901235637718936184198930226303761876517101208677107311006065728014220477966000620964056616058676999878976943319063836649085085377577273214792371548775204594097887078898598463892440141577974544939268247818937936607013100808169758675042264568547764031628431414727922168580998494695800403043312406643527637667466318473669542326169218665366423043579003388486634167642663495896607282155808331902351188500197960905672207046579647052764579411814305689137519860880916467272056778641442758940135016400808740387144508156358067955215018979153370552535153498477459720877329811204688208387543826122582132404214848454954722487086658061408795223805022202997613522014736983452121073860054851302343517756732701026667062765906277626879215457936330799698812755973057557620930172778859116538571207100424990838508255127616637334499680058645411786925302368790414768248611809358160197554369255458675450109457987698749584630551177577492043403656419968285163536823819817573531356497236154342689914525321673807925458651854768512396355389740863270148775362744448115581639629326236234216054850003500015609721544688125105550546571385417391314204097638250043518544252172112806210903061368369054309575159360374022344774547459345216907128193957592938071815865954073287532545947370671838372144806539753829484356064919357285623305209600680570975224639214396805124350862772159272362778768036844634760917612708721787320159318432456050806227784435091161119982613987303255995543165395426658059462110056431392517548717447898084915167661172362984251201688639469652283452307712821398857016487590794996544468826705600332208535201443322267298747117528882985955375246424812616478327182399461709978893464093245135530135430007842223389360212803439850867615121148050034887767584693608776323252233254261047
'''
```

两个n值公用了一个q，可以通过gcd函数很快找到q的值，就可以求出两个p的值，然后我们发现我们不知道e的值，然后e小于100000，又有关系式output=pow(294,e,n)，可以通过爆破e的取值很快得到e

```
// python2
from gmpy2 import *
from Crypto.Util.number import *
c1=126416356178037461503322326463545962927078614802002075371991411836244383037571205700967412480202366669657557980096565477386163990253001230437662555185961493489304445998206752300464233730530516319325572308490834268594901837323037517440048741830625948568703186142899916759800635483164994869089232096275638715548756127020791005670186989292358182061090875681660973923141057175554829261410305056395717088762131671121879625844840653215457275941351753692339259225077949996073235369768241831629233850056699304034488534651414058468359198429084697875473417523654718924952043076445861613932287760420155341479138883383162441691204
n1=135087741044602097433067140345467041372476273449811334618019534797360170214017258188084628983759947673756277494948396719445438224030599780738131224414076125306581689429878202567865830069470017117492301935423705709507055301679217028356271224014752510390007750173816339002224747273968237086950631362461156526222597696345913094217612695482609844261488246412850107309832153775092550112987378276216111580329764200116625478545156105979556288980735696841582256783334745439203265328934468498081128374766843900309764720539050698555222978506880269607011865434281398437839076243172747969262488295434134647541272088430703310630372c2=979153370552535153498477459720877329811204688208387543826122582132404214848454954722487086658061408795223805022202997613522014736983452121073860054851302343517756732701026667062765906277626879215457936330799698812755973057557620930172778859116538571207100424990838508255127616637334499680058645411786925302368790414768248611809358160197554369255458675450109457987698749584630551177577492043403656419968285163536823819817573531356497236154342689914525321673807925458651854768512396355389740863270148775362744448115581639629326362342160548500035000156097215446881251055505465713854173913142040976382500435185442521721n2=128062109030613683690543095751593603740223447745474593452169071281939575929380718158659540732875325459473706718383721448065397538294843560649193572856233052096006805709752246392143968051243508627721592723627787680368446347609176127087217873201593184324560508062277844350911611199826139873032559955431653954266580594621100564313925175487174478980849151676611723629842512016886394696522834523077128213988570164875907949965444688267056003322085352014433222672987471175288829859553752464248126164783271823994617099788934640932451355301354300078422233893602128034398508676151211480500348877675846936087763232522332542610471047q=gcd(n1,n2)#print(q)#998553537617649393082659514921169767986746812829414625169565777129437178500480512733587450959062070851709157941877499545886858504521621650598317493034731065419309487230008827134536799045256553271686652952074232579226667210777479118601591810414229930306183854365048589436156302194592624197158163617810628989llc_1 = 38163126882580646951816637038735203547577567716361573075945434391356361597088196733240770990123563771893618419893022630376187651710120867710731100606572801422047796600062096405661605867699987897694331906383664908508537757727321479237154877520459409788707889859846389244014157797454493926824781893793660701310080816975867504226456854776403162843141472792216858099849469580040304331240664352763766746631847366954232616921866536642304357900338848663416764266349589660728215580833190235118850019796090567220704657964705276457941181430568913751986088091646727205677864144275894013501640080874038714450815635806795521501816for i in range(100000):
 res=pow(294,i,n1)
 if (res==output):
  #print(i)
  #52361
  e=i
  break
e=52361
p=n1//q
phi=(p-1)*(q-1)
d=invert(e,phi)
m=pow(c1,d,n1)
flag=long_to_bytes(m)
print(flag)
```

运行得到结果 flag{p_is_common_divisor}

62.一张谍报

查看题目

国家能源时报 2015 年 3 月 5 日

平时要针对性的吃些防辐射菜

对于和电脑“朝夕相处”的人们来说，辐射的确是个让人忧心的“副产物”。因此，平时针对性的吃些可以防辐射的菜是很有好处的。特别是现在接近年底，加班加点是家常便饭，对着电脑更是辐射吸收得满满的，唯有趁一日三餐进食的时候吃点防辐射的食物了。

朝歌区梆子公司三更放炮

老小区居民大爷联合抵制

今天上午，朝歌区梆子公司决定，在每天三更天不亮免费在各大小区门口设卡为全城提供二次震耳欲聋的敲更提醒，呼吁大家早睡早起，不要因为贪睡断送大好人生，时代的符号是前进。为此，全区老人都蹲在该公司东边树丛合力抵制，不给公司人员放行，场面混乱。李罗鹰住进朝歌区五十年了，人称老鹰头，几年孙子李虎南刚从东北当猎户回来，每月还寄回来几块鼹鼠干。李罗鹰当年遇到的老婆是朝歌一枝花，所以李南虎是长得非常秀气的一个汉子。李罗鹰表示：无论梆子公司做的对错，反正不能打扰他孙子睡觉，子曰：‘睡觉乃人之常情’。梆子公司这是连菩萨睡觉都不放过啊。李南虎表示：梆子公司智商捉急，小心居民猴急跳墙！这三伏天都不给睡觉，这不扯淡么！

到了中午人群仍未离散，更有人提议要烧掉这个公司，公司高层似乎恨不得找个洞

朝歌区梆子公司三更放炮

老小区居民大爷联合抵制

喵天上午，汪歌区哞叽公司决定，在每天八哇天不全免费在各大小区门脑设卡为全城提供双次震耳欲聋的敲哇提醒，呼吁大家早睡早起，不要因为贪睡断送大好人生，时代的编号是前进。为此，全区眠人都足在该公司流边草丛合力抵制，不给公司人员放行，场面混乱。李罗鸟住进汪歌区五十年了，人称眠鸟顶，几年孙叽李熬值刚从流北当屁户回来，每月还寄回来几块报信干。李罗鸟当年遇到的眠婆是汪歌一枝花，所以李值熬是长得非常秀气的一个汉叽。李罗鸟表示：无论哞叽公司做的对错，反正不能打扰他孙叽睡觉，叽叶：‘睡觉乃人之常情’。哞叽公司这是连衣服睡觉都不放过啊。李值熬表示：哞叽公司智商捉急，小心居民猴急跳墙！这八伏天都不给睡觉，这不扯淡么！

到了中午人群仍未离散，哇有人提议要烧掉这个公司，公司高层似乎恨不得找个洞钻进去。直到治安人员出现才疏散人群归家，但是李值熬仍旧表示爷爷年纪大了，睡不好对身体不好。

听书做作业

喵汪哞叽双哇顶，眠鸟足屁流脑，八哇报信断流脑全叽，眠鸟进北脑上草，八枝遇孙叽，孙叽对熬编叶：值天衣服放鸟捉猴顶。鸟对：北汪罗汉伏熬乱天门。合编放行，卡编扯呼。人离烧草，报信归洞，孙叽找爷爷。

啥也不是，原理我暂时没弄懂，后期补上。

利用python3大佬写的脚本就能解出：

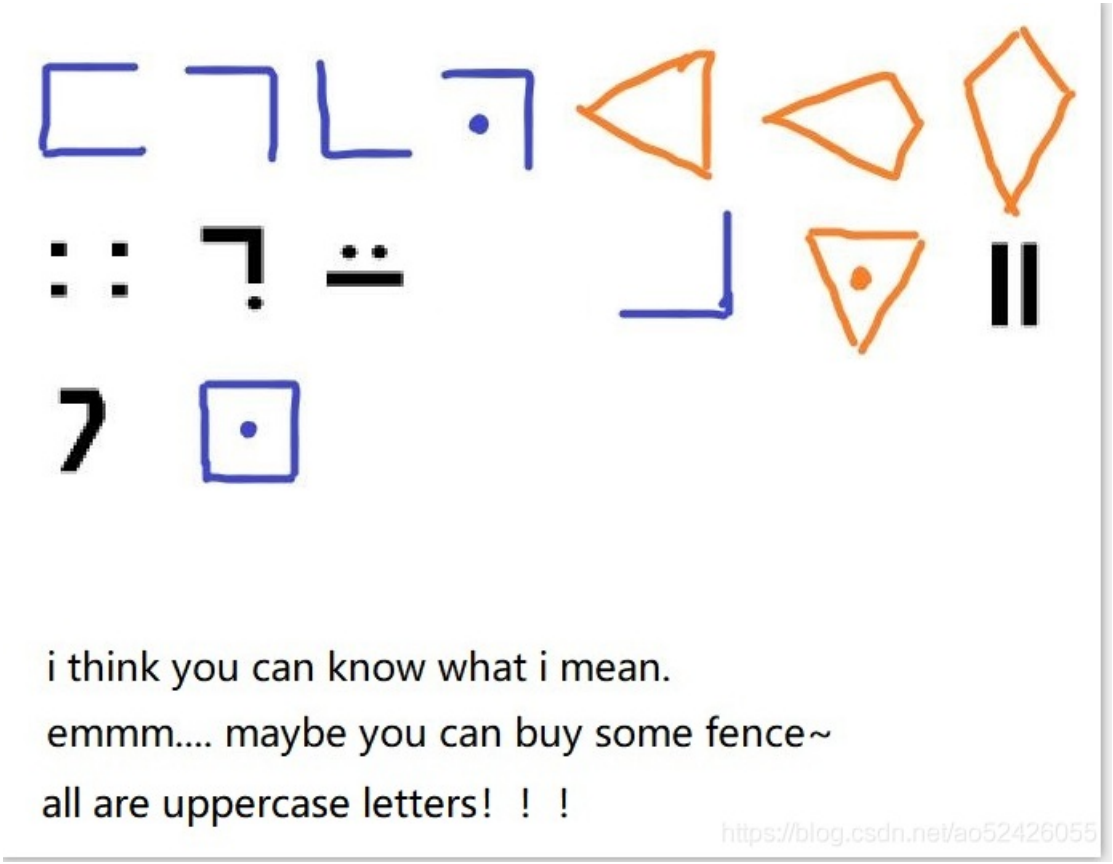
```
str1 = "今天上午，朝歌区梆子公司决定，在每天三更天不亮免费在各大小区门口设卡为全城提供二次震耳欲聋的敲更提醒，呼吁大家早睡早起，不要因为贪睡断送大好人生，时代的符号是前进。为此，全区老人都蹲在该公司东边树丛合力抵制，不给公司人员放行，场面混乱。李罗鹰住进朝歌区五十年了，人称老鹰头，几年孙子李虎南刚从东北当猎户回来，每月还寄回来几块鼹鼠干。李罗鹰当年遇到的老婆是朝歌一枝花，所以李南虎是长得非常秀气的一个汉子。李罗鹰表示：无论梆子公司做的对错，反正不能打扰他孙子睡觉，子曰：‘睡觉乃人之常情’。梆子公司这是连菩萨睡觉都不放过啊。李南虎表示：梆子公司智商捉急，小心居民猴急跳墙！这三伏天都不给睡觉，这不扯淡么！到了中午人群仍未离散，更有人提议要烧掉这个公司，公司高层似乎恨不得找个洞钻进去。直到治安人员出现才疏散人群归家，但是李南虎仍旧表示爷爷年纪大了，睡不好对身体不好。"
str2 = "喵天上午，汪歌区哗叽公司决定，在每天八哇天不全免费在各大小区门脑设卡为全城提供双次震耳欲聋的敲哇提醒，呼吁大家早睡早起，不要因为贪睡断送大好人生，时代的编号是前进。为此，全区眠人都足在该公司流边草丛合力抵制，不给公司人员放行，场面混乱。李罗鸟住进汪歌区五十年了，人称眠鸟顶，几年孙叽李熬值刚从流北当屁户回来，每月还寄回来几块报信干。李罗鸟当年遇到的眠婆是汪歌一枝花，所以李值熬是长得非常秀气的一个汉叽。李罗鸟表示：无论哗叽公司做的对错，反正不能打扰他孙叽睡觉，叽叶：‘睡觉乃人之常情’。哗叽公司这是连衣服睡觉都不放过啊。李值熬表示：哗叽公司智商捉急，小心居民猴急跳墙！这八伏天都不给睡觉，这不扯淡么！到了中午人群仍未离散，哇有人提议要烧掉这个公司，公司高层似乎恨不得找个洞钻进去。直到治安人员出现才疏散人群归家，但是李值熬仍旧表示爷爷年纪大了，睡不好对身体不好。"
str3 = "喵汪哗叽双哇顶，眠鸟足屁流脑，八哇报信断流脑全叽，眠鸟进北脑上草，八枝遇孙叽，孙叽对熬编叶：值天衣服放鸟捉猴顶。鸟对：北汪罗汉伏熬乱天门。合编放行，卡编扯呼。人离烧草，报信归洞，孙叽找爷爷。"
res = ""
for i in range(len(strs3)):
    for j in range(len(strs2)):
        if strs3[i] == strs2[j]:
            res += strs1[j]
            break
print(res)
```

运行脚本得到：
今朝梆子二更头，老鹰蹲猎东口，三更鼹鼠断东口亮子，老鹰进北口上树，三枝遇孙子，孙子对虎符曰：南天菩萨放鹰捉猴头。鹰对：北朝罗汉伏虎乱天门。合符放行，卡符扯呼。人离烧树，鼹鼠归洞，孙子找爷爷。

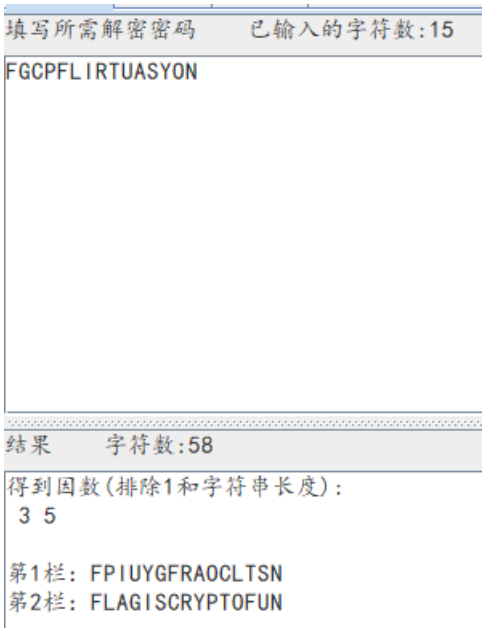
flag为{南天菩萨放鹰捉猴头}

63.[MRCTF2020]古典密码知多少

查看题目



翻译：
我想你能明白我的意思，
嗯...。也许你可以买一些围栏~都是大写字母！
一张图，标准银河字母+圣堂武士+猪圈变形，在网上找密码表对照解出 `FGCPFLIRTUASYON`
上面也说了 买一些围栏，在栅栏解密呗



flag{CRYPTOFUN}

64.[MRCTF2020]天干地支+甲子

查看题目

得到得字符串用MRCTF{}包裹
一天Eki收到了一封来自Sndav的信，但是他有点迷希望您来解决一下
甲戌
甲寅
甲寅
癸卯
己酉
甲寅
辛丑

网上找天干地支对应的数字表

01 甲子	11甲戌	21甲申	31 甲午	41甲辰	51甲寅
02 乙丑	12 乙亥	22乙酉	32 乙未	42乙巳	52 乙卯
03 丙寅	13丙子	23 丙戌	33丙申	43丙午	53丙辰
04丁卯	14丁丑	24丁亥	34丁酉	44丁未	54丁巳
05戊辰	15戊寅	25戊子	35戊戌	45戊申	55 戊午
06己巳	16己卯	26己丑	36 己亥	46 己酉	56 己未
07庚午	17 庚辰	27庚寅	37 庚子	47 庚戌	57 庚申
08辛未	18 辛巳	28 辛卯	38 辛丑	48 辛亥	58 辛酉
09壬申	19壬午	29 壬辰	39 壬寅	49 壬子	59 壬戌
10癸酉	20癸未	30 癸巳	40 癸卯	50 癸丑	60 癸亥

都加60

甲戌 23 83
甲寅 51 111
甲寅 51 111
癸卯 40 100
己酉 46 106
甲寅 51 111
辛丑 38 98

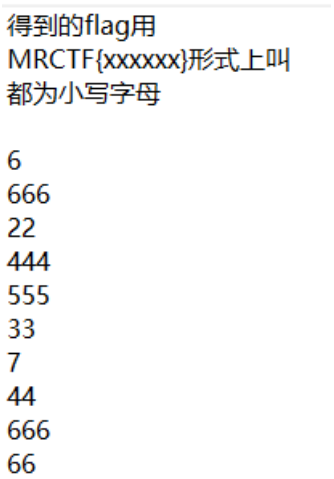
对照ascll码表

0	0	nul	20	32	sp	40	64	@	60	96	`
1	1	soh	21	33	!	41	65	A	61	97	a
2	2	stx	22	34	"	42	66	B	62	98	b
3	3	etx	23	35	#	43	67	C	63	99	c
4	4	eot	24	36	$	44	68	D	64	100	d
5	5	enq	25	37	%	45	69	E	65	101	e
6	6	ack	26	38	&	46	70	F	66	102	f
7	7	bel	27	39	`	47	71	G	67	103	g
8	8	bs	28	40	(	48	72	H	68	104	h
9	9	ht	29	41	)	49	73	I	69	105	i
A	10	nl	2A	42	*	4A	74	J	6A	106	j
B	11	vt	2B	43	+	4B	75	K	6B	107	k
C	12	ff	2C	44	,	4C	76	L	6C	108	l
D	13	cr	2D	45	-	4D	77	M	6D	109	m
E	14	so	2E	46	.	4E	78	N	6E	110	n
F	15	si	2F	47	/	4F	79	O	6F	111	o
10	16	dle	30	48	0	50	80	P	70	112	p
11	17	dc1	31	49	1	51	81	Q	71	113	q
12	18	dc2	32	50	2	52	82	R	72	114	r
13	19	dc3	33	51	3	53	83	S	73	115	s
14	20	dc4	34	52	4	54	84	T	74	116	t
15	21	nak	35	53	5	55	85	U	75	117	u
16	22	syn	36	54	6	56	86	V	76	118	v
17	23	etb	37	55	7	57	87	W	77	119	w
18	24	can	38	56	8	58	88	X	78	120	x
19	25	em	39	57	9	59	89	Y	79	121	y
1A	26	sub	3A	58	:	5A	90	Z	7A	122	z
1B	27	esc	3B	59	;	5B	91	[	7B	123	{
1C	28	fs	3C	60	<	5C	92	\	7C	124	|
1D	29	gs	3D	61	=	5D	93	]	7D	125	}
1E	30	re	3E	62	>	5E	94	^	7E	126	~
1F	31	us	3F	63	?	5F	95	_		127	del

flag{Goodjob}

65.[MRCTF2020]keyboard

查看题目



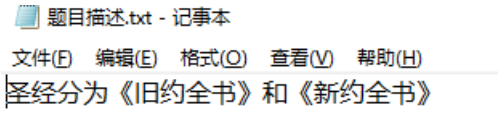
对照九键键盘，重复次数就是某个按键的第几个字母，mobilephone。

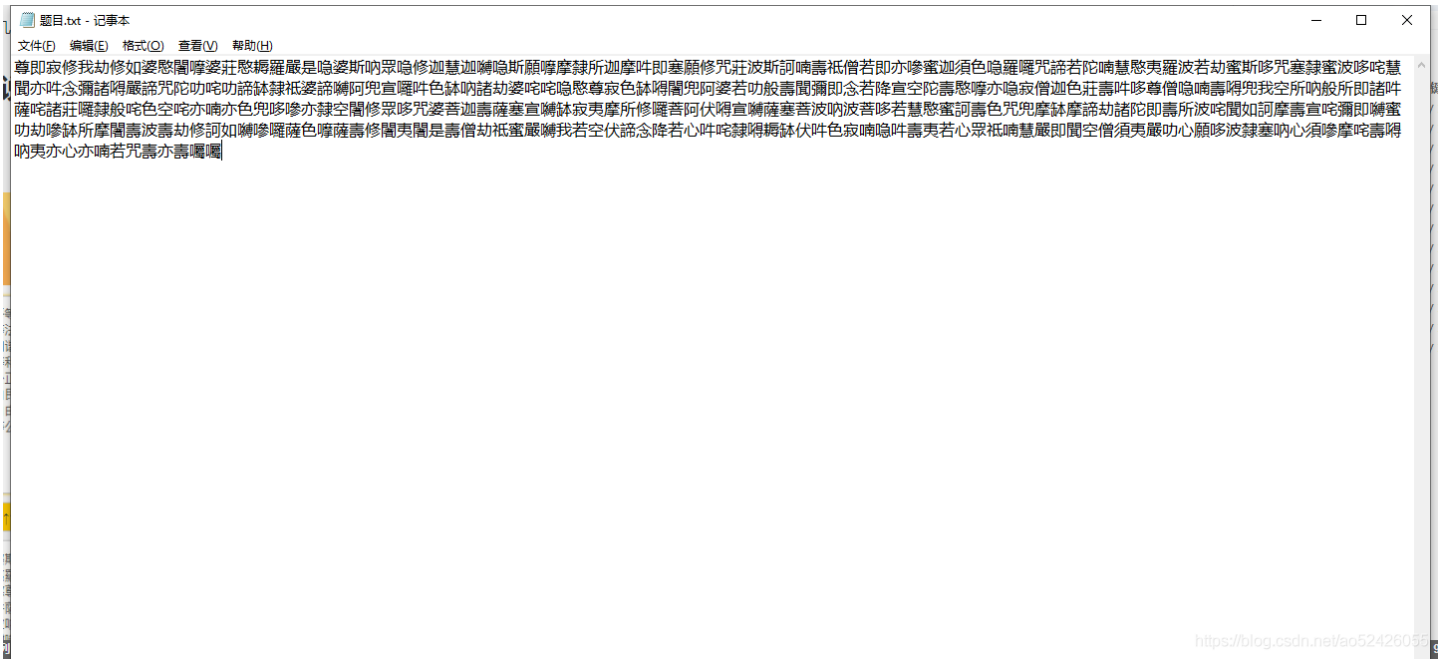
flag{mobilephone}

66.[WUSTCTF2020]佛说：只能四天

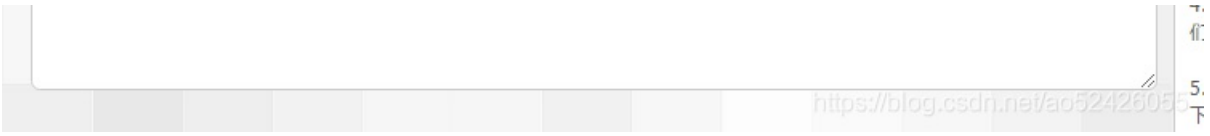
看题目

1. 虽然有点不环保，但hint好像是一次性的，得到后就没有利用价值了。
2. 凯撒不是最后一步，by the way，凯撒为什么叫做凯撒?





先把佛论解密一下

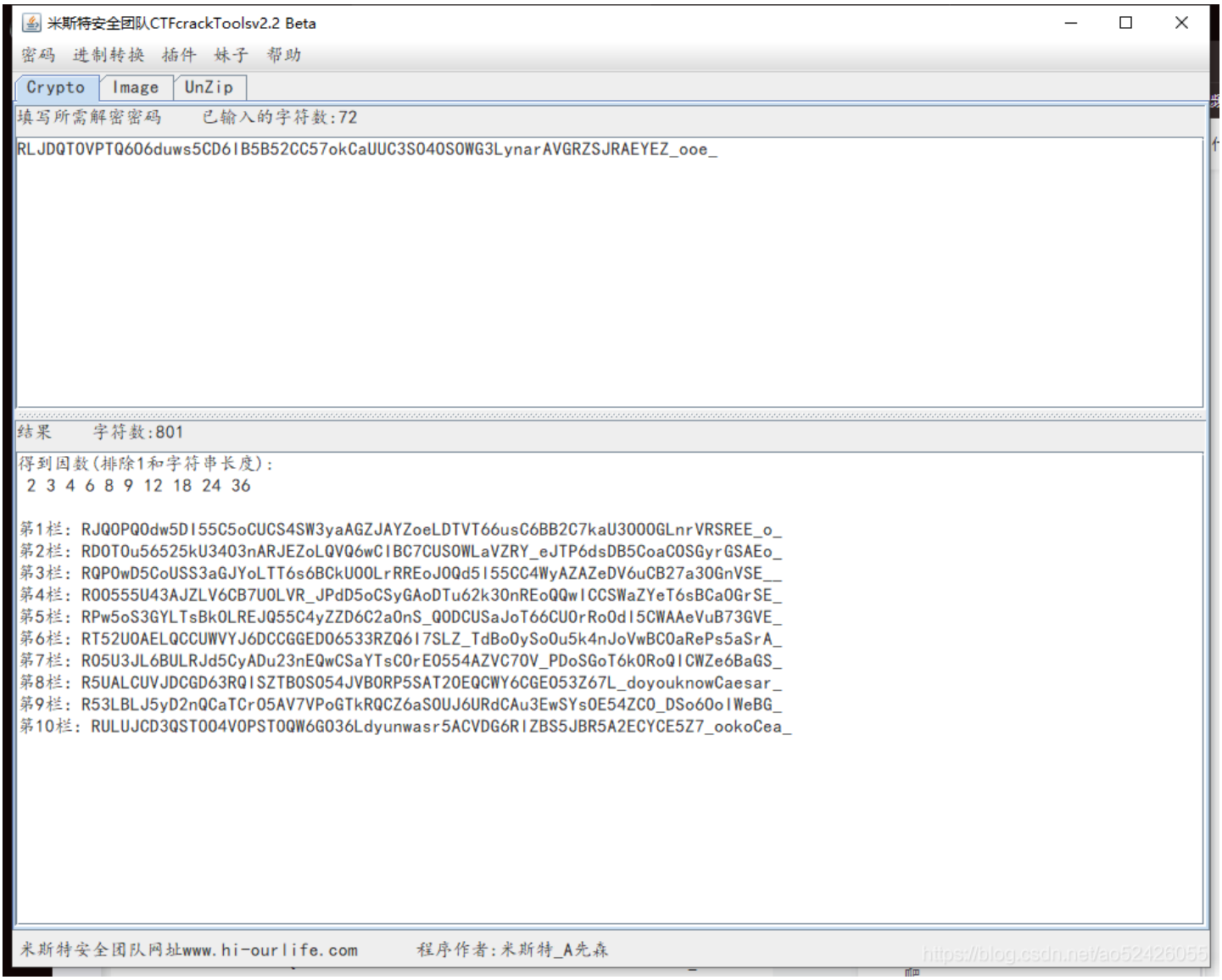


得到核心价值编码解码

`RLJDQTOVPTQ6O6duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_doyouknowfence`

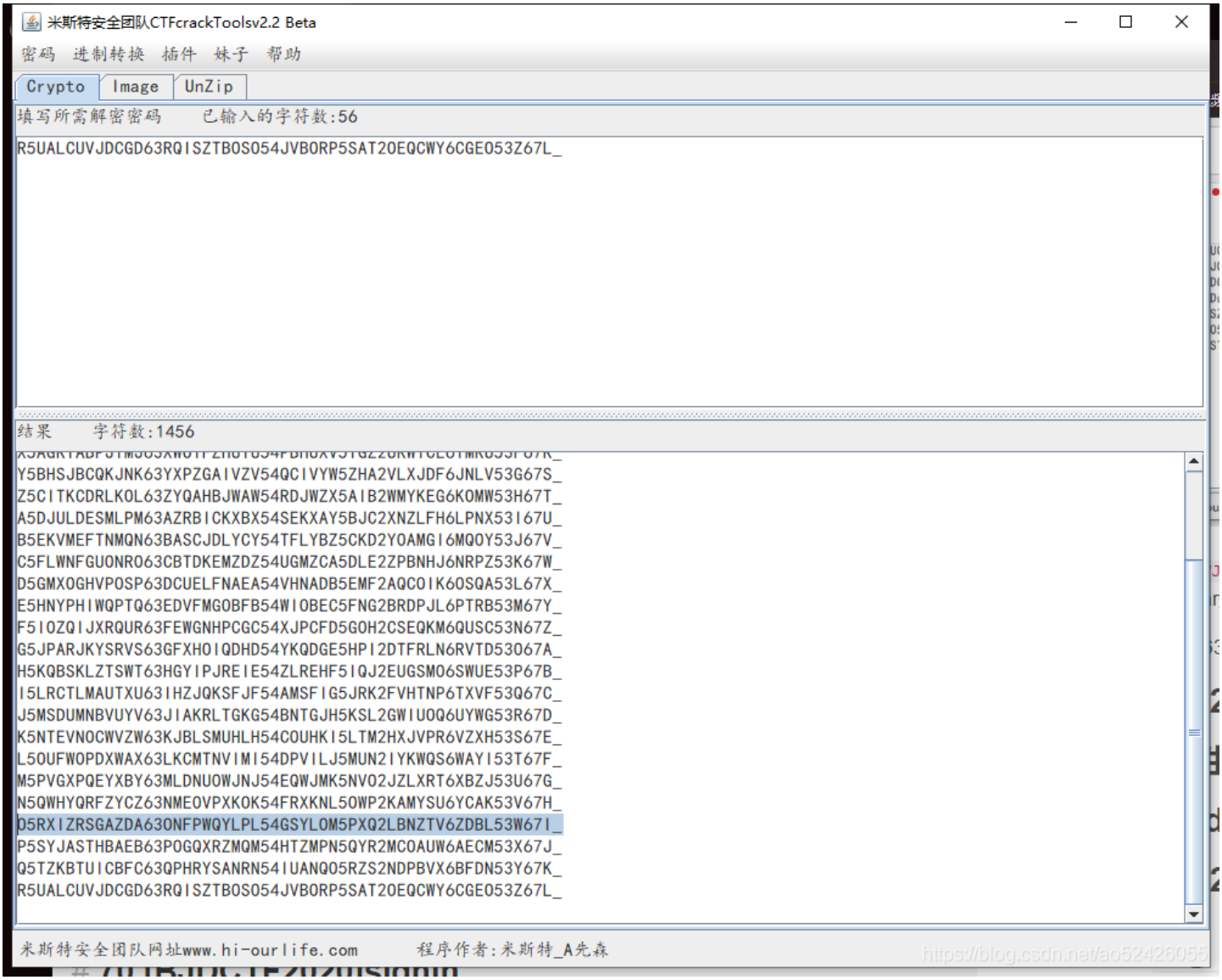
从doyouknowfence知道栅栏

解密



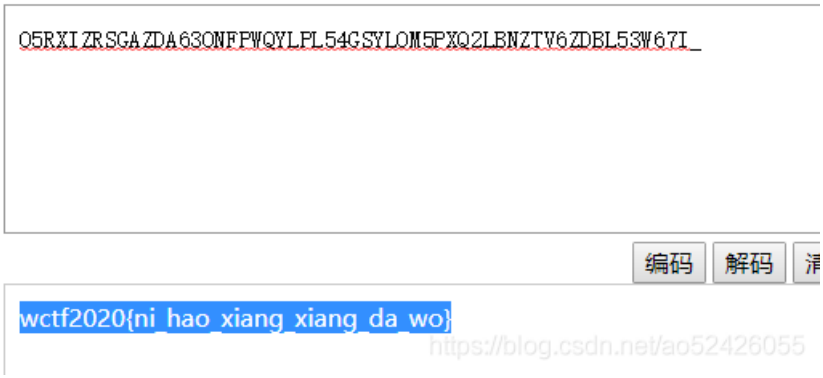
又看到一个`R5UALCUVJDCGD63RQISZTBOSO54JVBORP5SAT2OEQCWY6CGEO53Z67L_doyouknowCaesar_`

从doyouknowCaesar知道凯撒（位移为3叫凯撒，我们直接找到位移为3的）



O5RXIZRSGAZDA63ONFPWQYLPL54GSYLOM5PXQ2LBNZTV6ZDBL53W67I_

base32解密



得到wctf2020{ni_hao_xiang_xiang_da_wo}

67.[BJDCTF2020]rsa_output

查看题目

```
{210583393373542878475341075446136053050154410905089240941988166912191033995268001128024163830889952539088574602
667269256158268953033778016148293640346244751958599979431463055883159391307774504851962907662496123400543546225 1
620768154297375625767738809192654965516249087384995578376866302913864707987427824086793212719668625880014691162 0
730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362 5
283466847529599374738526301458937960566757936464307935782654182559193763237960445885597267038584293117847052450 6
984593831680268157565365377088361552573569030667463516711 1,2767}

{210583393373542878475341075446136053050154410905089240941988166912191033995268001128024163830889952539088574602
667269256158268953033778016148293640346244751958599979431463055883159391307774504851962907662496123400543546225 1
620768154297375625767738809192654965516249087384995578376866302913864707987427824086793212719668625880014691162 0
730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362 5
283466847529599374738526301458937960566757936464307935782654182559193763237960445885597267038584293117847052450 6
984593831680268157565365377088361552573569030667463516711 1,3659}

message1=201524901655224017477231939669021811510987317639980574219671553009337193782163420437308013025349784037 4
108688796904072195953319005834276205735943266371782582636544499691546903905642841616617392095824304483140492411 3
442512617599426876141184212121677500371236937127571802891321706587610393639446868836987170301813018218408886968 2
638821230841556074940763302569342851713707585865354151361628611388987289105851383788845308198574786097911269713 0
862431845490599291940535575149278911000931313841726512611727371081384392314338127620480251591052746888322427482 9
962479636527422350190210717694762908096944600267033351813929448599

message2=112986973231409888120577353242859084805047214541457965350144187389590352456006799472978745178189281815 0
908154502705652379002259823391801126101197319638639568937152677478558232612195918619558606985159246763781936662 4
044133661016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060 3
634829587083647269829087983401822887021010233938397814273865372304594365126130473115858750680082108189969414601 5
658931413501043836244752242820688494495263982667724781906681270683577310705956708282231230072104982701366041861 0
265189288840247186598145741724084351633508492707755206886202876227
```

RSA共模攻击

python2脚本

```
from gmpy2 import invert
def gongmogongji(n, c1, c2, e1, e2):
    def egcd(a, b):
        if b == 0:
            return a, 0
        else:
            x, y = egcd(b, a % b)
            return y, x - (a // b) * y
    s = egcd(e1, e2)
    s1 = s[0]
    s2 = s[1]

    if s1 < 0:
        s1 = - s1
        c1 = invert(c1, n)
    elif s2 < 0:
        s2 = - s2
        c2 = invert(c2, n)
    m = pow(c1, s1, n) * pow(c2, s2, n) % n
    return m

n1=21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111
e1=2767

n2=21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111
e2=3659

message1=20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403741086887969040721959533190058342762057359432663717825826365444996915469039056428416166173920958243044831404924113442512617599426876141184212121677500371236937127571802891321706587610393639446868836987170301813018218408886968263882123084155607494076330256934285171370758586535415136162861138898728910585138378884530819857478609791126971308624318454905992919405355751492789110009313138417265126117273710813843923143381276204802515910527468883224274829962479636527422350190210717694762908096944600267033351813929448599
message2=11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181509081545027056523790022598233918011261011973196386395689371526774785582326121959186195586069851592467637819366624044133661016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060363482958708364726982908798340182288702101023393839781427386537230459436512613047311585875068008210818996941460156589314135010438362447522428206884944952639826677247819066812706835773107059567082822312300721049827013660418610265189288840247186598145741724084351633508492707755206886202876227
m=gongmogongji(n1,message1,message2,e1,e2)
print(hex(m)[2:].decode('hex'))
```

运行得到

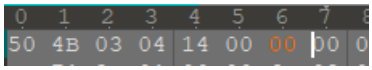
BJD{r3a_C0mmoN_moD@_4ttack}

换成flag包裹即可

68.[ACTF新生赛2020]crypto-rsa0

查看题目

解压说有密码，用010打开发现是伪密码



解压得到

```
90185880664342063772402771624767392713862401730886765262953151639909683470229228412991282745514829264909083992371538834949647434361938539784599470602104117540700567387773825783572976003776521334003669635076632422914361317993214512213068577850406241013704363595820880569869816984729352014957260502649275174022350996206925961019415256003394743594106061473865032792073035954925875056079762626648452348856255575840166640519334862690063949316515750256545937498213476286637455803452890781264446030732369871044870359838568618176586206041055000297981733272816089806014400846392307742065559331874972274844992047849472203390350
```

上脚本

```
p=90185880664342063772402771624767392713862401730886765262953151639909683470229228412991282745514829264909083992371538834949647434361938539784599470602104117
q=75470056738777382578357297600377652133400366963507663242291436131799321451221306857785040624101370436359582088056986981698472935201495726050264927517402235099620692596101941525600339474359410606147386503279207303595492587505607976262664845234885625557584016664051933486269006394931651575025654593749821347628663745580345289078126444603073236987104487035983856861817658620604105550002979817332728160898060144008463923077420655593318749722748449920478494722033903507
```

运行得到

```
D:\python38\python.exe D:/pycharm/venv/mima/RSA/crypto-rsao.py
b'actf{n0w_y0u_see_RSA}'
```

flag{n0w_y0u_see_RSA}

69.SameMod

查看题目

```
{6266565720726907265997241358331585417095726146341989755538017122981360742813498401533594757088796536341941659691259323065631249,773}
{6266565720726907265997241358331585417095726146341989755538017122981360742813498401533594757088796536341941659691259323065631249,839}

message1=34535205927234439354511515452450258642323888717216823264089150243498040620419767023647286606829123969039681939811311553111537349
message2=56728180268162933440701193325366296194571635700363052968690535322931053796907933860190657544652928677695217364141708032383095350
```

这道题一看题目SameMod就可以猜到是RSA中的共模攻击
关于共模攻击的原理这里就不多赘述了，可以参考ctfwiki
python2脚本

```
// python2
from gmpy2 import invert
def gongmogongji(n, c1, c2, e1, e2):
    def egcd(a, b):
        if b == 0:
            return a, 0
        else:
            x, y = egcd(b, a % b)
            return y, x - (a // b) * y
    s = egcd(e1, e2)
    s1 = s[0]
    s2 = s[1]
    if s1 < 0:
        s1 = - s1
        c1 = invert(c1, n)
    elif s2 < 0:
        s2 = - s2
        c2 = invert(c2, n)
    m = pow(c1, s1, n) * pow(c2, s2, n) % n
    return m

n= 626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124
9
e1= 773
e2= 839
c1= 345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398113155311153734
9
c2= 567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323830953
5

result = gongmogongji(n, c1, c2, e1, e2)
print(result)
#10210897103123119104101110119101116104105110107105116105115112111115115105981081011125
#flag=hex(result)[2:].decode('hex')
result=str(result)
flag=""
i=0
while i < len(result):
    if result[i]=='1':
        c=chr(int(result[i:i+3]))
        i+=3
    else:
        c=chr(int(result[i:i+2]))
        i+=2
    flag+=c
print(flag)
```

flag{whenwethinkitispossible}

70.[BJDCTF2020]signin

看题目

我当时弄了好久最后一个十六进制转字符
嗯没错就这么简单

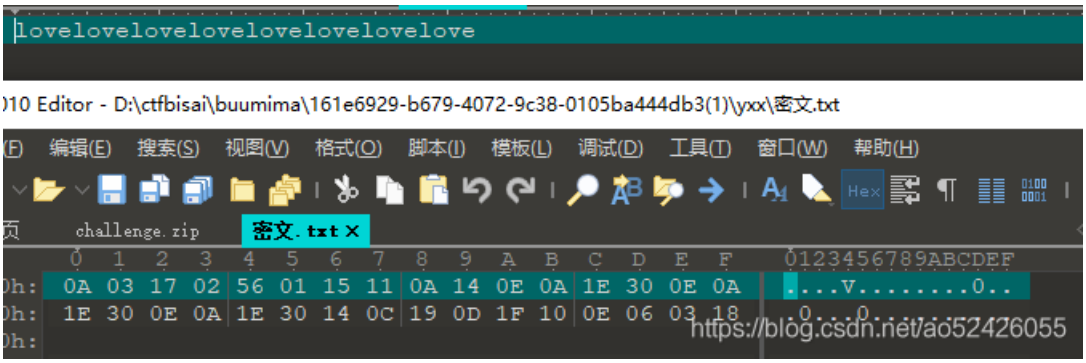
flag{We1c0me_t4_BJDCTF}

71.yxx

查看题目

啥也不是没看懂

用010查看进制



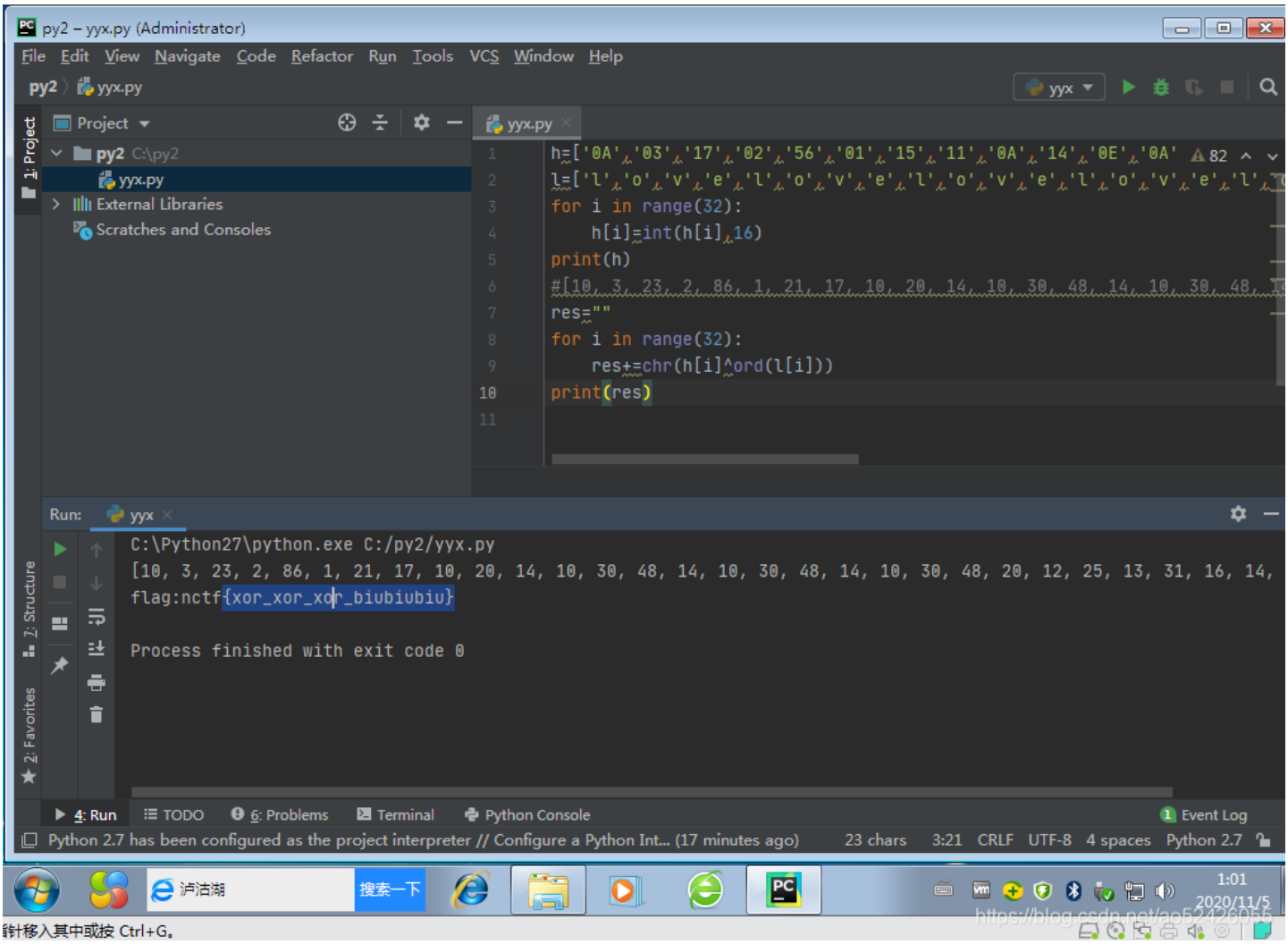
32位，再来查看明文.txt发现刚好也是32位字符，于是怀疑这道题是道一次性密码本OTP的题目，将明文与密文的txt一位一位异或即可得到flag

上脚本

python2的

```
h=['0A','03','17','02','56','01','15','11','0A','14','0E','0A','1E','30','0E','0A','1E','30','0E','0A','1E','30'
,'14','0C','19','0D','1F','10','0E','06','03','18']
l=['l','o','v','e','l','o','v','e','l','o','v','e','l','o','v','e','l','o','v','e','l','o','v','e','l','o','v','e','l','o','v','e','l','o','v','e']
for i in range(32):
    h[i]=int(h[i],16)
print(h)
res=""
for i in range(32):
    res+=chr(h[i]^ord(l[i]))
print(res)
```

运行得到



flag{xor_xor_xor_biubiubiu}

72.[AFCTF2018]Vigenère

就这名字就说明了一切了

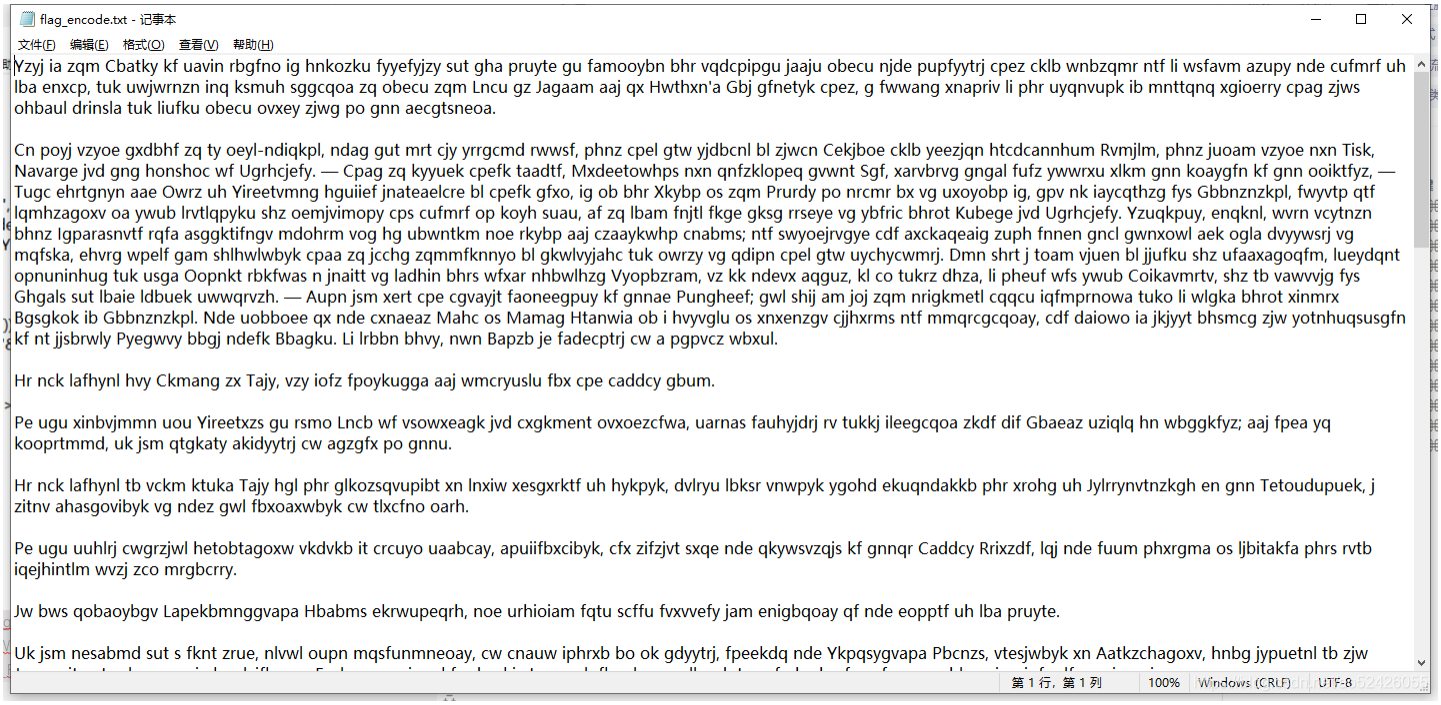
在看题目

Encode.c - 记事本

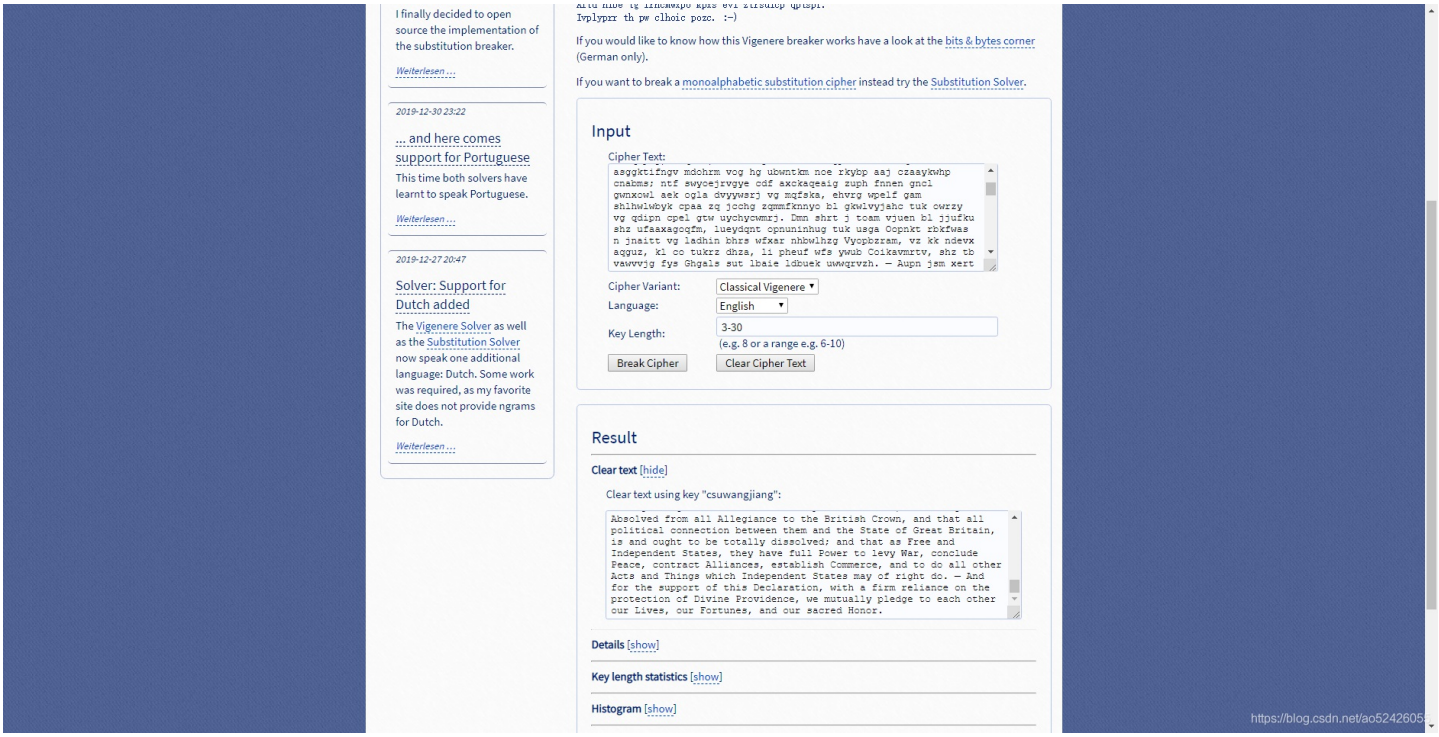
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
        freopen("flag.txt","r",stdin);
        freopen("flag_encode.txt","w",stdout);
        char key[] = /*SADLY SAYING! Key is eaten by Monster!*/;
        int len = strlen(key);
        char ch;
        int index = 0;
        while((ch = getchar()) != EOF){
                if(ch>='a'&&ch<='z'){
                        putchar((ch-'a'+key[index%len]-'a')%26+'a');
                        ++index;
                }else if(ch>='A'&&ch<='Z'){
                        putchar((ch-'A'+key[index%len]-'a')%26+'A')
                        ++index;
                }else{
```

```
                }else{
                    putchar(ch);
                }
            }
            return 0;
}
```

就这个题目，我们直接不管第一个，把第二个仍在vigenere暴力破解



破解得到，我们直接CTRL+F搜索flag

```
flag is afctf{Whooooooo_U_Gotcha!}
```

flag包裹提交即可

73.[GWCTF 2019]BabyRSA

查看题目

```
import hashlib
import sympy
from Crypto.Util.number import *

flag = 'GWHT{******}'
secret = '******'

assert(len(flag) == 38)

half = len(flag) / 2

flag1 = flag[:half]
flag2 = flag[half:]

secret_num = getPrime(1024) * bytes_to_long(secret)

p = sympy.nextprime(secret_num)
q = sympy.nextprime(p)

N = p * q

e = 0x10001

F1 = bytes_to_long(flag1)
F2 = bytes_to_long(flag2)

c1 = F1 + F2
c2 = pow(F1, 3) + pow(F2, 3)
assert(c2 < N)

m1 = pow(c1, e, N)
m2 = pow(c2, e, N)

output = open('secret', 'w')
output.write('N=' + str(N) + '\n')
output.write('m1=' + str(m1) + '\n')
output.write('m2=' + str(m2) + '\n')
output.close()
```

N=6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414136400065276148731106514101558937765487378231529437978847291301497582791274300447392540004266109228345730949570825895394456108282794288145243134912620619305128290744662326331305991044908935720939438327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089489301783553313909335367710657918176439787630450308337123261628838106381200293783370929386621741197476878994846036283440794935566014224984053607319581627192961605840426710571602412848525229136762645962019061631m1=900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987260127122886704588843739714198427509292876586402662196866469569298721157821730939797429587451216719285687094685269871592718982960049728311805164110730512885269703205336811518121606962660616550346512572520487557870123778929296621182400276148181527666623686900512913886278247685910308672609186049761488328294995502322241433324319326856478162169987041255782240438121380402668583122143072829075559781925933961665015867471324884165433851519940553200317373252045781390117026471308510707700147808334133900206987058537825705115021751175576149102155323 9m2=4874439857574051734266281883756571176042355079369675229932579721088722836983052384544657232142268714142767889120581861970398212429127367428240806276809718025112069143946721592402069107358506519993161000146910672957081386393632035962446939955627802866371163947382507741297590210801973237248054146680423188060106528144050787697385489136754661815510055270653095153649506101372063932571483576596666870916627498485602254538263622717042926928475963395332290880388205320861094211585758410776012687131750978740835362490060189487894132387839228456334940236088652560719628565812298900438969390256136005642833913293314521990628589303745659916341914951379395745395469

解析一下题目

```
import hashlib
import sympy
from Crypto.Util.number import *

flag = 'GWHT{******}'
secret = '******'

0assert(len(flag) == 38)

half = len(flag) / 2

flag1 = flag[:half]
flag2 = flag[half:]

secret_num = getPrime(1024) * bytes_to_long(secret)

p = sympy.nextprime(secret_num)
q = sympy.nextprime(p)

N = p * q

e = 0x10001

F1 = bytes_to_long(flag1)
F2 = bytes_to_long(flag2)

c1 = F1 + F2
c2 = pow(F1, 3) + pow(F2, 3)
assert(c2 < N)

m1 = pow(c1, e, N)
m2 = pow(c2, e, N)

output = open('secret', 'w')
output.write('N=' + str(N) + '\n')
output.write('m1=' + str(m1) + '\n')
output.write('m2=' + str(m2) + '\n')
output.close()
```

1 说明p,q是接近的

2 得到了e

3 说明了flag被加密成立F1F2

4 5 由4,5可以建立一个方程组，解出F1F2来，进而得到flag

6 m1m2是已知的了，可以用gmpy2求出c1c2，进而照应第4,5步

用yafu分解一下

```
D:\>cd yafu

D:\yafu>cd yafu-1.34

D:\yafu\yafu-1.34>yafu-x64.exe
factor(6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414
0922834573094957082589539445610828279428814524313491262061930512829074466232633130599104490893572093943832740301809630847541592548921200288222467710657918176439787630450308337123261628838106381200293783370929386621741197476878994846036283440794935566014224984053607319581627192961605840


fac: factoring 6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271000042661092283457309495708258953944561082827942881452431349126206193051282907446623263313059910449089357209394383274030180963084754159254892120390933536771065791817643978763045030833712326162883810638120029378337092938662174119747687899484603628344079493556601422498405360731958162719296160584067710657918176439787630450308337123261628838106381200293783370929386621741197476878994846036283440794935566014224984053607319581627192961605840

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 = 10.5799 seconds


***factors found***

P327 = 797862863902421984951231350430312260517773269684958456342860983236184129602390919026048496119757187702076499551310794177917920137646835891232053223784521942612170165525277545132155203294999671081969688331633297246202510960803777487377
P327 = 797862863902421984951231350430312260517773269684958456342860983236184129602390919026048496119757187702076499551310794177917920137646835891232053223784521942612170165525277545132155203294999671081969688331633297246202510960803777476997

ans = 1
```


pq

```
P = 7978628639024219849512313504303122605177732696849584563428609832361841296023909190260484961197571877020764995513107941779179201376468358888627061269240884115709971412571595639527258822141811855312091869723514699462695085113128637791232053223784521942612170165525277545132155203294999671081969688331633297246202510960803777487 37
q = 7978628639024219849512313504303122605177732696849584563428609832361841296023909190260484961197571877020764995513107941779179201376468358888627061269240884115709971412571595639527258822141811855312091869723514699462695085113128637791232053223784521942612170165525277545132155203294999671081969688331633297246202510960803777476 99
```

这两张图是从大佬那边找到的详解版

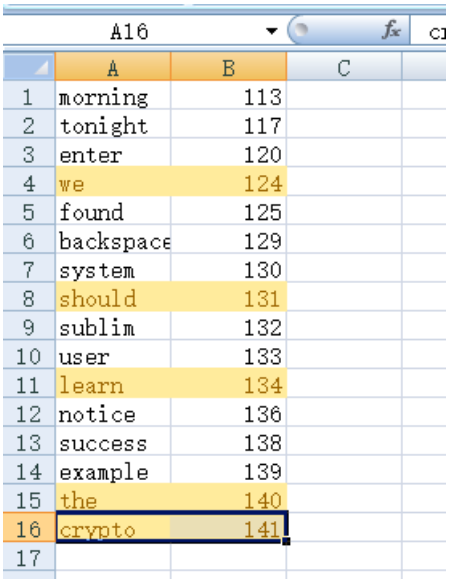
flag{f709e0e2cfe7e530ca8972959a1033b2}

74.浪里淘沙

查看题目

看到题目给的数字可以想到是字频

用word打开搜索统计个数



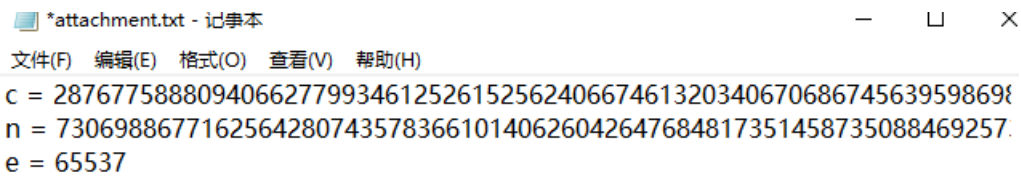
	A	B	C
1	morning	113	
2	tonight	117	
3	enter	120	
4	we	124	
5	found	125	
6	backspace	129	
7	system	130	
8	should	131	
9	sublim	132	
10	user	133	
11	learn	134	
12	notice	136	
13	success	138	
14	example	139	
15	the	140	
16	crypto	141	
17			

再把这个按照题目给的顺序排列以下就是flag

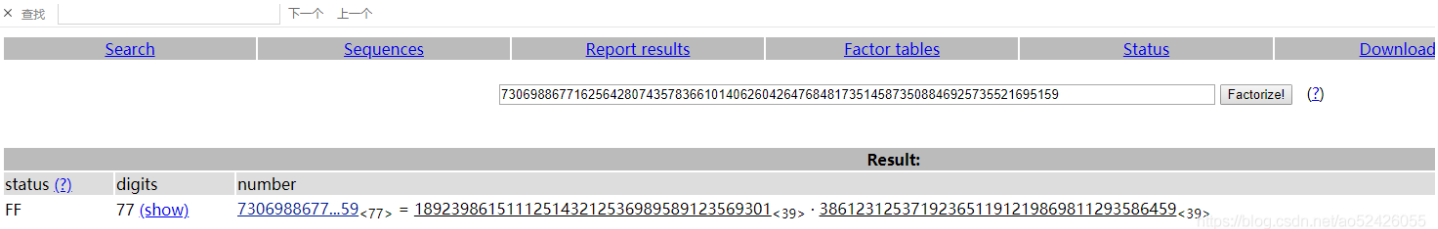
flag{weshouldlearnthecrypto}

75.[WUSTCTF2020]babyrsa

查看题目



看能不能分解N



有了pq就很简单了

```
p=386123125371923651191219869811293586459
q=189239861511125143212536989589123569301
c=28767758880940662779934612526152562406674613203406706867456395986985664083182

n= 73069886771625642807435783661014062604264768481735145873508846925735521695159

import  gmpy2
e=65537
d=gmpy2.invert(e,(p-1)*(q-1))
m=gmpy2.powmod(c,d,n)
import binascii
print(binascii.unhexlify(hex(m)[2:]))
```

运行得到b'wctf2020{just_@*piece_0f_cak3}'

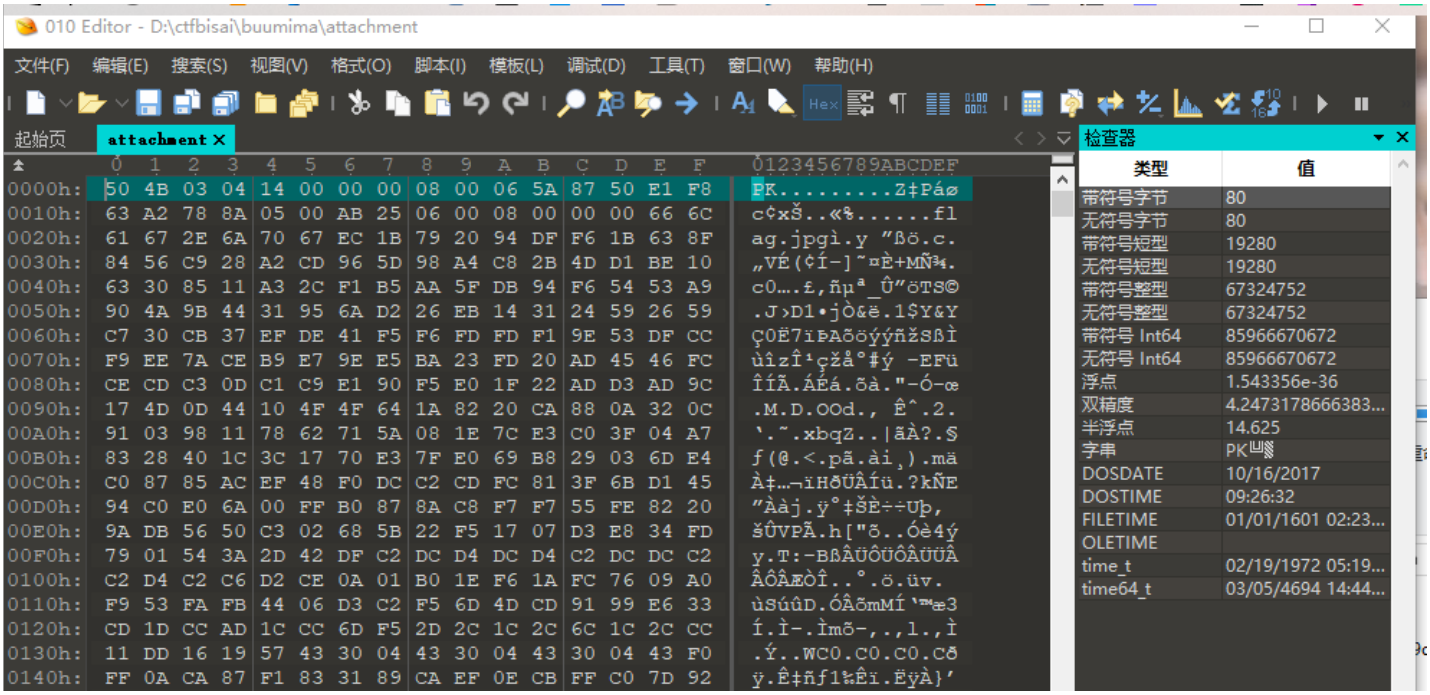
flag{just@_piece_0f_cak3}

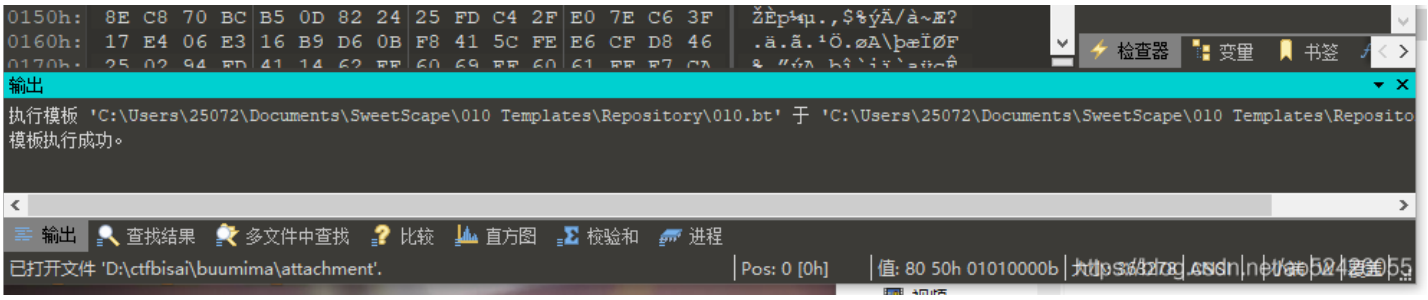
很简单的一道rsa

76.[NPUCTF2020]这是什么觅□

看题目的到一个没有后缀的文件

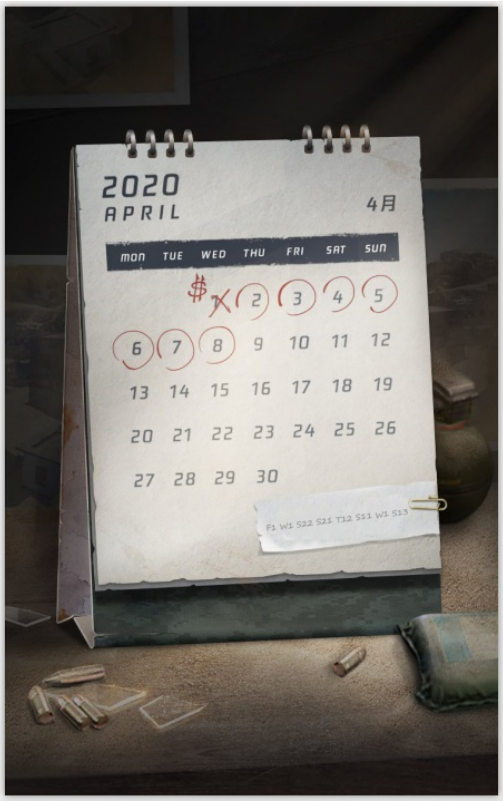
用记事本打开是一堆乱码我们就直接010打开





504B0304 很明显是一个zip文件

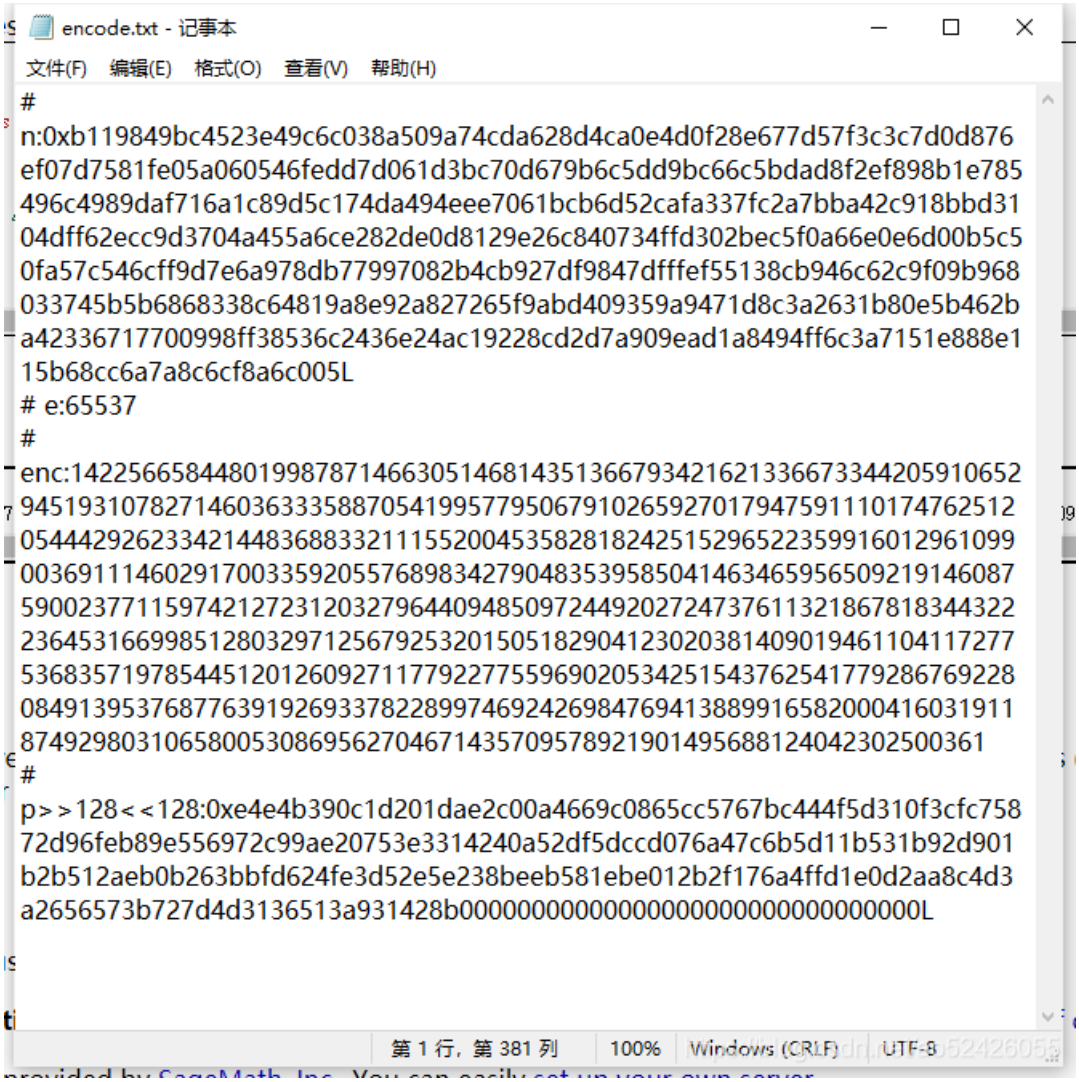
打开得到图片



字母代表星期的首字母，其中 S 和 T 都出现两次，所以 S1 代表 SAT，S2 代表 SUN，每组最后一个数字即代表第几行，F1 W1 S22 S21 T12 S11 W1 S13对应得到 3 1 12 5 14 4 1 18，对照字母表 calendar。

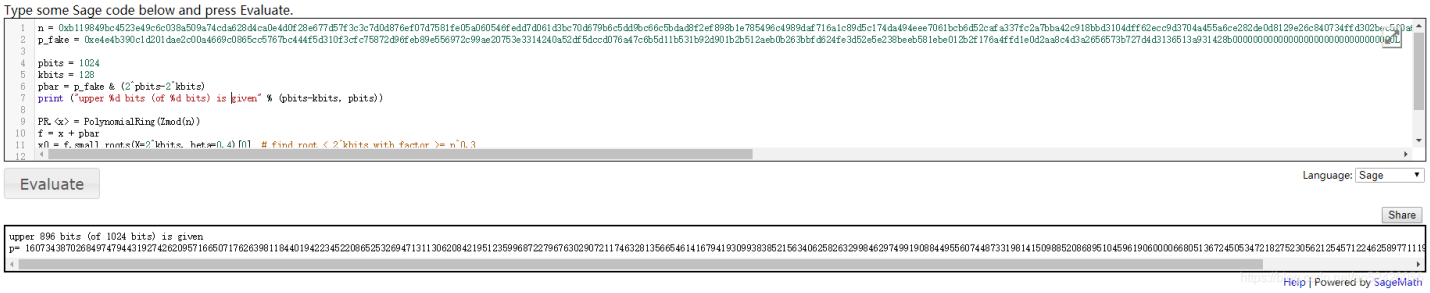
77.[GKCTF2020]babycrypto

查看题目



很明显是真的p高位和n

所有我们用sage来求出p



求出了p 直接求q就可以

```
q = n/p = 13909135305901812842174475152508005653030796591829887569129999277548406442659158145699896831558234902707198720634065398892592346522547166189394439774429339126927412434518902881897700260059973246982416421836639972623337306974283973706200406124478741363829076759002937606250889741710911718961445857024140745835 9
```

写脚本

```
import gmpy2 as gp
import binascii
p = 160734387026849747944319274262095716650717626398118440194223452208652532694713113062084219512359968722796763
029072117463281356654614167941930993838521563406258263299846297499190884495560744873319814150988520868951045961906000066805136724505347218275230562125457122462589771119429631727404626489634314291445667
q = 139091353059018128421744751525080056530307965918298875691299992775484064426591581456998968315582349027071987206340653988925923465225471661893944397744293391269274124345189028818977002600599732469824164218366399726233373069742839737062004061244787413638290767590029376062508897417109117189614458570241407458359
e = 65537
c = 142256658448019987871466305146814351366793421621336673344205910652945193107827146036333588705419957795067910265927017947591110174762512054442926233421448368833211155200453582818242515296522359916012961099003691114602917003359205576898342790483539585041463465956509219146087590023771159742127231203279644094850972449202724737611321867818344322236453166998512803297125679253201505182904123020381409019461104117277536835719785445120126092711779227755969020534251543762541779286769228084913953768776391926933782289974692426984769413889916582000416031911874929803106580053086956270467143570957892190149568812404230250036 1
p = gp.mpz(p)
q = gp.mpz(q)
e = gp.mpz(e)
c = gp.mpz(c)

n = p*q
phi = (p-1) * (q-1)
d = gp.invert(e, phi)
#print('d=',d)
m = pow(c, d, n)
#print('m=',m)
print('hex(m)=',hex(m)[2:])
print('bytes.fromhex(hex(m)=',bytes.fromhex(hex(m)[2:]))
```

运行得到

```
hex(m)= 666c61677b33643039313461312d316539372d343832322d613734352d6337653230633531373962397d
bytes.fromhex(hex(m)= b'flag{3d0914a1-1e97-4822-a745-c7e20c5179b9}'
```

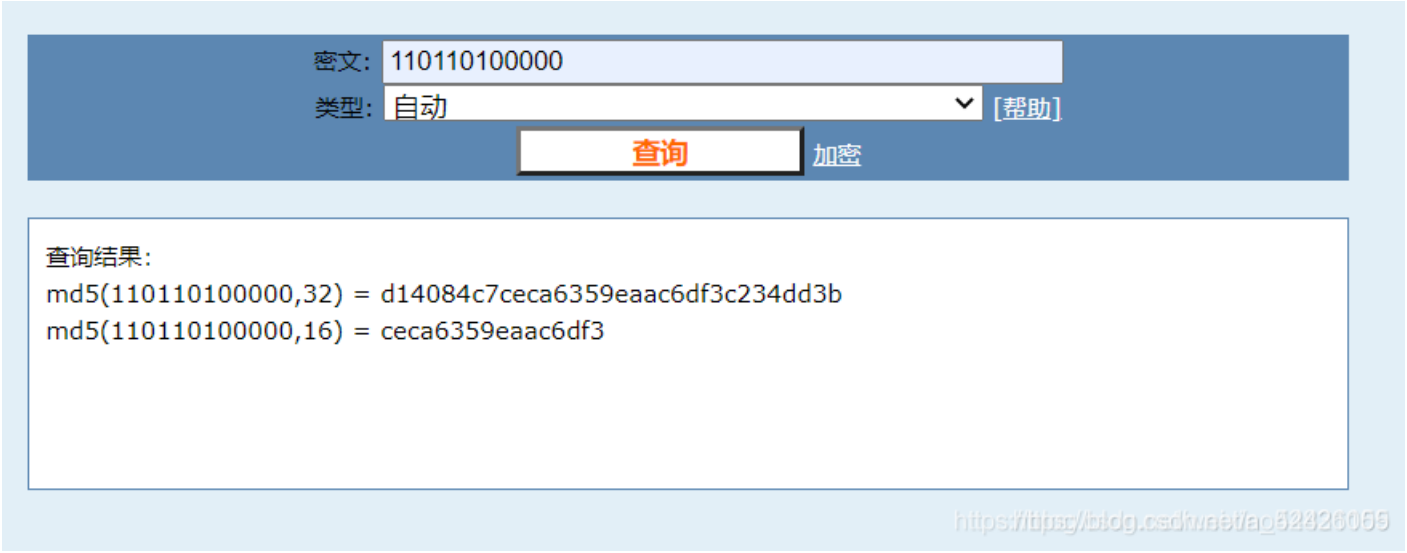
78.鸡藕椒盐味

直接将110010100000 用md5加密出来是不正确
使用海明校验码得到110110100000

海明校验码参考：

https://baike.baidu.com/item/%E6%B5%B7%E6%98%8E%E6%A0%A1%E9%AA%8C%E7%A0%81/7060344?fr=aladdin

https://baijiahao.baidu.com/s?id=1598006039749022275&wfr=spider&for=pc



得到flag：d14084c7ceca6359eaac6df3c234dd3b

79.RSA4

查看题目

```
N = 331310324212000030020214312244232222400142410423413104441140203003243002104333214202031202212403400220031202
142322434104143104244241214204444443323000244130122022422310201104411044030113302323014101331214303223312402430 4
024044130332431321010104222401331222114004340232222142314024034032000122210233413333400423431223021134102101102 2
123324130302443133000130340402010444244312013000033411004243201020340144040401000344200122304221144200141300 4
c = 310020004234033304244200421414413320341301002123030311202340222410301423440312412440240244110200112141140201
224032402232131204213012303204422003300004011434102141321223311243242010014140422411342304322201241112402132203 1
011312212230040220031200021102300233411432014043113403111342301402314122013333331424024231343332113021024131111 1
142443003244012334003404431422340040122411132300024223442044124041102102310022200312321434303012203230104224 3

N = 302240000040421410144422133334143140011011044322223144412002220243001141141114123223331331304421113021231204
322233120121444434210041232214144413244434424302311222143224402302432102242132244032010020113224011121043232143 2
212034242431340443140222120243431000423420024323311443002142124140334141200043442113302240203012230333343242440 3
120424012230124223201130321122004422241113440301213242031111030244234402112210122441123000220334414014304411 4
c = 112200203404013430330214124004404423210041321043000303233141423344144222343401042200334033203124030011440014
210112103234440312134032123400444344144233020130110134042102220302002413321102022414130443041144240310121020100 3
101043342042344124114244203212111122320311213303103334144234333433220244001212003333304322234214333441220230124 4
001304140142320221012402443104001341431312112343342411311341442204333042200231414411113414204433340411224034 4

N = 332200324410041111434222123043121331442103233332422341041340412034230003314420311333101344231212130200312041
044324431141033004333110021013020140020011222012300020041342040004002220210223122111314112124333211132230332124 0
224231412140313031444441344030244201114232444240300300033402130321213032133430204013042433300013140230301210341 1
333440444042124224011310320301334123133000433204030244001132400413032403432343014310240144013024232142402032 3
c = 100134441201411303224332041240022422243323340111242100124402414023421004103311314413032420110021013230404033
111204213044222222003244022442433224224444140433421301111113300222132030303244221011330322120420422431014343422 0
320412104211321210421242333033113431131111414320001124000211131212223434000340331204040104302143311203133432432 2
123304112340014030132021432101130211241134422413442312013042141212003102211300321404043012124332013240431242 
```

看到这一题，给到了三组三个n三个c，我们可以很容易的想到了中国剩余定理，但是我们发现这题并不是那么容易的，可以说是非常恶心人了，他题目给到的c和n并不是用的十进制，而是用的五进制，要先用int("******",5)的代码转换为十进制才能计算。后来我们又发现，将题目通过CRT解出来并不对，看来并不是flag直接模的n，于是我们猜测e=3，果然，成功解出flag，脚本如下

```
// python2
import gmpy2
```

```
from functools import reduce
import sympy

def chinese_remainder(n, a):
    sum = 0
    prod = reduce(lambda a, b: a * b, n)

    for n_i, a_i in zip(n, a):
        p = prod // n_i
        sum += a_i * sympy.invert(p, n_i) * p
    return int(sum % prod)
def wu_shi(n):
    strlen=len(n)
    s=0
    for i in range(0,strlen):
        s+=int(n[i])*pow(5,strlen-i-1)
    return s

n1 = int(str(331310324212000030020214312244232222400142410423413104441140203003243002104333214202031202212403400
220031202142322434104143104244241214204444443323000244130122022422310201104411044030113302323014101331214303223 3
124024304024044130332431321010104222401331222114004340232222142314024034032000122210233413333400423431223021134 1
021011022123324130302443133000130340402010444244312013000033411004243201020340144040401000344200122304221144200 1
413004),5)
c1 = int(str(310020004234033304244200421414413320341301002123030311202340222410301423440312412440240244110200112
141140201224032402232131204213012303204422003300004011434102141321223311243242010014140422411342304322201241112 4
021322031011312212230040220031200021102300233411432014043113403111342301402314122013333331424024231343332113021 0
241311111142443003244012334003404431422340040122411132300024223442044124041102102310022200312321434303012203230 1
042243),5)

n2 = int(str(302240000040421410144422133334143140011011044322223144412002220243001141141114123223331331304421113
021231204322233120121444434210041232214144413244434424302311222143224402302432102242132244032010020113224011210 4
432321432212034242431340443140222120243431000423420024323311443002142124140334141200043442113302240203012230333 3
432424403120424012230124223201130321122004422241113440301213242031111030244234402112210122441123000220334414014 3
044114),5)
c2 = int(str(112200203404013430330214124004404423210041321043000303233141423344144222343401042200334033203124030
011440014210112103234440312134032123400444344144233020130110134042102220302002413321102022414130443041144240310 1
210201003101043342042344124114244203212111122320311213303103334144234333433220244001212003333304322234214333441 2
202301244001304140142320221012402443104001341431312112343342411311341442204333042200231414411113414204433340411 2
240344),5)

n3 = int(str(332200324410041111434222123043121331442103233332422341041340412034230003314420311333101344231212130
200312041044324431141033004333110021013020140020011222012300020041342040004002220210223122111314112124333211322 3
303321240224231412140313031444441344030244201114232444240300300033402130321213032133430204013042433300013140230 3
012103411333440444042124224011310320301334123133000433204030244001132400413032403432343014310240144013024232142 4
020323),5)
c3 = int(str(100134441201411303224332041240022422243323340111242100124402414023421004103311314413032420110021013
230404033111204213044222220032440224424332242244441404334213011111133002221320303032442210113303221204204224310 1
143434220320412104211321210421242333033113431131111414320001124000211131212223434000340331204040104302143311203 1
334324322123304112340014030132021432101130211241134422413442312013042141212003102211300321404043012124332013240 4
31242),5)



n=[n1,n2,n3]
c=[c1,c2,c3]
ans=chinese_remainder(n, c)
ans=gmpy2.iroot(ans,3)[0]
print hex(ans)[2:].decode('hex')
#noxCTF{D4mn_y0u_h4s74d_wh47_4_b100dy_b4s74rd!}
```

80.[NCTF2019]babyRSA

查看题目

```
from Crypto.Util.number import *
from flag import flag

def nextPrime(n):
    n += 2 if n & 1 else 1
    while not isPrime(n):
        n += 2
    return n

p = getPrime(1024)
q = nextPrime(p)
n = p * q
e = 0x10001
d = inverse(e, (p-1) * (q-1))
c = pow(bytes_to_long(flag.encode()), e, n)

# d = 19275778946037899718035455438175509175723911466127462154506916564101519923603308900331427601983476886255849200332374081996442976307058597390881168155862238533018621944733299208108185814179466844504468163200369996564265921022888670062554504758512453217434777820468049494313818291727050400752551716550403647148197148884408264686846693842118387217753516963449753809860354047619256787869400297858568139700396567519469825398575103885487624463424429913017729585620877168171603444111464692841379661112075123399343270610272287865200880398193573260848268633461983435015031227070217852728240847398084414687146397303110709214913
# c = 53827231680738281106961685582942066817579911490227778211275633014134832238745272333007211808392986170767056850411742474158261570965830550693373939878922627642112252270358807544174570567239091355252449579359069026656797771011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329232598986279971458038927539892556152731403000210406545059014427878106536265243057063166631693417972057529387555900565689867382278034874672741143982571879621407965511362205328096876068673856393677437055275116807199553807463776311564686898441508783814605609907556528994493400453135218 04
```

这道题没有给出n，只给出了e和d，于是我们可以尝试通过爆破的方法求得欧拉函数进而求得，p和q。然后我们可以看到p和q的生成函数，p和q非常的接近，而且都在2**1023和2*1024之间，于是可以发现把欧拉函数phi开平方根之后的下一个素数，就是生成的q，进而可以解出本题，下面给出脚本

```
// python2
from gmpy2 import *
from sympy import *
from Crypto.Util.number import *
d = 19275778946037899718035455438175509175723911466127462154506916564101519923603308900331427601983476886255849200332374081996442976307058597390881168155862238533018621944733299208108185814179466844504468163200369996564265921022888670062554504758512453217434777820468049494313818291727050400752551716550403647148197148884408264686846693842118387217753516963449753809860354047619256787869400297858568139700396567519469825398575103885487624463424429913017729585620877168171603444111464692841379661112075123399343270610272287865200880398193573260848268633461983435015031227070217852728240847398084414687146397303110709214913
c = 53827231680738281106961685582942066817579911490227778211275633014134832238745272333007211808392986170767056850411742474158261570965830550693373939878922627642112252270358807544174570567239091355252449579359069026656797771011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329232598986279971458038927539892556152731403000210406545059014427878106536265243057063166631693417972057529387555900565689867382278034874672741143982571879621407965511362205328096876068673856393677437055275116807199553807463776311564686898441508783814605609907556528994493400453135218041
e=0x10001
src=d*e-1
i=2**15
while True:
 if(src%i==0):
  if((src//i)>=2**2046 and (src//i)<=2**2048):
   phi=src//i
   q_1=iroot(phi,2)[0]
   q=nextprime(q_1)
   if(phi%(q-1)==0):
    p_1=phi//(q-1)
    p=p_1+1
    if(isprime(p)):
     print(p,q)
     #(14319361159175221091877047640238478335174002884176322323610288522183996663707318846280819597454857983336831390408309578690647941634768192373110026035965242644159310775589248594480941918934831195630845645952343745996971306065343290987398659604248269967045171629674372752558643724846243232742336108081122507583 9L, 14319361159175221091877047640238478335174002884176322323610288522183996663707318846280819597454857983336831390408309578690647941634768192373110026035965242644159310775589248594480941918934831195630845645952343745996971306065343290987398659604248269967045171629674372752558643724846243232742336108081122507649 7L)
     break
 i+=1
 if((src//i)<2**2046):
  print 0
  break
n=p*q
m=pow(c,d,n)
flag=long_to_bytes(m)
print flag
#NCTF{70u2_nn47h_14_v3ry_gOO0000000d}
```

81.[BJDCTF2020]easyrsa

查看题目

```
from Crypto.Util.number import getPrime,bytes_to_long
from sympy import Derivative
from fractions import Fraction
from secret import flag

p=getPrime(1024)
q=getPrime(1024)
e=65537
n=p*q
z=Fraction(1,Derivative(arctan(p),p))-Fraction(1,Derivative(arth(q),q))
m=bytes_to_long(flag)
c=pow(m,e,n)
print(c,z,n)
'''
output:
792254786685776145980749150265421628301277617778951154935067295810181028134840228409831014779654943068925380351099487742013553726854941065265447962085869132411036718202564878840704159994309138622754318215774620294709957238967608439270640608430765700010466569665440915500631320395729288574379171519878197420557865479212319158495766529320839045374836918233315280988231245335970614780819892291676277372172668158897710387745411904374488916452938318807749919493290964391869664876907327364751380953182517883134591810800848971719184808713694342985458103006676013451912221080252735948993692674899399826084848622145815461035
321157486776232096674716228721852750702579247660150200728052673598390593932843165958829333722897321272740764345875193333001424730103446948038851685575488012024959332262154377633292802421135565244984575595628729008116020569444239674037776233069618807576132463287296166430326289640729312720858669280459737993747118468251577810569651641785052325242458091792356075715671742288225616978886459685593436083753319880971571452643576267381416465563535009949241158757481983180362968986040970009382721959030567335658801505402753692396377939759233295987160033503082593214367525792910003555604315422296997599551411529147083624944821531074516133689541340669000932476620078917924889695194204723544890161235112845930914582554756929847982110124909416186720768653760704744796870875899095013638092474735905257054959409856997063285435182595072975256350228484926373012758638252270395989339232933376092763735305225027419582146902340144384139509641023184359210142659188257340593418867512432699727777523828792840374332429770515173252464121351630658529772219078008818070507035946971986934393910652920479828595751686077438400189277752591616774327241995857205533223205609597944815508246597778148259837199479887191751476750839473044797477032996768176762549539444 1

'''
```

从题目上我们可以看出cnz
通过对z的化简可以得到

```
p2+q2=z
pxq=n

因此可以得到

(p+q)2=z+2n
(p-q)2=z-2n
```

直接上脚本就可以了

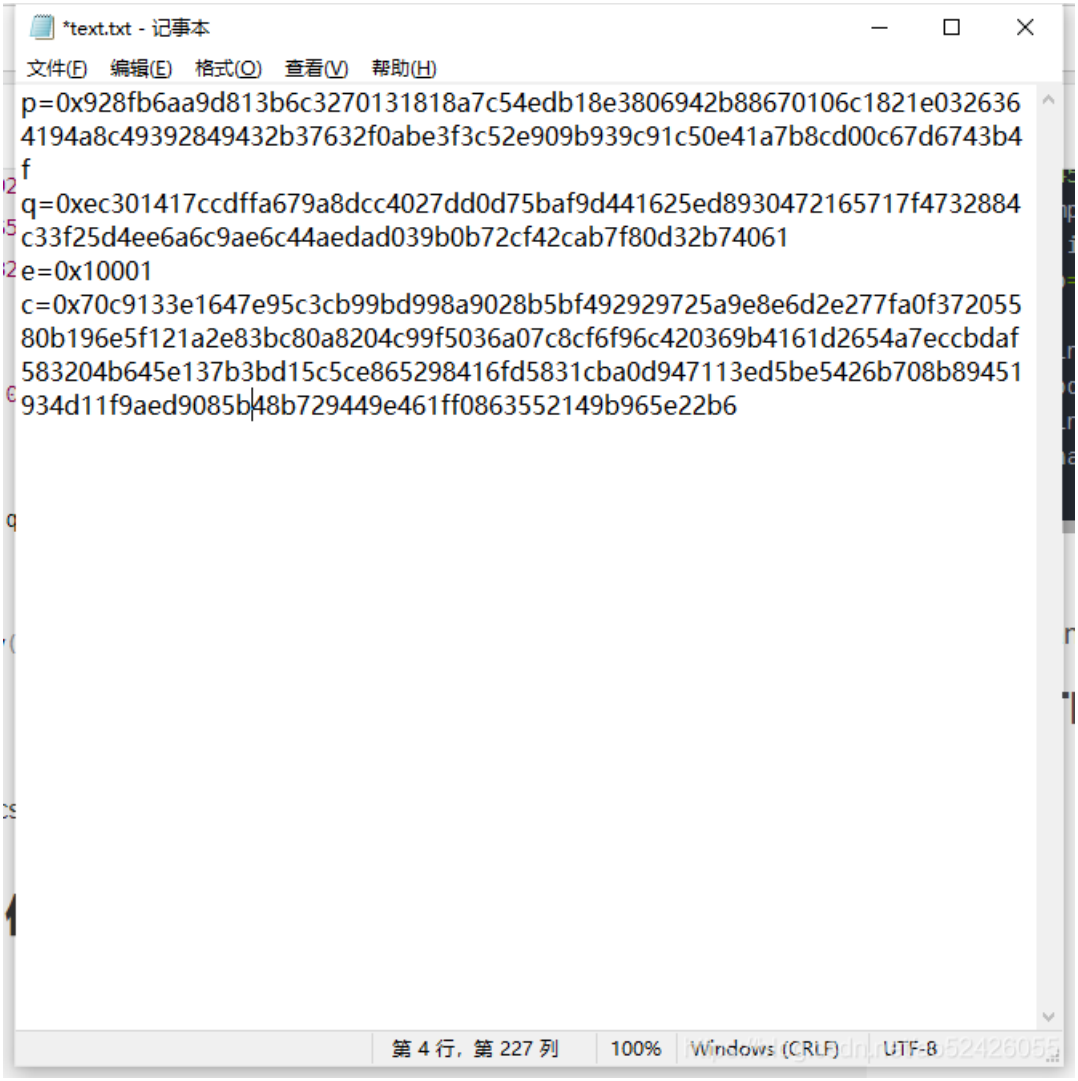
```
e=65537
c=79225478668577614598074915026542162830127761777895115493506729581018102813484022840983101477965494306892538035
10994877420135537268549410652654479620858691324110367182025648788407041599943091386227543182157746202947099572387
96760843927064060843076570001046656966544091550063132039572928857437917151987819742055786547921231915849576652932
08390453748369182333152809882312453359706147808198922916762773721726681588977103877454119043744889164529383188077
49991949329096439186966468769073273647513809531825178831345918108008489717191848087136943429854581030066760134519
12221080252735948993692674899399826084848622145815461035
z=32115748677623209667471622872185275070257924766015020072805267359839059393284316595882933372289732127274076434587
51933330014247301034469480388516855754880120249593322621543776332928024211355652449845755956287290081160205694442
39674037776233069618807576132463287296166430326289640729312720858669280459737993747118468251577810569651641785052
32524245809179235607571567174228822561697888645968559343608375331988097157145264357626738141646556353500994924115
87574819831803629689860409700093827219590305673356588015054027536923963779397592332959871600335030825932143675257
92910003555604315422296997599551411529147083624944821532106745161336895413406690009324766200789179248896951942047
23544890161235112845930914582554756929847982110124909416186720768653760704744796870875899095013638092474735905257
05495940985699706328543518259507297525635022848492637301275863825227039598933923293337609276373530522502741958214
69023401443841395096410231843592101426591882573405934188675124326997277775238287928403743324297705151732524641213
51630658529772219078008818070507035946971986934393910652920479828595751686077438400189277752591616774327241995857
20553322320560959794481550824659777814825983719947988719175147675083947304479747703299676817676254953944415321067
45161336895413406690009324766200789179248896951942047235448901612351128459309145825547569298479821101249094161867
20768653760704744796870875899095013638092474735905257054959409856997063285435182595072975256350228484926373012758
63825227039598933923293337609276373530522502741958214690234014438413950964102318435921014265918825734059341886751
24326997277775238287928403743324297705151732524641213516306585297722190780088180705070359469719869343939106529204
79828595751686077438400189277752591616774327241995857205533223205609597944815508246597778148259837199479887191751
47675083947304479747703299676817676254953944415321067451613368954134066900093247662007891792488969519420472354489
01612351128459309145825547569298479821101249094161867207686537607047447968708758990950136380924747359052570549594
09856997063285435182595072975256350228484926373012758638252270395989339232933376092763735305225027419582146902340
14438413950964102318435921014265918825734059341886751243269972777752382879284037433242977051517325246412135163065
85297722190780088180705070359469719869343939106529204798285957516860774384001892777525916167743272419958572055332
23205609597944815508246597778148259837199479887191751476750839473044797477032996768176762549539444153210674516133
68954134066900093247662007891792488969519420472354489016123511284593091458255475692984798211012490941618672076865
37607047447968708758990950136380924747359052570549594098569970632854351825950729752563502284849263730127586382522
70395989339232933376092763735305225027419582146902340144384139509641023184359210142659188257340593418867512432699
72777752382879284037433242977051517325246412135163065852977221907800881807050703594697198693439391065292047982859
57516860774384001892777525916167743272419958572055332232056095979448155082465977781482598371994798871917514767508
39473044797477032996768176762549539444153210674516133689541340669000932476620078917924889695194204723544890161235
11284593091458255475692984798211012490941618672076865376070447447968708758990950136380924747359052570549594098569
97063285435182595072975256350228484926373012758638252270395989339232933376092763735305225027419582146902340144384
13950964102318435921014265918825734059341886751243269972777752382879284037433242977051517325246412135163065852977
22190780088180705070359469719869343939106529204798285957516860774384001892777525916167743272419958572055332232056
09597944815508246597778148259837199479887191751476750839473044797477032996768176762549539444153210674516133689541
34066900093247662007891792488969519420472354489016123511284593091458255475692984798211012490941618672076865376070
44744796870875899095013638092474735905257054959409856997063285435182595072975256350228484926373012758638252270395
98933923293337609276373530522502741958214690234014438413950964102318435921014265918825734059341886751243269972777
75238287928403743324297705151732524641213516306585297722190780088180705070359469719869343939106529204798285957516
86077438400189277752591616774327241995857205533223205609597944815508246597778148259837199479887191751476750839473
04479747703299676817676254953944415321067451613368954134066900093247662007891792488969519420472354489016123511284
59309145825547569298479821101249094161867207686537607044744796870875899095013638092474735905257054959409856997063
28543518259507297525635022848492637301275863825227039598933923293337609276373530522502741958214690234014438413950
96410231843592101426591882573405934188675124326997277775238287928403743324297705151732524641213516306585297722190
78008818070507035946971986934393910652920479828595751686077438400189277752591616774327241995857205533223205609597
94481550824659777814825983719947988719175147675083947304479747703299676817676254953944415321067451613368954134066
90009324766200789179248896951942047235448901612351128459309145825547569298479821101249094161867207686537607044744
79687087589909501363809247473590525705495940985699706328543518259507297525635022848492637301275863825227039598933
92329333760927637353052250274195821469023401443841395096410231843592101426591882573405934188675124326997277775238
28792840374332429770515173252464121351630658529772219078008818070507035946971986934393910652920479828595751686077
43840018927775259161677432724199585720553322320560959794481550824659777814825983719947988719175147675083947304479
74770329967681767625495394441
import gmpy2
p=(gmpy2.iroot(z+2*n,2)[0]+gmpy2.iroot(z-2*n,2)[0])//2
print(n%p==0)
q=n//p
d=gmpy2.invert(e,(p-1)*(q-1))
m=gmpy2.powmod(c,d,n)
import binascii
print(binascii.unhexlify(hex(m)[2:]))
```

运行得到

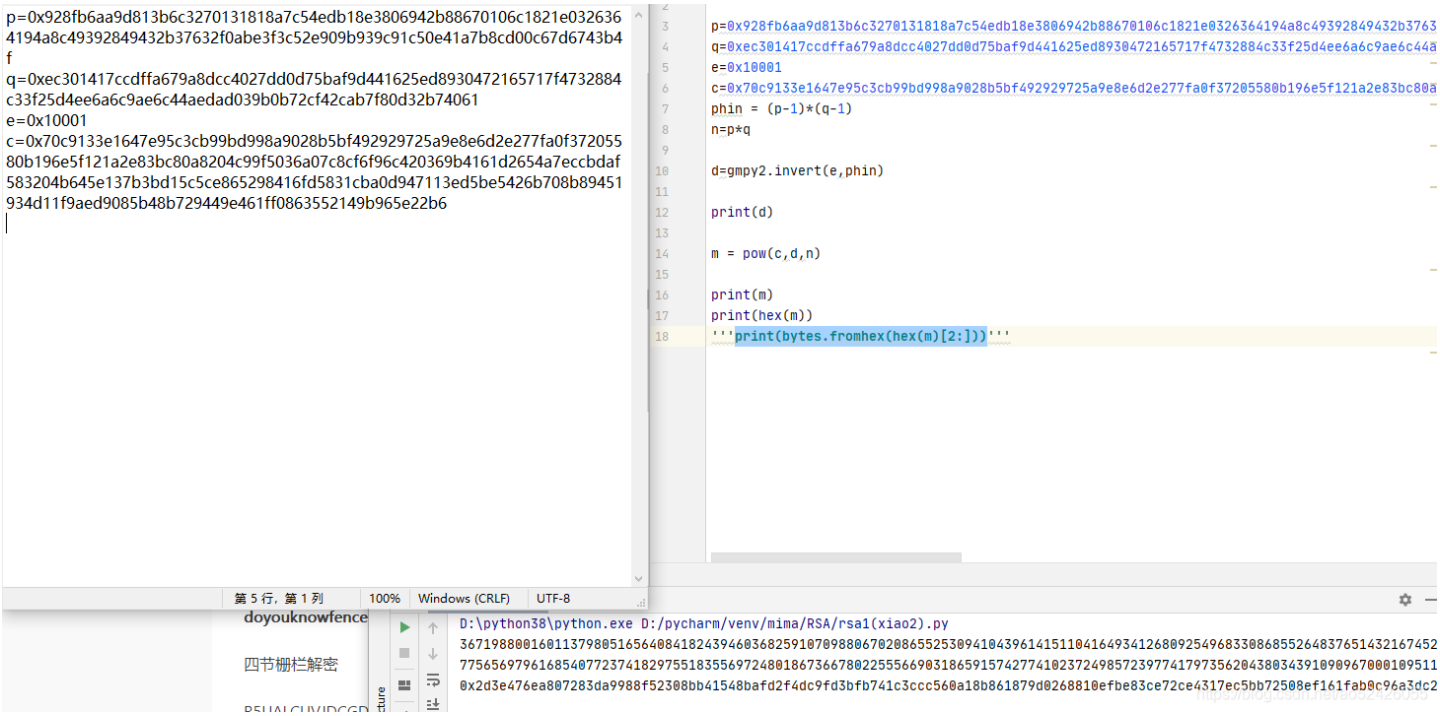
BJD{Advanced_mathematics_is_too_hard!!!}

82.[AFCTF2018]你能看出这是什么加密么

查看题目



我一开始看见这题 不知道是陷阱还是怎么的



我当时一直用的这个发现解不出来

上真正的代码

```
from gmpy2 import*
from Crypto.Util.number import long_to_bytes

p=int('0x928fb6aa9d813b6c3270131818a7c54edb18e3806942b88670106c1821e0326364194a8c49392849432b37632f0abe3f3c52e909b939c91c50e41a7b8cd00c67d6743b4f',16)

q=int('0xec301417ccdffa679a8dcc4027dd0d75baf9d441625ed8930472165717f4732884c33f25d4ee6a6c9ae6c44aedad039b0b72cf42cab7f80d32b74061',16)

e=int('0x10001',16)

c=int('0x70c9133e1647e95c3cb99bd998a9028b5bf492929725a9e8e6d2e277fa0f37205580b196e5f121a2e83bc80a8204c99f5036a07c8cf6f96c420369b4161d2654a7eccbdaf583204b645e137b3bd15c5ce865298416fd5831cba0d947113ed5be5426b708b89451934d11f9aed9085b48b729449e461ff0863552149b965e22b6',16)

phi = (q-1)*(p-1)
n=p*q

d=invert(e,phi)
m=pow(c,d,n)
print(long_to_bytes(m))
```

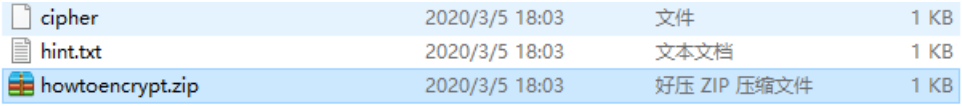
运行得到

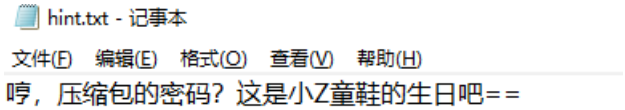
```
b'\x02\xd3\xe4v\xea\x80r\x83\xda\x99\x88\xf5#\x08\xbbAT\x8b\xaf\xd2\xf4\xdc\x9f\xd3\xbf\xb7A\xc3\xcc\xc5`\xa1\x8b\x86\x18y\xd0&\x88\x10\xef\xbe\x83\xcer\xceC\x17\xec[\xb7%\x08\xef\x16\x1f\xab\x0c\x96\xa3\xdc N^\x8e,\xa3\x11{\x99U\xcd\x15o\xd7B\xf4L\x8f}&\xc5$\xca\xd5;\xf9\x02Y\xc1\xbbS\xfd4\x83M\x96\xa9\xbd;\x83/\xf7\x00afctf{R54_|5_$0_$imp13}'
```

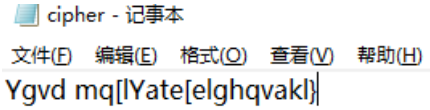
这个就很明显了afctf{R54_|5_KaTeX parse error: Expected group after '_' at position 2: 0_imp13}换成flag提交就行了

83.[ACTF新生赛2020]crypto-classic0

查看题目

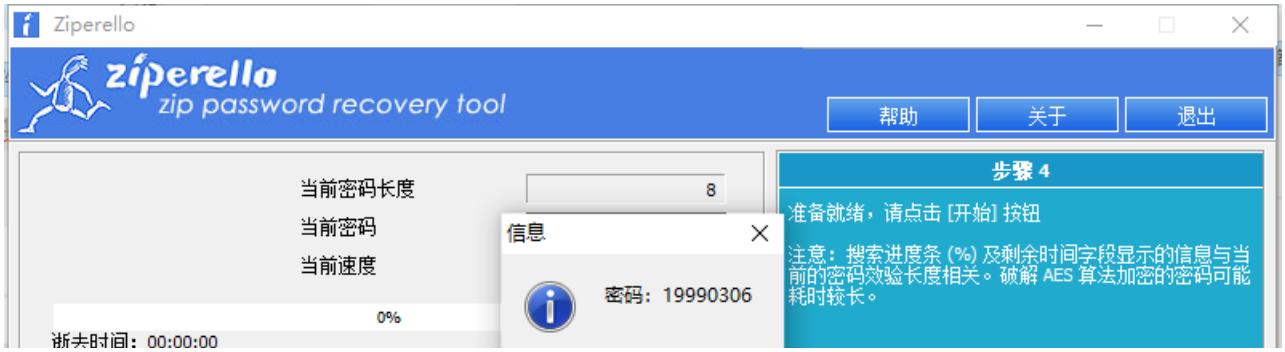


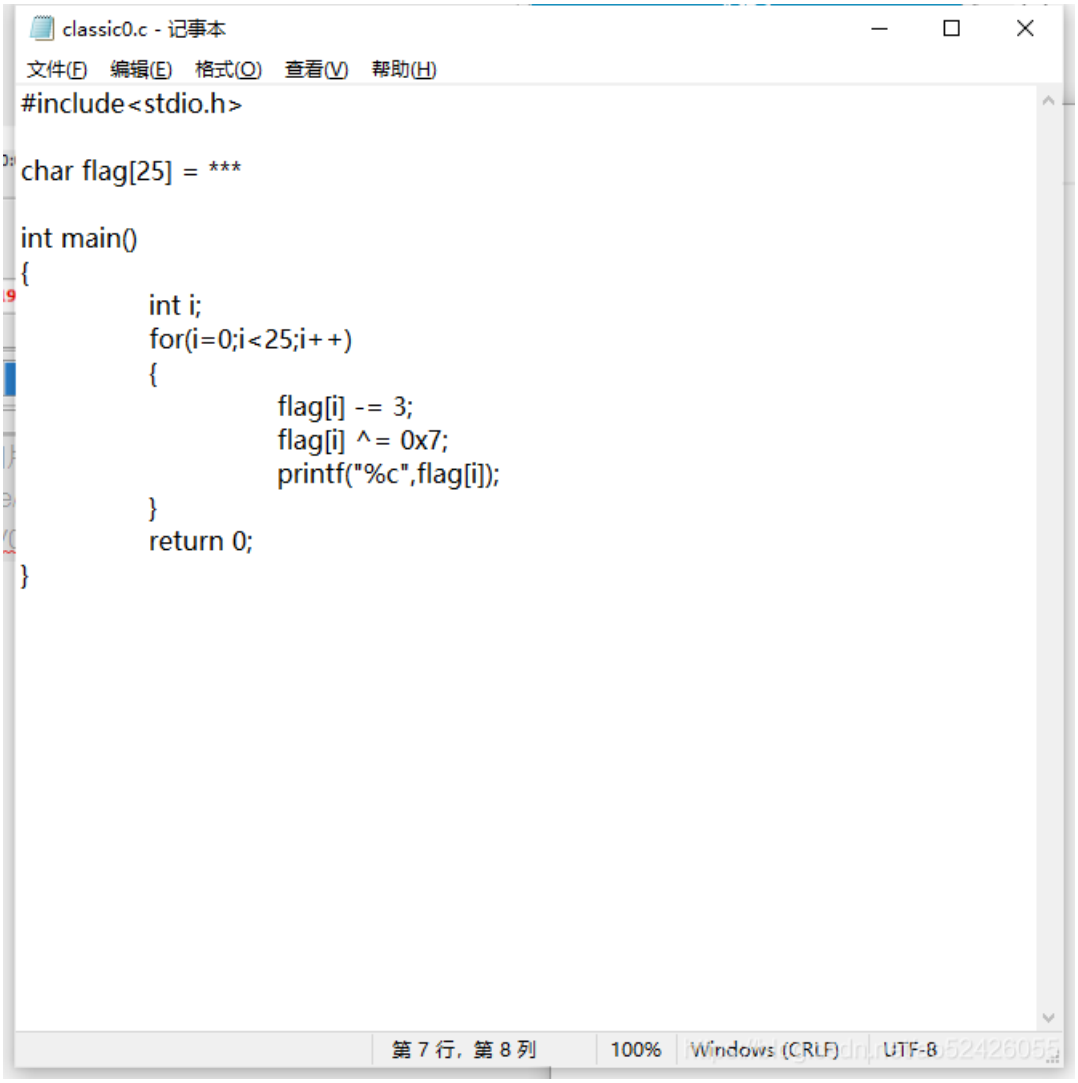




既然压缩包密码是生日那么就直接暴力破解吧

说一下思路 生日八位数 我选择从19000000开始的





很明显这是一个循环，我们反过来写一个脚本就可以了

```
cipher = 'Ygvdmq[lYate[elghqvakl}'

for i in range(0, 24):
    flag = ord(cipher[i]) ^ 0x7
    flag += 3
    print(chr(flag), end='')
```

运行得到

```
Traceback (most recent call last):
  File "D:/pycharm/venv/mima/xunhuan.py", line 4, in <module>
    flag = ord(cipher[i]) ^ 0x7
IndexError: string index out of range
actfmy_naive_encrytion}
进程完成，退出码 1
```

可能是代码哪写错了我也不知道
最后flag{my_naive_encrytion}

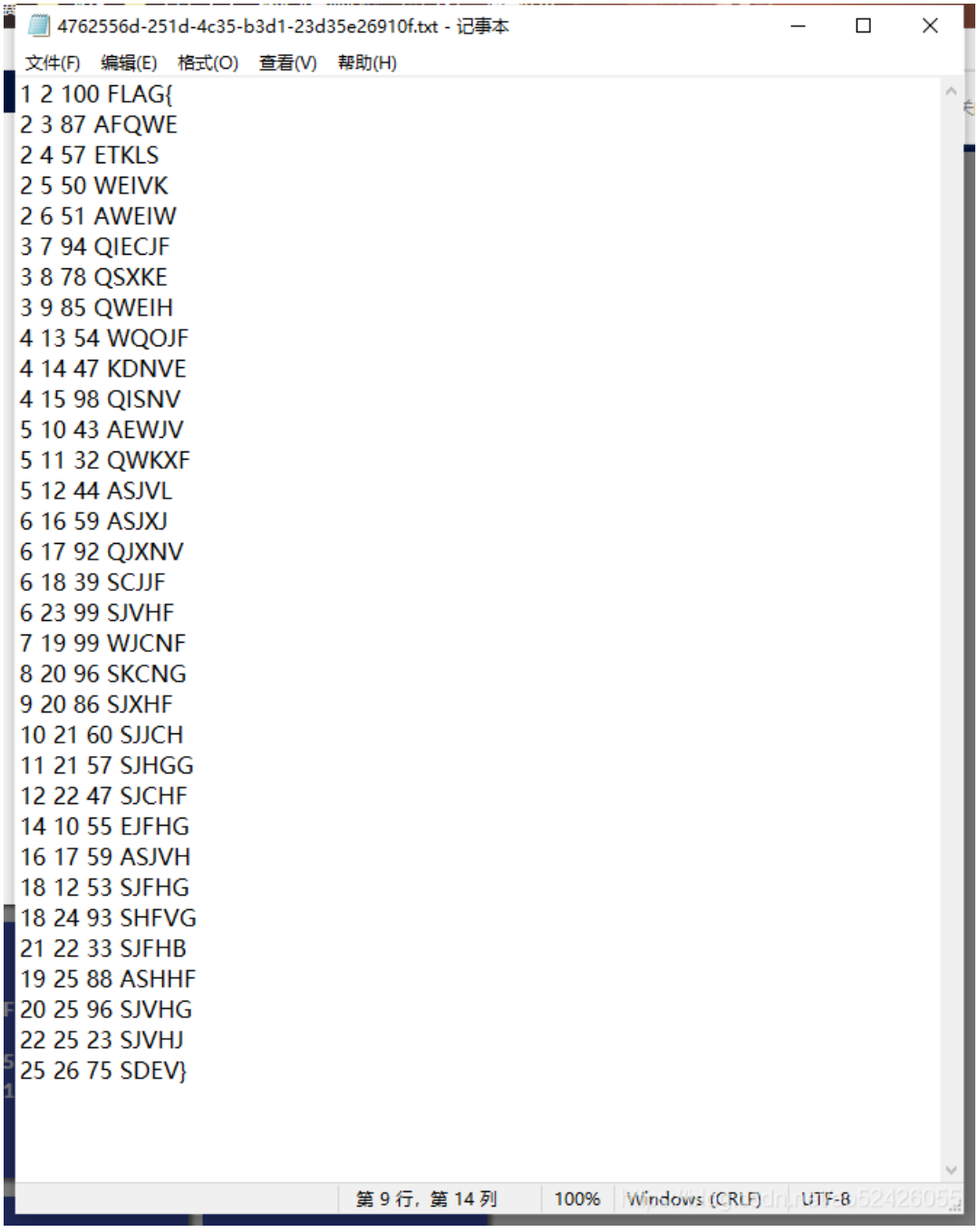
84.救世捷径

查看题目

救世捷径

1

一个名叫CPU的神秘大陆有26个国家，有些国家之间会有一条无向路，每条路径都有不同的长度和一段神秘代码，救世主尼奥要从国家1出发，赶往国家26拯救大陆，请你帮助救世主选择最短路径，而走过的路的神秘代码连接起来便是flag。 以下是数行数据，每行第一个，第二个数字代表这条路的两个端点国家，第三个数字代表路途长度，最后一个字符串便是神秘代码。路在附件中~ 帮助救世主尼奥吧，他快被吓尿了。。。
注意：得到的 flag 请包上 flag{} 提交



这个其实可以不用写脚本，直接把路径画出来找最短就可以了而且也很好画（我最开始也是画出来的）
给你们一个大佬写的脚本

```
graph=[]
for i in range(27):
    graph.append([])
for i in range(27):
    for j in range(27):
        graph[i].append(0x3f3f3f)
f=open('zdlj.txt','r').readlines()#这里需要手动将原文中的最后一行换行给去掉
li=[]
for x in f:
    li.append(x.strip().split(' '))
#print(li)
#print(graph)
for x in li:
    graph[int(x[0])][int(x[1])]=int(x[2])
    graph[int(x[1])][int(x[0])]=int(x[2])
#print(graph)
def dijkstra():
    dv=[0x3f3f3f for i in range(27)]#点i到起点1的最短距离
    route=[1 for i in range(27)]#记录每点和与它对应的上一点
    vis=[0 for i in range(27)]#各点到起点的最短距离是否已定.
    for i in range(2,27):
        dv[i]=graph[i][1]
    dv[1]=0
    #print(dv)
    vis[1]=1
    for i in range(26):
        minn=0x3f3f3f
        temp=-1
        for j in range(2,27):
            if vis[j]==0 and minn>dv[j]:
                minn=dv[j]
                temp=j
        vis[temp]=1
        #print(temp)
        for j in range(2,27):
            if dv[j]>dv[temp]+graph[temp][j]:
                dv[j]=dv[temp]+graph[temp][j]
                route[j]=temp
    return (route,dv)
route,dv=dijkstra()
print(dv[26])
print(route)
y=26
while y!=1:
    print(y)#这里输出路径
    y=route[y]
```

运行得到

```
339
[1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 5, 5, 5, 4, 4, 4, 6, 6, 6, 25, 9, 11, 12, 6, 18, 22, 25]
26
25
22
12
5
2
```

这把得到的连接起来就得到flag了
flag{WEIVKASJVLSJCHFSJVHJSDEV}