HGAME2023-writeup-Week2 by crumbling

Crypto

RSA大冒险1

challenge1，根据关系式得到q*r的值再分解求得q，随后求解，两个q值都可以得出答案；

challenge2，一次加密输出后仅对q进行便进行第二次加密输出，为rsa的模不互素情况；

challenge3，e=3很小，考虑低加密指数分解攻击；

challenge4，共模攻击。

以下为4题解密代码：

```
from Crypto.Util.number import long_to_bytes,getPrime
from gmpy2 import *
from sympy import *
from random import randint

#challenge1
#p*q*r=206092439343141429470550251539957298073693780430420277630362448059515786104510122064546486779694057
p=233034090030258190050744902053941896701
print(206092439343141429470550251539957298073693780430420277630362448059515786104510122064546486779694057//p)
#884387513073222308467722680786628306570443102374267091512157
q=[885389086482581573264236788611,998868775971320978170449373087]
e=65537
c=0x31606194813744794ed4c863b1c03bb31c91310e11404e329cee79f46c359c21d1453cb685baeafdbd
for i in range(2):
    n = p * q[i]
    r = (p - 1) * (q[i] - 1)
    d = gmpy2.invert(e, r)
    flag = long_to_bytes(pow(c, d, n))
    print(flag)
#m<n_But_also_m<p

#challenge2
n=
[1419977443789128945287691156413504867768817636289570497867266973974864059353463
35523022424109746617126901304932859944046627055313364739901754509131320440711062
64177369265391395943222376939205068794311958102052472433350166677664816086226674
1604355017058980943905514228663098364347973943621573383615058531490473,152096797
87694110238913560969807709193495365230505386744157270667673277742174821475038010
30096286153409620117422682659913305461536377824688504099500678643634651417047854
91031142432238168589562340251150991261003069246122599420749749315547984451236011
596671680593241261297474617090590632363451642933743113082671,1304146184861973049
32776239062654489591722816520480142366690241231091697160315505887309525118887945
90048979882952179025780834404921217104208695815067121055652845011901002880410471
534255535980187914764137439389728337497437052589924364636187434687605440647483967
06858624472318777998966224215240863827569051172283]
e=gmpy2.mpz(65537)
c=0x67860743d4471e29a24887a2c0f23f3a9f44bf6838a87f05d7534a313f47771fe75ce449ec4f
b741affeeff0ac155ae61b2b691c6b1f445c4fefd46c829b502a0dee8e7abaf8e0ffbe7244b5064f
8c96503e45c3d659e45a8597173cae342b7621239fee1920d694aa5ad651c9ad911e7bcf62c40bc3
af384e13f729f6abfbb6
p=gcd(n[0],n[1])
q=n[0]//p
r=gmpy2.mpz((p-1)*(q-1))
d = gmpy2.invert(e, r)
flag = long_to_bytes(pow(c, d, n[0]))
print(flag)
#make_all_modulus_independent

#challenge3
n=731125569477353039130661594079873841365551589558882829118740715645673563122392
21898448524872419513472902921070322015948196922088099316540421125591769639954009
21577768171490640026674231792238976873968326372665430310475753844120805794330449
9970780075171993610038166611084820422422535780004713129332765367378059
e=3
c=0xfec61958cefda3eb5f709faa0282bffaded0a323fe1ef370e05ed3744a2e53b55bdd43e95944
27c35514505f26e4691ba86c6dcff6d29d69110b15b9f84b0d8eb9ea7c03aaf24fa957314b89febf
46a615f81ec031b12fe725f91af9d269873a69748
k=0
c = k * n + c
m = gmpy2.iroot(c, 3)[0]
print(long_to_bytes(m))
#encrypt_exponent_should_be_bigger

#challenge4
n=811203169834403878832267811867729211326724175802851411085742545827869308709363
42881975416292621555290915577900376656192520708398587611048786348944784906032710
27777395514443118530504066038272151904784337098381693112009875896337897906563564
9371936545299452153341771418121344227951468873342338867369764424890119
e=[91841,80953]#互质
```

```
c=
[0x70fd8e8b37aa83508865b2788adddf65b9b299564bea48ad41f9af986ed1eaa829c21ac1442d0
105bbd6c8c4850c126303b214c89a51d6a01611dc59d79d40cb8aa674bdb92a13fdf8dc0e7282482
763ac444f32a9c2570e1b4444cb210b73178cc173f5634e1661e0e39dfdf6547a14ffa8c77c340f5
40692f6e99eb712f7c7,0x23b0c976478c8739c0916f970022abad5eff04136bea7a8790394b1732
57593da79e7be524203cf851bc643cf6b79e38eeada7b479032e0ea0f1456ea509c85ff53214fe6d
70ea12293cbdb489789b6042986db7a8f2db86c73abe240de6a1191ac613e56b134e1c80026fba9a
5fad26d7b25218f8fd17608bf57511366c25ef]
t=gmpy2.gcdext(e[0],e[1])
s=[t[0],t[1],t[2]]
if s[1] < 0:
    s[1] = -s[1]
    c[0] = gmpy2.invert(c[0], n)
elif s[2] < 0:
    s[2] = - s[2]
    c[1] = gmpy2.invert(c[1], n)
m=pow(c[0],s[1],n)*pow(c[1],s[2],n)%n
print(long_to_bytes(m))
#never_uese_same_modulus
```

flag：hgame{W0w_you^knowT^e_CoMm0n&*t$ack*@bout|RSA}

Rabin

考察rsa衍生算法rabin；

思路为：通过同余方程求解得到4个同余式，两两使用中国剩余定理，其中一个为flag。

以下为解密代码：

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

def crt(b,m):
    for i in range(len(m)):
        for j in range(i+1,len(m)):
            if gmpy2.gcd(m[i],m[j]) != 1:
                return -1
    M = 1
    for i in range(len(m)):
        M *= m[i]
    Mm = []
    for i in range(len(m)):
        Mm.append(M // m[i])
    Mm_ = []
    for i in range(len(m)):
        _,a,_ = gmpy2.gcdext(Mm[i],m[i])
        Mm_.append(int(a % m[i]))
    y = 0
    for i in range(len(m)):
        y += (Mm[i] * Mm_[i] * b[i])
    y = y % M
    return y
```

```
p=65428327184555679690730137432886407240184329534772421373193521144693375074983
q=98570810268705084987524975482323456006480531917292601799256241458681800554123
c=0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5ee5
38b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556
d=[(p+1)//4,(q+1)//4]
p=
[65428327184555679690730137432886407240184329534772421373193521144693375074983,9
8570810268705084987524975482323456006480531917292601799256241458681800554123]
x1=pow(c,d[0],p[0])
x2=pow(c,d[1],p[1])
x3=-x1
x4=-x2
x=[x1,x2,x3,x4]
p=[p[0],p[1],p[0],p[1]]
for i in range(4):
    for j in range(i,4):
        t=[x[i],x[j]]
        r=[p[i],p[j]]
        m=crt(t,r)
        if m>0:
            print(long_to_bytes(m))
```

包里有什么

如题名，考察背包问题引出的Merkle–Hellman加密算法

参考了ctfwiki https://ctf-wiki.org/crypto/asymmetric/knapsack/knapsack/

超递增数列ai=2^i,已知条件可计算得出二进制位数l与公钥b。

需要注意的是，加密中明文的高位bit乘的是a的低位，所以二进制形式需要逆序才是明文对应的二进制。

```
from Crypto.Util.number import long_to_bytes
from gmpy2 import *
from libnum import n2s



m = 1528637222531038332958694965114330415773896571891017629493424
for l in range(3,200<<1):
    a = [2 << i for i in range(l)]
    if m>=sum(a) and m<=(2<<l+1):
        print(l)#l=198
        break

l=198
a = [2 << i for i in range(l)]
b0 = 69356606533325456520968776034730214585110536932989313137926
w=(b0//2)%(m//2)
inv_w=gmpy2.invert(w,m)
b = [i *w% m for i in a]
d=[i for i in a]
c = 93602062133487361151420753057739397161734651609786598765462162
flag=(inv_w*c)%m

```

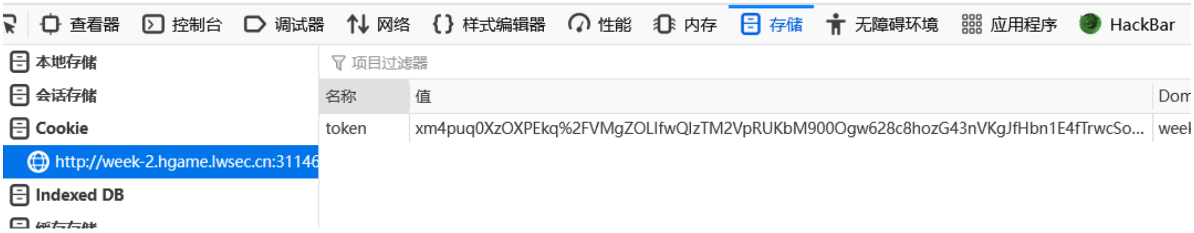
```python
print(bin(flag))#0b11111100001011101001011011111101011101100011101101100111010010110111111010100111001000011001000110111111010100111101100111010000110110011001111101001110110001011001111110101100111011100100000101110100011 0
flag=0b0110001011101000001001110111001101011111100110100011011100101111110011001101100001011100110111110010101111110110001001100001001110010101111110110100101110011011011100011011101011111101101001011101000001111 11

flag=int(flag)
print(long_to_bytes(flag))
```

零元购年货商店

查看附件可知：网页需要用户名为"Vidar-Tu”的用户才能购买flag，但输入"Vidar-Tu”却会被拦住。

再仔细查看router.go文件与util.go文件可以发现：登录网页通过读取用户名的输入（拼接created与uid）先后进行aes-ntr模式加密与base64加密，再设置cookie（如下图）；



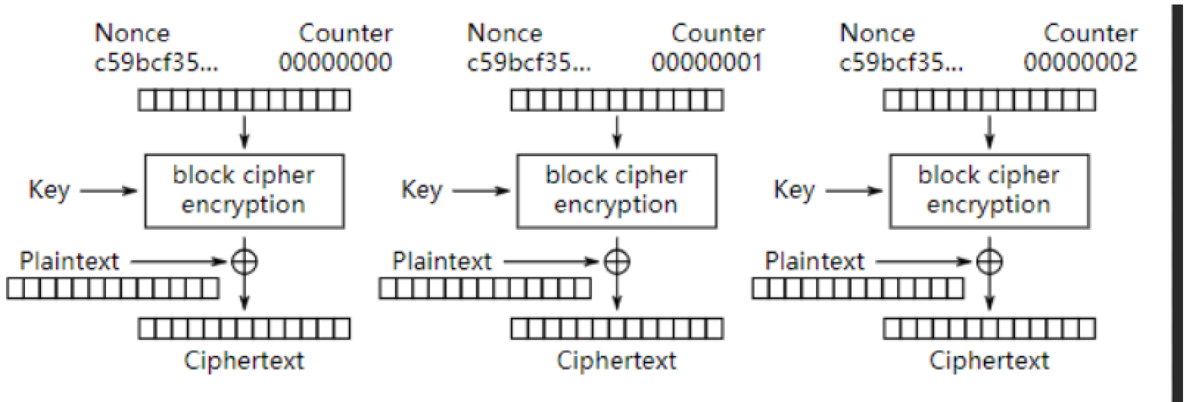
名称	值	Dom
token	xm4puq0XzOXPEkq%2FVMgZOLIfwQIzTM2VpRUKbM900Ogw628c8hozG43nVKgJfHbn1E4fTrwcSo...	weel

而购买页会利用相同的16字节key与16字节iv对token值进行解密，查看用户名是否为“Vidar-Tu”。

因为16字节的key与iv无法获得也难以爆破，提前计算“Vidar-Tu”对应的token值并无法达到要求。

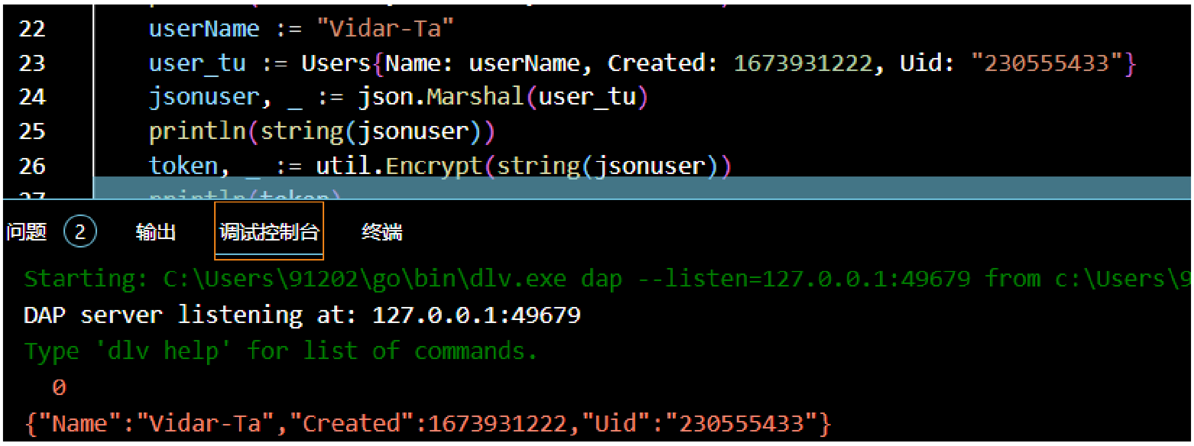
利用相关关键词搜索，能看到cbc模式下的字节翻转攻击与https://soloist777.gitee.io/2019/04/24/AES-CTR%E6%A8%A1%E5%BC%8F%E7%9A%84%E4%B8%A4%E7%A7%8D%E6%94%BB%E5%87%BB%E6%96%B9%E5%BC%8F/文章提到的对aes-ctr模式的明文攻击。

引用一张搜出来的原理图：



Counter (CTR) mode encryption

在key与iv均固定的情况下，aes-ctr的加密中每16字节并不相互影响，并且被加密字符串前16位正好到“Vidar-Tu”的“T”字符



```
    userName := "Vidar-Ta"
    user_tu := Users{Name: userName, Created: 1673931222, Uid: "230555433"}
    jsonuser, _ := json.Marshal(user_tu)
    println(string(jsonuser))
    token, _ := util.Encrypt(string(jsonuser))
    println(token)
```

```
Starting: C:\Users\91202\go\bin\dlv.exe dap --listen=127.0.0.1:49679 from c:\Users\9
DAP server listening at: 127.0.0.1:49679
Type 'dlv help' for list of commands.
  0
{"Name":"Vidar-Ta","Created":1673931222,"Uid":"230555433"}
```

通过这一点，分别输入用户名“Vidar-Ta””aidar-Tu”获得对应token值

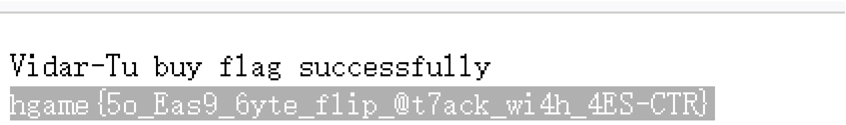
“xm4puq0XzOXPEkq%2FVMgZOKYfwQIzTM2VpRUKbM900Ogw628c8Rc9G43nVKgJfHbn1E4fTrwcSoAQGA%3D%3D”

“xm4puq0XzOXPJUq%2FVMgZOLIfwQIzTM2VpRUKbM900Ogw628c8hozG43nVKgJfHbn1E4fTrwcSoAQGA%3D%3D”

合并，得到“Vidar-Tu”对应token值：

xm4puq0XzOXPEkq%2FVMgZOLIfwQIzTM2VpRUKbM900Ogw628c8hozG43nVKgJfHbn1E4fTrwcSoAQGA%3D%3D

修改token值购买flag：

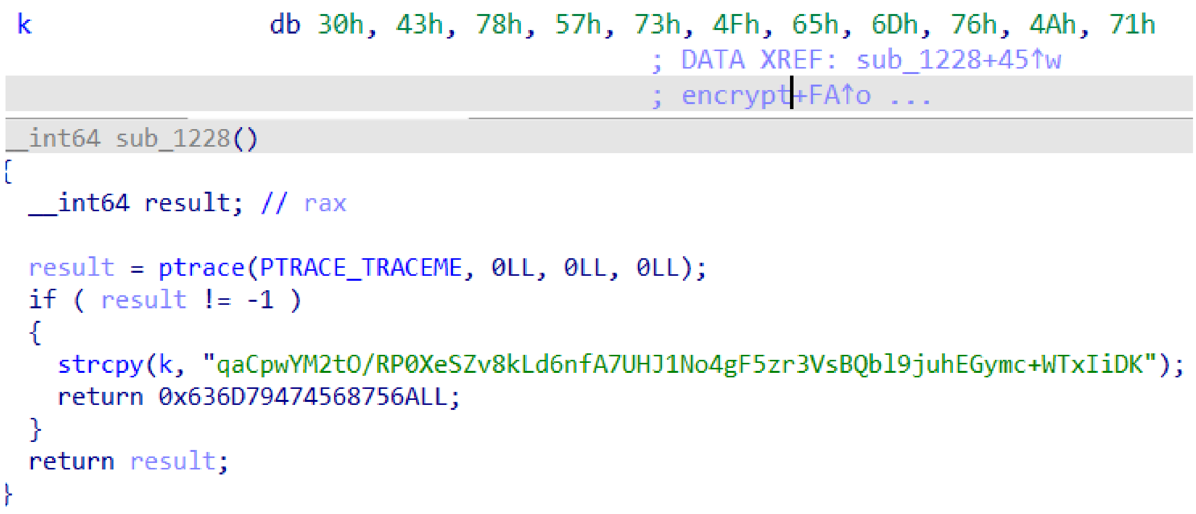


```
Vidar-Tu buy flag successfully
hgame{5o_Eas9_6yte_flip_@t7ack_wi4h_4ES-CTR}
```

Reverse

before_main

base64编码，但是魔改了替换表，并且替换表（数组k）的数据在sub_1228函数（位于start以前）进行过一次修改。



```
k               db 30h, 43h, 78h, 57h, 73h, 4Fh, 65h, 6Dh, 76h, 4Ah, 71h
                                        ; DATA XREF: sub_1228+45↑w
                                        ; encrypt+FA↑o ...
__int64 sub_1228()
{
  __int64 result; // rax

  result = ptrace(PTRACE_TRACEME, 0LL, 0LL, 0LL);
  if ( result != -1 )
  {
    strcpy(k, "qaCpwYM2tO/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK");
    return 0x636D79474568756ALL;
  }
  return result;
}
```

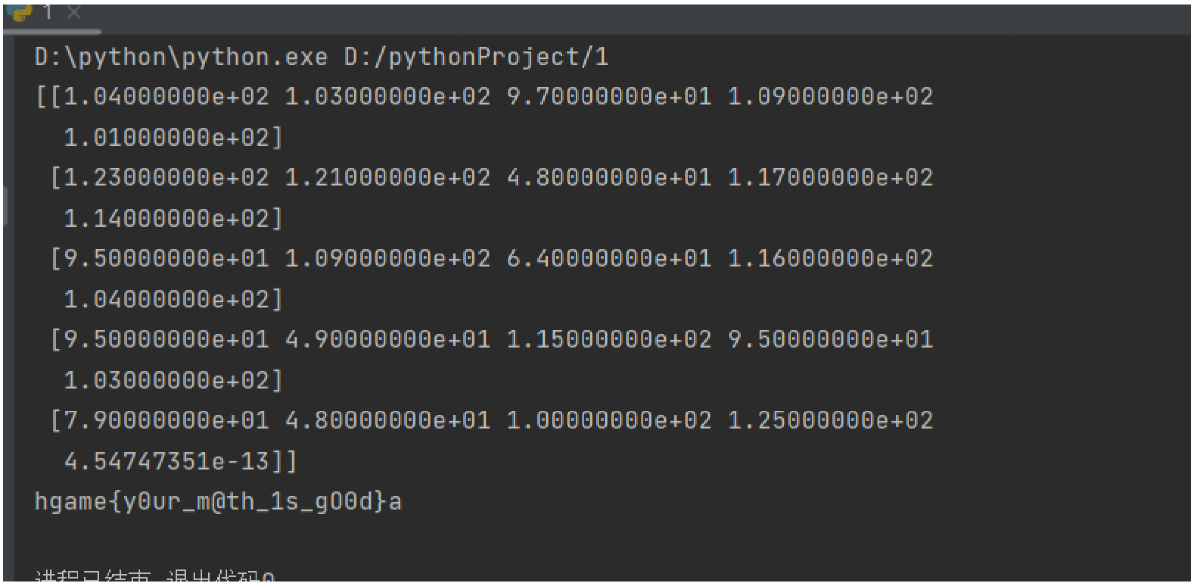
以下为解密代码：

```
#include<stdio.h>
char enc[] = "AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG=";
__int64 v4=44;
unsigned char k[] =
"qaCpwYM2tO/RPOXeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK";
unsigned char ki[256]={0};
int main()
{
    int i;
    for (i = 0; i < 64; i++)
        ki[k[i]] = unsigned char(i);
    int v2 = 0, v3 = 0;
    unsigned char flag[50];
    (unsigned int*)enc;
    while (v2<40)
    {
        flag[v3] = ki[enc[v2]] << 2 | (ki[enc[v2 + 1]] >> 4) & 0x03;
        flag[v3 + 1] = ki[enc[v2 + 1]]<<4 & 0xF0 | ki[enc[v2 + 2]]>>2 & 0x0F;
        flag[v3 + 2] = ki[enc[v2 + 2]]<<6 & 0xC0 | ki[enc[v2 + 3]]&0x3F;
        v3 += 3;
        v2 += 4;
    }
    flag[v3] = ki[enc[v2]] << 2 | (ki[enc[v2 + 1]] >> 4) & 0x03;
    flag[v3 + 1] = ki[enc[v2 + 1]] << 4 & 0xF0 | ki[enc[v2 + 2]] >> 2 & 0x0F;
    flag[v3 + 2] = ki[enc[v2 + 2]] << 6 & 0xC0;
    printf("%s", flag);
    return 0;
}
```

math

2个5阶矩阵乘法的实现，以下为python解题代码：

```
import numpy as np
def matrix2bytes(matrix):
    s=""
    for i in range(5):
        for j in range(5):
            s+=chr(matrix[i][j])
    return s
v7=np.mat([[126,225,62,40,216],[253,20,124,232,122],[62,23,100,161,36],
[118,21,184,26,142],[59,31,186,82,79]])
v9=([[63998,33111,67762,54789,61979],[69619,37190,70162,53110,68678],
[63339,30687,66494,50936,60810],[48784,30188,60104,44599,52265],
[43048,23660,43850,33646,44270]])
A = np.linalg.inv(v7)
v6=np.dot(v9,A)
print(v6)#v6需要调整
v6=[[104,103,97,109,101],[123,121,48,117,114],[95,109,64,116,104],
[95,49,115,95,103],[79,48,100,125,97]]
print(matrix2bytes(v6))
```



```
D:\python\python.exe D:/pythonProject/1
[[1.04000000e+02 1.03000000e+02 9.70000000e+01 1.09000000e+02
  1.01000000e+02]
 [1.23000000e+02 1.21000000e+02 4.80000000e+01 1.17000000e+02
  1.14000000e+02]
 [9.50000000e+01 1.09000000e+02 6.40000000e+01 1.16000000e+02
  1.04000000e+02]
 [9.50000000e+01 4.90000000e+01 1.15000000e+02 9.50000000e+01
  1.03000000e+02]
 [7.90000000e+01 4.80000000e+01 1.00000000e+02 1.25000000e+02
  4.54747351e-13]]
hgame{y0ur_m@th_1s_g00d}a

进程已结束，退出代码0
```

stream

python的逆向。

先用pyinstxtractor.py文件获得stream.exe_extracted文件夹，在文件夹中找到stream.pyc文件，用010editor增加magic number，https://tool.lu/pyc/在线反编译pyc文件得到py文件。

反编译出来的源文件在对齐上有些问题，简单分析后手动调整了对齐。

主要加密为异或后base64加密。

以下为解密代码：

```
import base64

def gen(key):
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
    i = j = 0
    data = []
    for _ in range(50):
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
        data.append(s[(s[i] + s[j]) % 256])
    return data


def decrypt(enc,key):
    flag=''
    for c,k in zip(enc,gen(key)):
        flag += chr(ord(c)^k)
    return flag

key = 'As_we_do_as_you_know'
enc='wr3ClVcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx3DqcKqwr0hw70
1Ly57w63CtcOl'
de_enc=base64.b64decode(enc.encode()).decode()
flag=decrypt(de_enc,key)
print(flag)
```