​

这是我写的第一篇博客，记录一下在BUUCTF中的刷题记录。

刷题网址：<https://buuoj.cn/challenges>

**1. 摩丝**

这道题非常简单，对照下面的摩丝表就能很快得到答案。需要特别注意的是，摩丝编码里面只有大写字母，没有小写字母。

​编辑

（图片来源：[[摩斯密码表]摩斯密码对照表-CSDN博客](https://blog.csdn.net/acmdream/article/details/78329497?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522172161266716800222835263%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=172161266716800222835263&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-78329497-null-null.142%5Ev100%5Epc_search_result_base2&utm_term=%E6%91%A9%E4%B8%9D%E5%AF%86%E7%A0%81%E8%A7%84%E5%BE%8B&spm=1018.2226.3001.4187)）

**2. MD5**

使用解密MD5的网站可以轻松解决。

解密MD5的相关网站：<https://www.cmd5.com/>

一个包含了挺多MD5解密的工具集：[CTF资源库|CTF工具下载|CTF工具包|CTF工具集合](http://www.ctftools.com/)

**3. [BJDCTF 2nd]cat\_flag**

打开附件是一个动图，动图由两种小图片组成，每一行有八张图片，很明显是用两种不同的图片来代替0、1。把两种情况都考虑一遍，再用ASCII码表进行解码，即可得到结果。

这里记录一下ASCII码表，以后就不用每次都去搜索了。

​编辑

（图片来源：[字母数字-ASCII码表（最详细，最直观）-CSDN博客](https://blog.csdn.net/qq_53030983/article/details/121687624?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522172155958516800225532531%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=172155958516800225532531&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-121687624-null-null.142%5Ev100%5Epc_search_result_base2&utm_term=ascll%E7%A0%81%E8%A1%A8&spm=1018.2226.3001.4187)）

**4. [GKCTF2020]小学生的密码学**

题目如下：

e(x)=11x+6(mod26)

密文：welcylk

（flag为base64形式）

这是一个简单的仿射密码。

假如有

c=a*m+b(mod n)

则有

m=(c-b)*m^{-1}(mod n)

其中m^{-1}表示a在模n下的逆。

在sagemath中可编写如下代码：

a='welcylk'

d=pow(11,-1,26)

s=''

for i in a:

aa=ord(i)-97

ab=(aa-6)\*d

ac=ab%26

ad=int(ac)+97

print(hex(ad))

s=s+chr(ad)

print(s)



再把结果进行base64编码，然后包上flag{}即可得到答案。

**5. [NewStarCTF 2023 公开赛道]babyrsa**

这个rsa中的n是由多个小质数组成的，使用素因数分解的网站可以轻松进行分解。

网站：<http://factordb.com/>

分解后，需要计算phi。这里的phi需要将所有的素数减一后相乘。

**6. [Dest0g3 520迎新赛]babyRSA**

这道题中p和q存在next\_prime的关系，所以两个数的差较小，可以遍历这个差值找到答案。我们先对n开平方根，此时得到的结果x应该在p、q之间，且与p、q的差值不大。不断遍历x、x+1、x+2···，判断他们是否可以整除n，即可找到p、q的值。

sagemath中的脚本如下：

n=27272410937497615429184017335437367466288981498585803398561456300019447702001403165885200936510173980380489828828523983388730026101865884520679872671569532101708469344562155718974222196684544003071765625134489632331414011555536130289106822732544904502428727133498239161324625698270381715640332111381465813621908465311076678337695819124178638737015840941223342176563458181918865641701282965455705790456658431641632470787689389714643528968037519265144919465402561959014798324908010947632834281698638848683632113623788303921939908168450492197671761167009855312820364427648296494571794298105543758141065915257674305081267

c=14181751948841206148995320731138166924841307246014981115736748934451763670304308496261846056687977917728671991049712129745906089287169170294259856601300717330153987080212591008738712344004443623518040786009771108879196701679833782022875324499201475522241396314392429412747392203809125245393462952461525539673218721341853515099201642769577031724762640317081252046606564108211626446676911167979492329012381654087618979631924439276786566078856385835786995011067720124277812004808431347148593882791476391944410064371926611180496847010107167486521927340045188960373155894717498700488982910217850877130989318706580155251854

for i in range(1000000000):

a=n.nth\_root(2, truncate\_mode=True)

a=int(a[0])+i

if n%a==0:

print(a)

break



**7. [NewStarCTF 公开赛赛道]caeser**

这道题的密文为

synt{uvfgbevpny\_pvcure\_vf\_ihyarenoyr}

根据经验判断，synt的明文应该是flag，计算他们每个字母的ASCII码之间的差，发现相等。对其他字母也减去相同的差值，即可得到结果。

**8. [NewStarCTF 公开赛赛道]吉奥万·巴蒂斯塔·贝拉索先生的密码**

题目为

pqcq{gteygpttmj\_kc\_zuokwv\_kqb\_gtofmssi\_mnrrjt}

Hint: key length is 3

上面一行是密文，下面一行是提示。

这是一个维吉尼亚密码，key的长度为3表示上面的字母每3个为一组，每组的解密方式都是相同的。

猜测前面四个字母为flag，即可知道分组中每一个字母需要减去的数值，编写C语言代码可以得到结果：

#include <stdio.h>

#include <string.h>

int main(){

char \*s="pqcqgteygpttmjkczuokwvkqbgtofmssimnrrjt";

for(int j=0;j<strlen(s);j++){

int m=(s[j]-97)%26+97;

while(m<97){

m+=26;

}

if(j%3==0){

m=m-10;

while(m<97){

m+=26;

}

printf("%c",m);

}else if(j%3==1){

m=m-5;

while(m<97){

m+=26;

}

printf("%c",m);

}else{

m=m-2;

while(m<97){

m+=26;

}

printf("%c",m);

}

}

return 0;

}



**9. [NewStarCTF 2023 公开赛道]不止一个pi**

这道题目中p、q均已知，主要的问题在于计算phi的值。

这里的n是p的三次方乘以q的二次方，而欧拉函数中phi的定义是小于n且与n互质的数的个数。

首先考虑phi（p\*q），我们知道是（p-1）\*（q-1）。因此在1到p\*q中有（p-1）\*（q-1）个数与p\*q互质。由于n中只有p、q这两个质数，所以在1到p\*q中也正好有（p-1）\*（q-1）个数与n互质。

由剩余系的概念可以知道在p\*q+1到2\*p\*q中也正好有（p-1）\*（q-1）个数与n互质，即每p\*q个数中有（p-1）\*（q-1）个数与n互质。而n中总共有p\*p\*q个p\*q，所以phi（n）的值就是p\*p\*q\*（p-1）\*（q-1）。

使用该phi计算d的值，再计算明文，就能得到结果。

**10. 一眼就解密**

这道题就是一道base64的解密题，这里记录一个解密的在线网站

[Base64 在线编码解码 | Base64 加密解密 - Base64.us](https://base64.us/)

**11. [NewStarCTF 2023 公开赛道]滴啤**

这道rsa的题目中给了dp、e、n和c。

因为

d*e\equiv 1(modphi)

d\equiv dp(mod(p-1))

所以有

d*e=1+k_{1}*(p-1)*(q-1)(1)

d=dp+k_{2}*(p-1)(2)

把（2）式代入（1）式，两边再同时对（p-1）取模，即可得到

dp*e\equiv 1(mod(p-1))

所以有

dp*e-1=k*(p-1)

那么我们就可以尝试计算出dp\*e-1的值，再对其进行分解，以尝试获取到p-1的值。

这一题中，分解后可知该数共有8个素因数，其因数的总数量并不多，完全可以进行遍历。由于p的位数约为512位，而素因数有一个近400位的素数，所以它一定是p-1的一个因数（反证法易证）。那么我们遍历的数量就可以进一步减少了。

使用sagemath进行遍历的代码如下：

p2=2\*13576168482904074972916434606484064894094750779290370955123911937153145555527439626414947690638030717530975726269496353

m=[3,3,3,7,251,133201,52499805387917,2237058624000923]

n=93172788492926438327710592564562854206438712390394636149385608321800134934361353794206624031396988124455847768883785503795521389178814791213054124361007887496351504099772757164211666778414800698976335767027868761735533195880182982358937211282541379697714874313863354097646233575265223978310932841461535936931

for i in range(256):

x=bin(i)[2:].zfill(8)

p=p2

for j in range(len(x)):

if int(x[j])==1:

p=p\*int(x[j])\*int(m[j])

p=p+1

if n%p==0:

print(p)

break



**12. [NewStarCTF 2023 公开赛道]babyencoding**

part 1 of flag: ZmxhZ3tkYXp6bGluZ19lbmNvZGluZyM0ZTBhZDQ=

part 2 of flag: MYYGGYJQHBSDCZJRMQYGMMJQMMYGGN3BMZSTIMRSMZSWCNY=

part 3 of flag: =8S4U,3DR8SDY,C`S-F5F-C(S,S<R-C`Q9F8S87T`

这道题就是三种不同的编码方式。

第一部分中有大小写字母和数字，最后还有一个等号，是base64编码，使用网站可以解决（网站见“一眼就解密”模块）。

第二部分只有大写字母和数字，是base32编码，我用sagemath写了个脚本进行解码：

a='MYYGGYJQHBSDCZJRMQYGMMJQMMYGGN3BMZSTIMRSMZSWCNY'

c=[]

for i in a:

m=ord(i)

if m>0x40:

c.append(int(m-ord('A')))

else:

c.append(int(m-ord('2'))+26)

s=''

for i in c:

a=bin(i)[2:].zfill(5)

s=s+a

print(s)



这个脚本得到的结果为二进制的结果，最后需要进行补零，使其位数为8的倍数。

第三部分的编码方式为UUencode编码，可以利用在线网站进行解密。

可使用网站：[在线工具 - Bugku CTF](https://ctf.bugku.com/tools)

**13. [NewStarCTF 2023 公开赛道]babyxor**

这道题目把flag中的每一个字符与一个未知的key异或之后进行了输出，因此我们需要解出这个key的值。直接猜测flag中的前四个字符为“flag”，把这四个字符与已知的输出值的前三十二位进行一次异或，发现得到的结果是重复的0x8f，因此key的值就是0x8f。

**14. [NewStarCTF 公开赛赛道]eazyxor**

这道题目和13里面一模一样，这里就把sagemath计算key的脚本放一下：

a=0x9b919c9a

b=0x666c6167

def xor(p,q):

p=int(p)

q=int(q)

return p&(~q)|(~p)&q

c=xor(a,b)

print(hex(c))



（最近发现在sagemath中^^可以表示异或，所以其实并不需要写一个异或的函数）

**15. [NewStarCTF 2023 公开赛道]Affine**

这也是一道仿射变换的题目，需要求出key的值（key是一个包含两个元素的列表）。其仿射变换的公式为：

![key[0]*flag[i]+key[1]\equiv output[i](mod256)](data:image/png;base64,)

我们只需要知道flag中两个的两个字母和他们对应的output的值，就能求解出key的值，具体的操作如下：

假设已知x、y，未知数为a、b，有

![a*x[0]+b\equiv y[0](mod256)(1)](data:image/png;base64,)

![a*x[1]+b\equiv y[1](mod256)(2)](data:image/png;base64,)

联立上面的两个式子，用（1）式减去（2）式，得到

![a*(x[0]-x[1])\equiv y[0]-y[1](mod 256)](data:image/png;base64,)

则可计算出a的值为

a=(y[0]-y[1])*(x[0]-x[1])^{-1}(mod256)

再把a的值代入（1）式或（2）式中，即可得到b的值为

b=y-a*x(mod 256)

通过上面的分析，可以用sagemath写出解密的代码：

f1=0x66

f2=0x61

b1=0xdd

b2=0x88

b=b1-b2

f=f1-f2

x=pow(f,-1,256)

a0=(b\*x)%256

a1=(b1-f1\*a0)%256

n=pow(a0,-1,256)

h='dd4388ee428bdddd5865cc66aa5887ffcca966109c66edcca920667a88312064'

c=''

for i in range(len(h)//2):

x='0x'+h[2\*i]+h[2\*i+1]

b=int(x,16)

m=((b-a1)\*n)%256

c=c+chr(m)

print(c)



在“flag”中使用差值计算出f时，结果可能不会和256互质，比如“f”和“l”这两个字符，所以我在选择明文时选择了“f”和“a”这两个字符。

**16. [NewStarCTF 2023 公开赛道]halfcandecode**

这道题是上面的第二题和第六题的结合体，需要用到rsa和MD5。

rsa的部分也可以使用第五题中的网站进行分解。需要注意的是出题人在前半部分flag的后面加入了一些乱码，在解密的时候需要删除。MD5的部分则只要使用第六题的网站就能解决。其中有些MD5的值需要充钱获取，可以使用网站[MD5免费在线解密破解\_MD5在线加密-SOMD5](https://www.somd5.com/)破解。

**17. [NewStarCTF 2023 公开赛道]Rotate Xor**

这道题中k1的加密方法为进行十二轮的循环左移和异或操作，其中每一轮的加密方法可由下面的公式表示，其中roundrotateleft函数为给出的循环左移函数，这里表示循环左移三位。

c=roundrotateleft(m, 3) \bigoplus k2

由此写出每一轮的解密方法， 再用此方法对已知的加密后的k1的密文进行十二轮的解密，就能得到最后的答案。解密方法可用下面的公式表示，其中roundrotateright函数需要自己编写。

m=roundrotateright(c\bigoplus k2, 3)

用sagemath编写下面的代码得到k1的值：

enc\_k1 = 7318833940520128665

k2 = 9982833494309156947

key=enc\_k1

def round\_rotate\_right(num, step):

return ((num) >> step | num << (64-step))&0xffffffffffffffff

for \_ in range(12):

key=xor(key,k2)

key=round\_rotate\_right(key,3)

print(hex(key))



代码中的xor函数在第十四题的脚本中进行了定义。

得到k1后，发现k1的长度与给出的密文ciphertext的长度是不一致的，因此需要知道题目中加密用到的函数xor的用法。这个是从pwn库中引入的函数，我猜测是像分组密码一样分组进行异或，就用最前面的四个字符尝试了一下，发现得到的确实是“flag”。所以按照分组密码的加密方式对密文进行了异或，就得到了答案。

**18. [NewStarCTF 2023 公开赛道]babyNTRU**

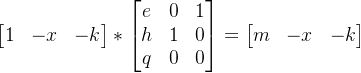
这是一道格密码的题目，需要求解的公式为：

e\equiv x*h+m(modq)

其中e、h、q已知，x和m未知。我们可以把公式改写为：

e-x*h-k*q=m

由此可以写出矩阵计算公式：



对其中3×3的矩阵使用LLL算法，就能得到结果，写出的sagemath代码如下：

h=8916452722821418463248726825721257021744194286874706915832444631771596616116491775091473142798867278598586482678387668986764461265131119164500473719939894343163496325556340181429675937641495981353857724627081847304246987074303722642172988864138967404024201246050387152854001746763104417773214408906879366958729744259612777257542351501592019483745621824894790096639205771421560295175633152877667720038396154571697861326821483170835238092879747297506606983322890706220824261581533324824858599082611886026668788577757970984892292609271082176311433507931993672945925883985629311514143607457603297458439759594085898425992

q=31985842636498685945330905726539498901443694955736332073639744466389039373143618920511122288844282849407290205804991634167816417468703459229138891348115191921395278336695684210437130681337971686008048054340499654721317721241239990701099685207253476642931586563363638141636011941268962999641130263828151538489139254625099330199557503153680089387538863574480134898211311252227463870838947777479309928195791241005127445821671684607237706849308372923372795573732000365072815112119533702614620325238183899266147682193892866330678076925199674554569018103164228278742151778832319406135513140669049734660019551179692615505961

e=20041713613876382007969284056698149007154248857420752520496829246324512197188211029665990713599667984019715503486507126224558092176392282486689347953069815123212779090783909545244160318938357529307482025697769394114967028564546355310883670462197528011181768588878447856875173263800885048676190978206851268887445527785387532167370943745180538168965461612097037041570912365648125449804109299630958840398397721916860876687808474004391843869813396858468730877627733234832744328768443830669469345926766882446378765847334421595034470639171397587395341977453536859946410431252287203312913117023084978959318406160721042580688

mat=[[e,0,1],[h,1,0],[q,0,0]]

M = Matrix(QQ,mat)

lll=M.LLL()

print((lll[0]))



其中打印出来的结果即为上面的公式中等号右边的向量。

**19. [GKCTF2020]babycrypto**

这道题中的n可以直接使用第五题中的网站进行分解。

**20. [NewStarCTF 2023 公开赛道]Smart**

这是一道ecc的题目，根据提示可以使用smart attack的方法解题。使用sagemath可以编写出下面的代码：

p = 75206427479775622966537995406541077245842499523456803092204668034148875719001

a = 40399280641537685263236367744605671534251002649301968428998107181223348036480

b = 34830673418515139976377184302022321848201537906033092355749226925568830384464

E = EllipticCurve(GF(p),[a,b])

P=E(63199291976729017585116731422181573663076311513240158412108878460234764025898,11977959928854309700611217102917186587242105343137383979364679606977824228558)

Q=E(75017275378438543246214954287362349176908042127439117734318700769768512624429,39521483276009738115474714281626894361123804837783117725653243818498259351984)

def SmartAttack(P,Q,p):

E = P.curve()

Eqp = EllipticCurve(Qp(p, 2), [ ZZ(t) + randint(0,p)\*p for t in E.a\_invariants() ])

P\_Qps = Eqp.lift\_x(ZZ(P.xy()[0]), all=True)

for P\_Qp in P\_Qps:

if GF(p)(P\_Qp.xy()[1]) == P.xy()[1]:

break

Q\_Qps = Eqp.lift\_x(ZZ(Q.xy()[0]), all=True)

for Q\_Qp in Q\_Qps:

if GF(p)(Q\_Qp.xy()[1]) == Q.xy()[1]:

break

p\_times\_P = p\*P\_Qp

p\_times\_Q = p\*Q\_Qp

x\_P,y\_P = p\_times\_P.xy()

x\_Q,y\_Q = p\_times\_Q.xy()

phi\_P = -(x\_P/y\_P)

phi\_Q = -(x\_Q/y\_Q)

k = phi\_Q/phi\_P

return ZZ(k)

r = SmartAttack(P, Q, p)

print(r)



（函数脚本来自Lazzaro @ [分类 | Lazzaro](https://lazzzaro.github.io/categories/)）

**21. [V&N2020 公开赛]Backtrace**

这是一道向前还原梅森旋转算法产生的随机数的题目，可以使用python3中已有的库中的函数进行解密，脚本如下：

from extend\_mt19937\_predictor import ExtendMT19937Predictor

predictor = ExtendMT19937Predictor()

x=[1954915070, 880079313, 360906530, 1379349539, 2990290602, 2779090715, 1195886637, 26260733, 1583365168, 3665368407, 32035604, 23154243, 4175855420, 4087729539, 4061778341, 1842490363, 24026296, 506531036, 3305517750, 2480717543, 3912881681, 2674873504, 3352869450, 3592419601, 3036323492, 1082383946, 3598180449, 547795339, 549590551, 3028169916, 2073824705, 1647033805, 1120414536, 3923037909, 3779751646, 2143420925, 3908526958, 3918342792, 3262098248, 3804166870, 1736455411, 2417662325, 3135605338, 428676667, 3664109984, 2807365630, 592564266, 1511929376, 3720241883, 3519131811, 88062728, 2452008344, 1376675709, 666109304, 4110417201, 4220702851, 3778634814, 4211738848, 715753516, 2658852621, 2979894716, 1247212269, 4178194405, 1781101726, 1041419594, 1709310125, 3813614389, 458927554, 3764695336, 2441752341, 1253881549, 1225143052, 3557368845, 3838600385, 704097569, 1795077589, 1307245977, 1717115299, 983409487, 2129778243, 3432359775, 1630272089, 1044229449, 1934797629, 256992360, 739187055, 2003823253, 2520758703, 4188045315, 1831601825, 1613890879, 43212525, 286036422, 730199546, 2111917986, 2296406771, 1523494866, 22453673, 1713781269, 289582587, 1190283387, 215790012, 3341532120, 4143468851, 3911058947, 1744160183, 1737482991, 995300206, 1209025742, 2548392860, 2112397256, 2328723112, 3252904928, 2986221228, 920151702, 2869410691, 297356863, 1113160659, 1723226249, 91041567, 1364664100, 1603980497, 1950077086, 2463918675, 1270243131, 2690292580, 1422349327, 862261952, 967425382, 2141373124, 784240543, 3431833360, 490754262, 3848551199, 3374708506, 1730158500, 1679948052, 1037254337, 1542986195, 3179769778, 525752391, 2138533768, 4060094504, 3749351453, 1653628056, 3891244202, 587949751, 2117666819, 3989217878, 236634714, 347834593, 36437255, 857428838, 802901512, 2150725842, 1293111161, 149116333, 1195588424, 2803507599, 1721551167, 3936542960, 3957883800, 126595404, 871451152, 565527716, 1622812264, 2809311587, 3016835340, 162902761, 4206750381, 3921119290, 1504723782, 3293690462, 3083506005, 811746919, 3455687374, 767562328, 3777952933, 2172820246, 2605716515, 1756246148, 2243128034, 1344314147, 3763482539, 2562146979, 3163206461, 3618311150, 1189035477, 7231879, 815006324, 195187971, 1578170963, 1078365572, 463220548, 92395730, 2138595439, 1322999823, 2469535746, 2749391749, 3433099223, 3983509306, 2111542870, 3745476971, 919481220, 1230712391, 3811227914, 1736577469, 2419466842, 1541026951, 398886450, 1510256083, 3548576351, 1983859038, 938962757, 172522539, 1019282319, 332128756, 3299336175, 2841132609, 2197477198, 3371426587, 1442166578, 3238850747, 1747647226, 2283411662, 1204012944, 3295272849, 1586724848, 4022995561, 3742366393, 79843478, 2109666146, 2406199134, 420281144, 1098025843, 519159390, 536903419, 2387574872, 585556144, 4037282670, 1816756967, 3987753062, 1495823125, 1010457539, 720442860, 8816820, 1812416478, 3356964625, 158012176, 902760599, 1749660162, 4160594328, 393212714, 877871013, 3616848204, 4126377209, 4259811533, 1989917399, 582786867, 1093424901, 1333179279, 1800875923, 1598085790, 627659277, 32688263, 2222145572, 938619663, 1467384991, 2777852620, 4233202126, 295416244, 3515893229, 572401093, 2526938242, 3851968999, 3599541582, 91522073, 1825233519, 2339315265, 4123218487, 2650729621, 287304953, 1423724344, 2908364142, 2748769808, 3418694754, 3917094690, 2646486860, 2858627000, 3305171165, 3401244148, 2180778517, 1737578689, 1322497258, 4144511518, 1856725795, 1477318062, 911248289, 1535195390, 3144994767, 4101205532, 3808591560, 2213985378, 3252595633, 2337367507, 534460335, 2441670955, 1990989375, 3808438725, 3804201721, 3999376545, 1014481854, 2090372701, 2927739146, 1300032996, 534772267, 3335311855, 664080148, 3129094809, 1075386087, 3651483413, 3363859879, 3226823199, 2964465151, 3285786367, 2581149182, 262038551, 2762741963, 922363707, 1260642832, 1183500610, 2803108134, 3592591385, 1484954269, 161632434, 3860482129, 3556119993, 1784107097, 1861391286, 3891693742, 1469562815, 4052616817, 1735123567, 3981493161, 1172433134, 4142248853, 1586222290, 3539037789, 1427799338, 1876694928, 1200583260, 1752440053, 291416352, 2106928780, 936954012, 197470641, 2337076377, 1421119809, 3135686533, 398635959, 4172800048, 399974723, 287928291, 529512170, 1921498685, 567654083, 1812953083, 4068864497, 2881211664, 2429816180, 4084286231, 1928525086, 3886022768, 1443924813, 1820391016, 3624477251, 1452182198, 3501771914, 3307175554, 1233333926, 4242916900, 1054132061, 1725357747, 4125133480, 2860075310, 3091839411, 3820381997, 1335282777, 2927671741, 2241788945, 3488880264, 2095731302, 1214404049, 592041371, 3490153501, 238412730, 85745765, 1498653208, 288707861, 2868112869, 491758034, 375898933, 529291211, 1526086068, 1434399655, 758215890, 1276881381, 3266726291, 3413372766, 3488548535, 145011592, 45063434, 3505575389, 1780705822, 2662969703, 856437165, 3634545960, 2660427148, 977850868, 1582014613, 4189751436, 866116713, 1216606975, 420500706, 2031559891, 1026985902, 3713148233, 494872740, 590852345, 1318692699, 1051590564, 365025448, 210057222, 4081233470, 836354250, 2395488189, 3089620058, 3371189485, 2492384566, 317404020, 3448768638, 1914822123, 2007144480, 3375740426, 4089530814, 1711142699, 1845862399, 2063343799, 3364295943, 3659690944, 546211494, 1414275727, 3044491383, 3050919928, 3390640243, 3843994607, 4283790197, 1589762674, 2433218003, 3728021067, 1131986129, 2737833357, 2800289380, 2753380618, 900710707, 4131195838, 2668161437, 940253312, 4137815270, 939482424, 1748091577, 3546489510, 3421396386, 4148510311, 4165939267, 726262463, 3050178325, 3177146914, 3563060942, 1534352901, 3355774700, 3917022484, 2589474555, 3331795641, 1102689414, 3183908672, 1002812809, 152821999, 2292133358, 1383462496, 1723940974, 1109447662, 3735817236, 2653647203, 4026819614, 1197259783, 2243347947, 4231464616, 2510151911, 4222963505, 1139252638, 756514223, 3417745074, 1468496068, 3161226317, 1246424725, 1872920987, 2647633430, 3258968525, 253034280, 1645087047, 3247041431, 1054772676, 379436873, 2600620229, 2594042244, 661729386, 1665438570, 1017686809, 233096083, 845378242, 3372582854, 410404381, 2007910443, 4292394895, 3237528904, 3176696608, 3307483127, 1197245101, 162556298, 2316117728, 1197610943, 2268702243, 3342707155, 3853282575, 916356664, 679322020, 3607041524, 437255662, 1779501694, 2192606690, 3849753853, 2382746933, 3266572424, 1897113113, 3929458586, 2295574218, 3398667663, 489651240, 554749675, 2685580250, 271948130, 3301333969, 2402822376, 2006322914, 3300613934, 2720365129, 1219841771, 253738666, 4062523722, 908057314, 734682735, 3993608743, 2005954106, 4196754468, 2180968904, 2927088293, 3201719969, 3258216746, 1697012102, 451497250, 4140321978, 1626314925, 2634645740, 2650768687, 4037717892, 3530817306, 3154301628, 607258212, 1138057824, 1483931056, 1779526200, 772778910, 3269411312, 2168622217, 3392463299, 2129731229, 663170712, 3481534219, 3813779502, 4003014679, 1417316990, 3758399028, 3050577451, 4287031625, 4219536484, 3110762846, 3162149172, 2925735942, 3222005405, 3901793171, 2438213679, 2761390133, 2868086632, 2262419687, 2548717542, 662411455, 2628823526, 1947247956, 192432795, 356145859, 1388455292, 1337879771, 1318475009, 4194024566, 3055182968, 3818161981, 4122245354, 1163215145, 4064532043, 1427071812, 3338509721, 996878503, 4035314277, 4009039326, 916055062, 3817247526, 1111659075, 2245077089, 1027546064, 3342483659, 1793234141, 1571978271, 1560889567, 3201761295, 2525084128, 1111852520, 147467714, 3308281891, 2593100275, 1890216029, 2245520263, 3905179223, 4166586105, 4058069805, 2234254266, 897808812, 1276352683, 4089183900, 1496670572, 2979089043, 1110628736, 3320051082, 3285541627, 4026820048, 414845486, 2742275697, 2611076257, 2183923292, 715603118, 3655611426, 1042307613, 1538032337, 3770267700, 2701884650, 807662565, 1412021992, 2190384229, 636193740, 2262474952, 1037800098, 2251081302, 2110190277, 1818325893, 357637012, 3220347372, 3779111771, 2794951662, 2053531638, 3865719745, 1117551032, 2005076639, 3052789596, 1850346070, 3250668743, 3748374363, 3366024365, 1797731176, 4167453426, 3273801260, 270498944, 3409183492, 2435803649, 4214667811, 3331238798, 2204118424, 124311085, 821069678, 402529460, 4241047514, 3597966193, 2484307360, 745429577, 1575021475, 2880532249, 1653073618, 2042789973, 1988689809, 2986312451, 998861281, 3049071914, 3105090905, 1948439372, 1812644387, 2463127929, 4111481103, 986806020, 1163777124, 2780762085, 2078725375, 2407568318, 3342714967, 2772408458, 490512106, 143570762, 1232716574, 519717220, 158618961, 1436523816, 4216701765, 307191458, 2912138910, 3468252635, 386069354, 1634782909, 392302934, 781725611, 3497023560, 4225924396, 1795951064, 3928854990, 1539311975, 3391120265, 232461937, 4184782267, 1731411767, 1647711068, 2103390822, 3545452835, 3898032707, 2174552994, 1311574900, 822414312, 1839460614, 1762983458, 2140522204, 417114171, 2942594202, 3153277613, 706508900, 1240369844, 2262617676, 2812407518, 1840197328, 1824634385, 859165172, 1576784364, 585466601, 1765607687, 3423031247, 1284381057, 3382474784, 3123254074, 2194861604, 495637126, 2802357617, 1226312598, 1818564904, 2754819901, 3465896582, 414214415, 1786854851, 3480143070, 1969087474, 2384690047, 764711574, 888969866, 291549264, 1623430983, 1065342949, 3771290713, 2607894650, 2849509696, 633784811, 592845780, 3632194205, 889199132, 4102077198, 29170934, 34456705, 2613055456, 1078880049, 1106268704, 22020620, 2586845047, 2925005739, 1854339682, 1649915319, 4071303129, 4140750565, 897249960, 1279599439, 1549858622, 2530236320, 357028938, 187680170, 2825532344, 3474432045, 1459157318, 1078630130, 522441272, 1021727782, 756127542, 670068161, 1933389429, 2126666549, 3233405034, 691338491, 617019618, 1572960014, 2476894878, 3114330689, 190756003, 343951393, 254882512, 3232970324, 1278199314, 136867483, 1288546656, 3526700531, 2819009094, 3204612611, 1139498748, 3875800906, 255305389, 2504064025, 3125108091, 2176350989, 2469637779, 3339575355, 2999412374, 1282126909, 1551339993, 3573997525, 3812552026, 2137007302, 3302434775, 1777310304, 2053104074, 2328784509, 2188979055, 1553054099, 857550765, 3714986904, 21124773, 1651956692, 220434238, 483698922, 1770276048, 3987677675, 2078271890, 4127930299, 913495199, 1552004068, 318877141, 3540501698, 1999579835, 935784539, 390615388, 2903539749, 3837464977, 3922761762, 108363706, 767261731, 592768246, 100601999, 767766613, 1170959252, 723154566, 2277226093, 1249730541, 4207253124, 3412009687, 2427450601, 3312907551, 1823542002, 2660960835, 3032107443, 3010173489, 3100247405, 1939956123, 630159399, 2588902313, 2901045261, 3788987361, 802990870, 333237497, 1240204630, 2845468633, 3360840386, 3763275014, 2632404055, 3815462044, 3818227716, 1049832368, 54028478, 758732492, 1815724078, 1246513687, 2374051260, 807659196, 3751209486, 3409043148, 559342443, 2430700654, 398612276, 241887832, 1939697736, 2288135888, 655726394, 1451329969, 1119873489, 799074425, 2253637924, 2053706296, 1548769628, 3867902836, 1247111501, 1004835651, 2850968664, 653021090, 1292848338, 154967348, 4194736783, 3102822469, 2261232318, 1657165783, 182571243, 2866887396, 4290974852, 2261061037, 3629684605, 2482236569, 1916603941, 3382603226, 999768427, 3171175558, 1709542380, 2086924233, 3964979595, 85833410, 1122206428, 1432694206, 4230090721, 2632877999, 834731838, 2344308939, 2399519878, 4086951271, 2681850703, 3113324115, 2045734372, 1980879301, 2869222671, 2270508944, 3549736736, 3871179085, 1302255582, 4224050094, 2185529812, 165196590, 2581134223, 579189627, 4045953032, 2910507974, 633486844, 4092863379, 279952867, 654229974, 3590879432, 3494667849, 3762736646, 4057616614, 1378958651, 3737744634, 2799387186, 127072071, 1886119253, 3542028713, 3083942542]

for i in range(624):

predictor.setrandbits(x[i],32)

a= [predictor.backtrack\_getrandbits(32) for \_ in range(628)]

print(a)



（函数脚本来自 Lazzaro @ [分类 | Lazzaro](https://lazzzaro.github.io/categories/)）

打印出来的a中最后四个数即为所需的向前还原的随机数。按提示即可得到flag。

本题中，由于给的output中的随机数没有以列表的形式给出，所以我用python3写了一个读取整数的脚本：

import re

with open('output.txt', 'r') as file:

content = file.read()

numbers = re.findall(r'\d+', content)

numbers = [int(num) for num in numbers]

print(numbers)



**22. [NewStarCTF 公开赛赛道]Affine**

这道题和第十五题一模一样，这次还提示了要使用flag的格式。

**23. 变异凯撒**

这道题的题目为：

加密密文：afZ\_r9VYfScOeO\_UL^RWUc  
格式：flag{ }

说是变异凯撒，那肯定是和凯撒密码有关。一开始我以为密文中去掉下划线之后剩下的部分进行凯撒密码的变换之后可以得到三个单词，最后再包上flag{}进行提交。结果穷举了半天都没有找到什么合适的结果。

后来想起来其他题目好像都是说包上flag{}进行提交，而这个只说了格式是flag{}，我猜测是一个已知明文攻击的题目，就用前四个密文字符的ASCII码减去了flag对应的ASCII码，这才找到了这道题目的加密方法。用sagemath写的解密脚本如下：

a='afZ\_r9VYfScOeO\_UL^RWUc'

b=''

for i in range(len(a)):

b=b+chr(ord(a[i])+5+i)

print(b)



**24. Quoted-printable**

[Quoted-printable\_ 引用可打印-CSDN博客](https://blog.csdn.net/aaum556/article/details/132263994?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522172224834616800178573185%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=172224834616800178573185&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-1-132263994-null-null.142%5Ev100%5Epc_search_result_base2&utm_term=Quoted-printable&spm=1018.2226.3001.4187)这篇博客中有详细的介绍，具体的编码转换可以使用下面的网址：[Quoted-printable编码|Quoted-printable解码|Quoted-printable编码原26理介绍--查错网](http://web.chacuo.net/charsetquotedprintable)

**25. 篱笆墙的影子**

题目为felhaagv{ewtehtehfilnakgw}

根据提示是栅栏密码，找到{的位置，发现是第9位，前面单数位的字符刚好组成“flag”，很明显栏数为2，简单用sagemath写个脚本解密：

a='felhaagv{ewtehtehfilnakgw}'

b=''

for i in range(len(a)//2):

b=b+a[2\*i]

for i in range(len(a)//2):

b=b+a[2\*i+1]

print(b)



**26. Rabbit**

具体的编码方法可以参考[BUUCTF Rabbit 1-CSDN博客](https://blog.csdn.net/YueXuan_521/article/details/131316653?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522172224992716800211582864%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=172224992716800211582864&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-5-131316653-null-null.142%5Ev100%5Epc_search_result_base2&utm_term=rabbit%E7%BC%96%E7%A0%81&spm=1018.2226.3001.4187)这篇博客，其中用到的在线解密工具为[Rabbit加密-Rabbit解密-在线Rabbit加密解密工具](http://www.jsons.cn/rabbitencrypt/)

**27. 丢失的MD5**

打开附件发现一个python代码，尝试运行，发现有报错。把报错的地方修改后即可运行得到正确答案，修改后的代码如下：

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').encode('utf-8'))

des=m.hexdigest()

if 'e9032' in des and 'da' in des and '911513' in des:

print(des)



**28. [GKCTF2020]汉字的秘密**

这道题中用汉字的笔划出头的数量来表示数字。

比如说“口”，没有出头，表示0，“由”有一个出头的笔划，表示1。把题目中涉及的所有汉字的笔划数计算出来，然后可以用sagemath写出下面的脚本解密：

a='田口由中人工大土士王夫井羊壮'

b='00123455567899'

c='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 c:

s=s+b[a.index(i)]

print(s)



这样能得到一串数字‘69746267118837277865571578257646351107’，然后看给的文档里面汉字的组合情况，发现有些是两个汉字连在一起，有些是三个汉字连在一起，感觉像是每个字符的ASCII编码。根据提示将这些数值与flag中的字符的ASCII进行对比，发现这道题的加密方式与第二十三题变异凯撒差不多，于是可以用sagemath写出解密代码如下：

m='69746267118837277865571578257646351107'

s=''

for i in range(len(m)//2-2):

if i <4:

x=int(m[2\*i])\*10+int(m[2\*i+1])+i+1

s=s+chr(x)

elif i ==4:

x=100+int(m[2\*i+1])\*10+int(m[2\*i+2])+i+1

s=s+chr(x)

else:

x=int(m[2\*i+1])\*10+int(m[2\*i+2])+i+1

s=s+chr(x)

s=s+chr(107+len(m)//2-1)

print(s)



最后得到的结果是大写的字母，按提示需要改成小写的字母进行提交。

**29. [NewStarCTF 2023 公开赛道]Rabin's RSA**

在rsa的rabin加密中，e取2，需要知道p、q的值才能解密，用sagemath写的具体的解密脚本如下：

p=13934102561950901579

q=14450452739004884887

n=201354090531918389422241515534761536573

c=20442989381348880630046435751193745753

inv\_p=int(pow(p,-1,q))

inv\_q=int(pow(q,-1,p))

mp=int(pow(c,(p+1)//4,p))

mq=int(pow(c,(q+1)//4,q))

a=(inv\_p\*p\*mq+inv\_q\*q\*mp)%n

b=n-int(a)

c=(inv\_p\*p\*mq-inv\_q\*q\*mp)%n

d=n-int(c)

result=[a,b,c,d]

for m in result:

t=hex(m)

g=str(t)

l=len(g)//2-1

s=''

for i in range(l):

s=s+chr(int(g[2+2\*i]+g[3+2\*i],16))

print(s)



运行后得到的四段中只有一段是有效的flag，提交这一段即可。

**30. [BJDCTF 2nd]燕言燕语-y1ng**

首先把给的十六进制的数用ASCII码编码为对应的字符，得到：

yanzi ZJQ{xilzv\_iqssuhoc\_suzjg}

后面的部分显然是flag经过维吉尼亚密码变换后得到的结果，尝试猜测明文。首先这个比赛是BJDCTF，所以ZJQ大概率是由BJD变换过来的。再看到前面的“yanzi”刚好有五个字符，而大括号里面下划线隔开的部分中也有两个部分刚好是由五个字符组成的，可以合理猜测这两个里面有一个就是“yanzi”这五个字符。先尝试前面这五个字符，发现正好能解密出一句完整的话，成功获取到flag。

这道题的关键在于明文的猜测，也就是需要理解出题人给的“yanzi”这个提示的含义。因为在flag里面组成的话一般都是英语的，所以我一开始以为这是什么特殊的加密方法。百度之后没有什么收获，最后才知道这个提示原来就是明文的一部分。

**31. 信息化时代的步伐**

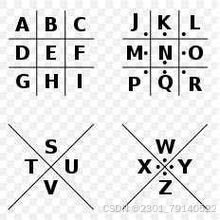
这是一种中文电码，可以使用下面的网站解码：[中文电码查询 Chinese Commercial Code - 标准电报码免费在线查询|姓名电码|美国签证电码](http://code.mcdvisa.com/)

**32. 凯撒？替换？呵呵!**

记录一个可以很方便列举凯撒密码的可能性的网站：[quipqiup - cryptoquip and cryptogram solver](https://quipqiup.com/)

**33. 萌萌哒的八戒**

看到八戒和密码就知道是猪圈密码，这里记录一个猪圈密码的图片。

​编辑

（图片来自[CTF密码学常见加密解密总结\_wydfxjmgvswodododod-CSDN博客](https://blog.csdn.net/qq_40837276/article/details/83080460?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522172240020116800180633253%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=172240020116800180633253&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-83080460-null-null.142%5Ev100%5Epc_search_result_base2&utm_term=%E7%8C%AA%E5%9C%88%E5%AF%86%E7%A0%81&spm=1018.2226.3001.4187)）

**34. 传统知识+古典密码**

这道题中使用到了我国传统的天干地支，这里附一张天干地支的计数表：

​编辑

题目中提示说“+甲子”，甲子在表里面表示1，但是把解密出来的数字加一并没有什么效果，所以想到甲子还有另一层意思，就是一轮，也就是六十，再把解密出来的数字加六十，用ASCII码找到对应的英文字符，即可得到一串没有任何意义的字母。

看到题目里有“古典密码”的提示，使用[在线工具 - Bugku CTF](https://ctf.bugku.com/tools)中的栅栏加密和凯撒加密的工具进行穷举，终于找到了一串有含义的中文拼音：SHUANGYU，包上flag提交即可。

**35. [AFCTF2018]Morse**

这也是一道摩斯编码的题目，由于摩斯码较长，找了一个解码的网站：[摩尔斯电码转换器：在线进行摩尔斯电码和英文字母的相互转换，可以把莫尔斯电码转换为英文字母，也可把英文字母转换为摩尔斯电码](http://www.zhongguosou.com/zonghe/moErSiCodeConverter.aspx)

**36.[NewStarCTF 2023 公开赛道]RSA Variation II**

根据提示知道是一种Schmidt-Samoa密码系统，可以通过推导得到计算flag的方法。

首先列出题目条件：

N=p^2*q(1)

N*d\equiv 1(mod(p-1)*(q-1)\div gcd(p-1,q-1))(2)

c\equiv m^N(modN)(3)

由（2）式可知存在k使得

N*d=1+k*(p-1)*(q-1)\div gcd(p-1,q-1)

不妨设gcd(p-1,q-1)为x，那么有

x*N*d=x+k*(p-1)*(q-1)(4)

在传统的rsa加密中，由费马小定理我们可以得到：

1\equiv a^{(p-1)*(q-1)}(modp*q)

那么我们任取一个a，利用（4）式可以知道

a^{x*N*d}\equiv a^{x+k*(p-1)*(q-1)}\equiv a^x*a^{k*(p-1)*(q-1)}\equiv a^x(mod p*q)

将上式最左边和最右边同时乘以-x次方，可以得到

a^{N*d}\equiv a(mod p*q)

于是知道，对于任意一个a来说，存在t使得

t*p*q=a^{N*d}-a(5)

那么我们任取两个不同的a，计算出满足（5）式的两个不同的数值，然后求他们的最大公约数，就能得到p\*q的值，再利用（1）式就能计算出p、q的值。用sagemath写出脚本：

N = 1768427447158131856514034889456397424027937796617829756303525705316152314769129050888899742667986532346611229157207778487065194513722005516611969754197481310330149721054855689646133721600838194741123290410384315980339516947257172981002480414254023253269098539962527834174781356657779988761754582343096332391763560921491414520707112852896782970123018263505426447126195645371941116395659369152654368118569516482251442513192892626222576419747048343942947570016045016127917578272819812760632788343321742583353340158009324794626006731057267603803701663256706597904789047060978427573361035171008822467120148227698893238773305320215769410594974360573727150122036666987718934166622785421464647946084162895084248352643721808444370307254417501852264572985908550839933862563001186477021313236113690793843893640190378131373214104044465633483953616402680853776480712599669132572907096151664916118185486737463253559093537311036517461749439

d = 20650646933118544225095544552373007455928574480175801658168105227037950105642248948645762488881219576174131624593293487325329703919313156659700002234392400636474610143032745113473842675857323774566945229148664969659797779146488402588937762391470971617163496433008501858907585683428652637958844902909796849080799141999490231877378863244093900363251415972834146031490928923962271054053278056347181254936750536280638321211545167520935870220829786490686826062142415755063724639110568511969041175019898031990455911525941036727091961083201123910761290998968240338217895275414072475701909497518616112236380389851984377079

pq = gcd(Integer(pow(2,d\*N,N)-2),Integer(pow(3,d\*N,N)-3))

p=N//pq

q=pq//p

print(p)

print(q)



接下来需要变换（3）式尝试得到m的值。

结合（1）式和（3）式，我们知道

c\equiv m^N(modp*q)

等式两边同时乘以d\*x次方，则有

c^{d*x}\equiv m^{d*x*N}(mod p*q)

结合（4）式可以得到

c^{d*x}\equiv m^{x+k*(p-1)*(q-1)}\equiv m^{x}*m^{k*(p-1)*(q-1)}\equiv m^{x}(modp*q)

两边再同时乘以-x次方，即可得到

m\equiv c^d(modp*q)

利用这个式子就能计算出m的值，从而得到flag。

**37. [NewStarCTF 2023 公开赛道]Small d**

根据提示需要使用维纳攻击的方法，到网上找了一个相关的脚本，用sagemath运行一下：

def rational\_to\_contfrac(x,y):

a = x // y

pquotients = [a]

while a \* y != x:

x, y = y, x - a \* y

a = x // y

pquotients.append(a)

return pquotients

def convergents\_from\_contfrac(frac):

convs = [];

for i in range(len(frac)): convs.append(contfrac\_to\_rational(frac[0 : i]))

return convs

def contfrac\_to\_rational (frac):

if len(frac) == 0: return (0,1)

num = frac[-1]

denom = 1

for \_ in range(-2, -len(frac) - 1, -1): num, denom = frac[\_] \* num + denom, num

return (num, denom)

c = 6755916696778185952300108824880341673727005249517850628424982499865744864158808968764135637141068930913626093598728925195859592078242679206690525678584698906782028671968557701271591419982370839581872779561897896707128815668722609285484978303216863236997021197576337940204757331749701872808443246927772977500576853559531421931943600185923610329322219591977644573509755483679059951426686170296018798771243136530651597181988040668586240449099412301454312937065604961224359235038190145852108473520413909014198600434679037524165523422401364208450631557380207996597981309168360160658308982745545442756884931141501387954248

e = 8614531087131806536072176126608505396485998912193090420094510792595101158240453985055053653848556325011409922394711124558383619830290017950912353027270400567568622816245822324422993074690183971093882640779808546479195604743230137113293752897968332220989640710311998150108315298333817030634179487075421403617790823560886688860928133117536724977888683732478708628314857313700596522339509581915323452695136877802816003353853220986492007970183551041303875958750496892867954477510966708935358534322867404860267180294538231734184176727805289746004999969923736528783436876728104351783351879340959568183101515294393048651825

n = 19873634983456087520110552277450497529248494581902299327237268030756398057752510103012336452522030173329321726779935832106030157682672262548076895370443461558851584951681093787821035488952691034250115440441807557595256984719995983158595843451037546929918777883675020571945533922321514120075488490479009468943286990002735169371404973284096869826357659027627815888558391520276866122370551115223282637855894202170474955274129276356625364663165723431215981184996513023372433862053624792195361271141451880123090158644095287045862204954829998614717677163841391272754122687961264723993880239407106030370047794145123292991433

def egcd(a, b):

if a == 0: return (b, 0, 1)

g, x, y = egcd(b % a, a)

return (g, y - (b // a) \* x, x)

def mod\_inv(a, m):

g, x, \_ = egcd(a, m)

return (x + m) % m

def isqrt(n):

x = n

y = (x + 1) // 2

while y < x:

x = y

y = (x + n // x) // 2

return x

def crack\_rsa(e, n):

frac = rational\_to\_contfrac(e, n)

convergents = convergents\_from\_contfrac(frac)

for (k, d) in convergents:

if k != 0 and (e \* d - 1) % k == 0:

phi = (e \* d - 1) // k

s = n - phi + 1

# check if x\*x - s\*x + n = 0 has integer roots

D = s \* s - 4 \* n

if D >= 0:

sq = isqrt(D)

if sq \* sq == D and (s + sq) % 2 == 0: return d

d = crack\_rsa(e, n)

m = hex(pow(c, d, n))[2:]

print(bytes.fromhex(m))



（函数脚本来自Lazzaro @ [分类 | Lazzaro](https://lazzzaro.github.io/categories/)）

**38. [NewStarCTF 公开赛赛道]ezRabin**

这道题里面e使用了4，是2的两倍，所以只要使用两次rabin加密的解密算法就能得到结果，直接上sagemath的脚本：

c=59087040011818617875466940950576089096932769518087477304162753047334728508009365510335057824251636964132317478310267427589970177277870220660958570994888152191522928881774614096675980017700457666192609573774572571582962861504174396725705862549311100229145101667835438230371282904888448863223898642183925834109

p=10522889477508921233145726452630168129218487981917965097647277937267556441871668611904567713868254050044587941828674788953975031679913879970887998582514571

q=11287822338267163056031463255265099337492571870189068887689824393221951058498526362126606231275830844407608185240702408947800715624427717739233431252556379

n=p\*q

inv\_p=int(pow(p,-1,q))

inv\_q=int(pow(q,-1,p))

mp=int(pow(c,(p+1)//4,p))

mq=int(pow(c,(q+1)//4,q))

a=(inv\_p\*p\*mq+inv\_q\*q\*mp)%n

b=n-int(a)

c=(inv\_p\*p\*mq-inv\_q\*q\*mp)%n

d=n-int(c)

result=[a,b,c,d]

re=[]

for m in result:

mp=int(pow(m,(p+1)//4,p))

mq=int(pow(m,(q+1)//4,q))

a=(inv\_p\*p\*mq+inv\_q\*q\*mp)%n

b=n-int(a)

c=(inv\_p\*p\*mq-inv\_q\*q\*mp)%n

d=n-int(c)

re.append(a)

re.append(b)

re.append(c)

re.append(d)

for m in re:

m=m>>300

t=hex(m)

g=str(t)

l=len(g)//2-1

s=''

for i in range(l):

s=s+chr(int(g[2+2\*i]+g[3+2\*i],16))

print(s)



**39. [NewStarCTF 2023 公开赛道]partial decrypt**

首先在rsa中，我们有

c\equiv m^e(modn)

那么就有

c\equiv m^e(modp)

c\equiv m^e(modq)

考虑模p的情况，对于任意的a有

a^{p-1}\equiv a^{dp*e}\equiv 1(modp)

所以

m\equiv m_{1}\equiv c^{dp}(mod p)

即存在k_{1}使得

m=m_1+k_1*p(1)

同理，存在k_{2}使得

m=m_2+k_2*q(2)

题中已知

h\equiv q_{inv}*(m_1-m_2)(mod p)(3)

把（1）、（2）式代入（3）中，可以得到

h\equiv q_{inv}*(k_2*q-k_1*p)\equiv q_{inv}*q*k_2\equiv k_2(modp)

此时发现，我们知道了m_2、k_2、q的值，可以由（2）式计算出m的值了。

**40. [V&N2020 公开赛]CRT**

这是一道扩展的中国剩余定理的题目，直接上大佬的脚本得到所需求的数的模数和取模之后的值，然后遍历所有可能的结果，寻找符合条件的结果，即为最终的flag。下面是修改后sagemath的脚本：

import hashlib

def CRT(mi, ai):

curm, cura = mi[0], ai[0]

for (m, a) in zip(mi[1:], ai[1:]):

d =gcd(curm, m)

c = a - cura

assert (c % d == 0)

K = c // d \* gmpy2.invert(curm // d, m // d)

cura += curm \* K

curm = curm \* m // d

return (cura % curm, curm)

ms = [284461942441737992421992210219060544764, 218436209063777179204189567410606431578, 288673438109933649911276214358963643204, 239232622368515797881077917549177081575, 206264514127207567149705234795160750411, 338915547568169045185589241329271490503, 246545359356590592172327146579550739141, 219686182542160835171493232381209438048]

cs = [273520784183505348818648859874365852523, 128223029008039086716133583343107528289, 5111091025406771271167772696866083419, 33462335595116820423587878784664448439, 145377705960376589843356778052388633917, 128158421725856807614557926615949143594, 230664008267846531848877293149791626711, 94549019966480959688919233343793910003]

(x,y)=CRT(ms,cs)

x=int(x)

y=int(y)

x=331928385895936850327248455618550267979226366540503309505403897762924348214252184339508819167395418477591466337317495869131189862837240599867138359384794556128835111222068761517340369160010091387517644467690158341517036316714577728109817419385351326590048046021467528943086247709683222909672931154665939

y=466244535277126133494171720905467305227449898136540531771165702728811600035193763296997575538712880600873766336223021102495903920660547449598419752724436481022338453912400904360309909376397024992632893902959636099041017419464716113173886710394215080355542879719231888389121865360597540403463754208728800

k=1

while True:

flag = "flag{" + hashlib.sha256(str(x+k\*y).encode()).hexdigest() + "}"

if "4b93deeb" in flag:

print(flag)

break

k+=1



（函数脚本来自Lazzaro @ [分类 | Lazzaro](https://lazzzaro.github.io/categories/)）

​