觅码：

考点：

- python编程
- 编码原理

碎碎念：这题的本意是想让大家熟悉一下python的语法，了解信息是如何编码成为一串数字，并进行一系列数学运算操作的，中文flag属实是有点抽象。这道题应该算是本周的签到题，但是解出的人数不太符合预期，在这里给大伙道歉了。该给出的函数都给出了，那么都配置好环境了，接下来的题目应该会顺利多了。

一些小坑：中文编码，因为c1,c2,c3,c4之间可能会有一些比特是连着的构成中文字符，所以单解出部分密文是无法直接decode的。考虑到这点，每串字符里面都有一些英文字符，告诉做题的师傅解出来的是正确的。

小知识：在python中的几个常用的数字表示类型有：

- 十六进制(0x)开头数字
- 八进制(0o)开头数字
- 二进制(0b)开头数字
- 十进制无特殊开头

在python中，用(b')做开头的表示bytes类型的数据，一个bytes大小为8bit。

几个常用的函数解释：

```
from gmpy2 import iroot #引入函数库，用来开根用的，gmpy2里面有很多比较好用的密码算法可以使用，推荐使用。

long_to_bytes()#将数字类型的数据转换为bytes类型的数据，一个bytes占据8bit大小，那么0-256的一个数字，就可以表示一个bytes，因为在十进制下这个数字不太好直接书写表示，一般我们用十六进制去表示一个bytes，例如ascii码为65的字母'A'，表示为0x41，比特流b'AAA'表示的十六进制数就是0x414141。
bytes_to_long()#为上一个函数的逆操作。

encode()#将字符串编码成bytes数据，即：'0xGame'->b'0xGame'。
decode()#上一个函数的逆操作。

b64encode()#base64编码操作，作用和encode类似，但是由于算法的原因，其可以将不可打印字符编码成打印字符，具体算法流程可以参考“猜谜”，这道题的编码函数。
b64decode()#解base64编码。
```

有了上述这些工具基本就能解题了，题目给了三个数字，一个base64编码后的比特流，直接将相应的数字按照python识别数字的办法还原回去，然后long_to_bytes就完了。（不学会编码，接下来学习密码的路咋走嘛？）

exp:

```
from gmpy2 import iroot
from Crypto.Util.number import *
from base64 import b64decode
c1 =
260707623787245626570139440885928666036832741558210650868364883477202088780135306217121455435174905855360902283398577308320035628453160133922159075621327659089614389495405390297340763821485116417196863060231384402201613542856008184449935667269598175780475659189104923333435206197592402821830900455 1
c2 =
100100001000011011101000101001111010001111100100101110101000011011100100101111111010000111100110100000011010111111100110100110001010111111100101101001101000001011100100101111011001010111100111110111100 1
c3 = b'lueggeeahO+8jOmCo+S5iOW8gOWni+aIkQ=='
c4 = 'e4bbace79a8443727970746fe68c91e68898e590a72121217d'

flag = (long_to_bytes(iroot(c1,5)
[0])+long_to_bytes(eval('0b'+str(c2)))+b64decode(c3)+long_to_bytes(eval('0x'+str(c4)))).deco
de()
print(flag)
#0xGame{ 恭喜你,已经理解了信息是如何编码的，那么开始我们的Crypto挑战吧!!!}

```

RSA：

考点：

- RSA的基本概念
- 欧拉函数定义
- 逆元的运算与定义
- 基本分解模数的工具应用

碎碎念：出的是有点杂，基本入门需要掌握的知识都含括在里面了，对初次接触密码领域的新朋友有点小坑。我们可以通过这题知道：逆元可以代替除法在有限域（某个模数下的式子）中进行运算，同时逆元不是百分百存在的（必须与模数互质）。所以题目中的逆元不能直接求出，这是这题的一个小坑。

虽然这个坑不太符合预期，大伙都能直接除以解出来就是了，就当做下降难度了，，，

思路：RSA概念网上已经很多了，应该也不是特别难懂，比较令新人烦恼的可能是(mod)这个概念，就是取余这个操作，慢慢适应就好了。这道题用yafu.exe，还是用factordb网站，或者是自己随便写写就能分解模数，拿到欧拉函数直接解逆元了，得到flag了。

exp:

```
from Crypto.Util.number import *
n =
930994948999643179920008865859642211363687772193224025580837375468440670742343325642059703001591401117780849161624719938492333583069408682321574475405971
e = 65537
c =
543521224283321457248286747573088275648839740874007204491513488250827374740808497748142930279887847406021483177134027583536530289889606875252116351078011
mask=542575284508859742561171084795791838718957400526601525440498449686212248992471
```

```
fact=[2329990801,2436711469,2732757047,2770441151,2821163021,2864469667,2995527113,3111632101,3162958289,3267547559,3281340371,3479527847,3561068417,3978177241,4134768233,4160088337]
phi = 1
for i in fact:
    phi *= i-1
d = inverse(e,phi)
c =pow(c,d,n)
#这是在mask*m > n情况下的解法，虽然我看大伙直接除以就可以得到原文了，权当一种思路去应对以后见到的情况吧，，
mask_inv=(inverse(mask//GCD(mask,n),n))
c = c*mask_inv%n
m = long_to_bytes(c//GCD(mask,n))
print(m)
#b'0xGame{Magic_M@th_Make_Crypt0}'
```

Take my bag

考点：

- 逆元的运用
- 超递增数列
- 加密算法至数学式子的推导

碎碎念：这题主要是背包密码，这里给出数学公式

$$m = i_n i_{n-1} i_{n-2} \cdot \cdot \cdot \cdot \cdot i_2 i_1 (i_k \in 1, 0)$$

$$\text{加密公式：} \sum 3^{i_n} * w = c(mod n)$$

解密逻辑：w已经给出，那么就很自然的可以考虑到用逆元做除法化简式子得到：

$$c * w^{-1} = \sum 3^{m_{i_n}} * w * w^{-1} = \sum 3^{m_i} (mod n)$$

接下来通过尝试我们可以知道：

$$n > \sum 3^n$$

$$\text{且有} 3^{n+1} > \sum 3^n > 3^{n-1}$$

那么问题就很简单了，可能只需要对贪心算法有一点点了解，就可以直接写脚本了（如果对你来说贪心算法可能一时半会有点难以理解，或是没接触编程，不要紧，权当学习编程思想也是不错的选择。）

exp:

```
from Crypto.Util.number import *

w=16221818045491479713
n=97020742893487631311021743778998839045485841056410451502697635894312938269133486324967751730997769179305172703175867406860085390858989101104428207760010615
c=47959692895723145907874679908652055484301909215567228798917211077192628227894838637423565532499354370043784756616687688934626521037392500387005281115
c = c*inverse(w,n)%n

def decrypt(c):
    index = 0
    m = 0
```

```
    while pow(3,index)<c:
        index+=1
    for i in range(index-1,-1,-1):
        if ((pow(3,i+1)>c)&(pow(3,i)<=c)):
            c -= pow(3,i)
            m += pow(2,i)
    return m

print(long_to_bytes(decrypt(c)))
#b'0xGame{Welc0me_2_Crypt0_G@me!#$&%}'
```

CBC:

考点:

- 对称加密中的分组模式
- 密钥爆破

碎碎念：关于分组模式存在的原因，还望大家自行通过搜索引擎获取，一般的加密算法都是通过了世人的长久考验而留下来的，要想通过分析并攻破是及其困难的事情，但是由于分组模式不同而存在的某些缺陷却是可以利用的，在进行更深一步的探索之前，我想通过基本的概念题，让大家理解这个分组是怎么操作的、并有哪些好处和缺陷。关于解密脚本的编写入门，可能对新人不是那么友好，但是通过观察流程图可以发现，很多操作只要看懂了是可以硬抄的，就没啥太大的难度了，那么这题给出两种解法。

exp1:

可以观察得到，密钥空间并不是很大，可以通过穷举爆破的办法一个个尝试得到，接下来写出基本的CBC解密脚本就可以。

```
from Crypto.Util.number import *

def bytes_xor(a,b):
    a,b=bytes_to_long(a),bytes_to_long(b)
    return long_to_bytes(a^b)

def decrypt(text,key):
    result = b''
    for i in text:
        result += ((i^key)).to_bytes(1,'big')
    return result

def CBCinv(enc,iv,key):
    result = b''
    block=[enc[_*8:(_+1)*8] for _ in range(len(enc)//8)]
    for i in block:
        temp = decrypt(i,key)
        tmp = bytes_xor(iv,temp)
        iv = i
        result += tmp
    return result

iv = b'11111111'
enc = enc =
b"\x8e\xc6\xf9\xdf\xd3\xdb\xc5\x8e8q\x10f>7.5\x81\xcc\xae\x8d\x82\x8f\x92\xd9o'D6h8.d\xd6\x9a\xfc\xdb\xd3\xd1\x97\x96Q\x1d{\\TV\x10\x11"
```

```
for key in range(0xff):
    dec = (CBCinv(enc,iv,key))
    if b'0xGame' in dec:
        print(dec)
#b'0xGame{098f6bcd4621d373cade4e832627b4f6}\x08\x08\x08\x08\x08\x08\x08\x08'
#后面的填充部分就懒得去掉了，，
```

exp2:

因为已知明文、密钥固定的特点，这里利用了CBC分组模式的特点可以直接逆推出密钥，在这里给出这种解法，目的是让新师傅了解一下利用已知明文解密的这种思想。

```
iv = b'11111111'
enc = b"\x8e\xc6\xf9\xdf\xd3\xdb\xc5\x8e8q\x10f>7.5\x81\xcc\xae\x8d\x82\x8f\x92\xd9o'D6h8.d\xd6\x9a\xfc\xdb\xd3\xd1\x97\x96Q\x1d{\\TV\x10\x11"
test = b'0xGame'
key = (iv[0]^test[0]^enc[0])

dec = CBCinv(enc,iv,key)
print(dec)
#b'0xGame{098f6bcd4621d373cade4e832627b4f6}\x08\x08\x08\x08\x08\x08\x08\x08'
```

猜谜：

考点：

- 已知明文攻击
- base64编码算法

碎碎念：考虑到难度，当天还是放出了魔改base64解码函数，通过编码可以将不可打印字符转换成可打印字符(A-Z\a-z\0-9\\+)，以便于在网络传输中显示，随便写的算法就看个乐呵就行。重点是已知明文攻击这部分：如果我们知道了部分明文的情况下，可以通过一定的推导得到部分密钥的信息、甚至是密钥，这在密码学中是一个重要的攻击思想。

在这里我们可以知道，一般的正常加解密算法是难以攻破的，如果我们能在现实中通过侧信道攻击，获取了某些关键的信息呢?

exp:

```
from Crypto.Util.number import *

def dec(text):
    text = text.decode()
    code = 'AP3IXYxn4DmwqOlT0Q/JbKFecN8isvE6gWrto+yf7M5d2pjBuk1Hh9aCRZGUVzLS'
    unpad = 0
    tmp = ''
    if (text[-1] == '=') & (text[-2:] != '=='):
        text = text[:-1]
        unpad = -1
    if text[-2:] == '==':
        text = text[:-2]
```

```
        unpad = -2
    for i in text:
        tmp += str(bin(code.index(i)))[2:].zfill(3)
    tmp = tmp[:unpad]
    result = long_to_bytes(int(tmp,2))
    return result

c = b'IPxYIYPYXPAn3nXX3IXA3YIAPn3xAYnYnPIIPAYYIA3nxxInXAYnIPAIxnXYYYIXIIPAXn3XYXIYAA3AXnx='
enc = dec(c)

mask = b''
kown = b'0xGame{'
for i in range(7):
    mask += (enc[i]^(kown[i]+i)).to_bytes(1,'big')
flag = b''
for i in range(len(enc)):
    flag +=((mask[i%7]^enc[i])-i).to_bytes(1,'big')
print(flag)
#b'0xGame{Kn0wn_pl@intext_Att@ck!}'
```

维吉尼亚密码

这道古典密码题很简单，有不少师傅甚至直接猜都能猜得出密钥是啥（Game），在目前的CTF赛事中古典密码的题已经很少了。这种传统的加密技术中，就算猜不到密钥是啥，通过统计某些密文和密钥的规律基本都能还原信息。WP就不想写了，，

废话：

确实不可否认的是，第一周我弄得题不是很简单，基本都要沾点python编程，对想入门密码、或者是想尝试CTF的哥们不友好。

但是核心思路都非常简单，而且脚本的编写也不会太复杂（要么可以直接抄题目给的部分代码，要么就是想一下就出来了），因为我并不太想看到新人能够在第一周疯狂上分，然后到后面遇到比较复杂的题就开始放弃了，试着适应可能比较好。

秉持这个态度出题（思路唯一），，相信经过第一周的师傅对密码这个方向有一个初步的认识。那么既然坚持下来了，就开始试着去破译一些好玩的算法吧。