2020年国赛密码学&湖湘杯密码学第一题writeup

原创

不会vector 于 2020-11-03 11:16:45 发布 4662 收藏 5

分类专栏： ctf 文章标签： 密码学 加密解密 python

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/qq_44109982/article/details/109463046

版权

ctf 专栏收录该内容

1 篇文章 0 订阅

订阅专栏

国赛（总决赛）题目如下：

```
import random
import os
from Crypto.Cipher import DES

'''
sbox=list(range(256))
random.shuffle(sbox)
print(sbox)
'''
sbox=[217, 25, 248, 51, 6, 37, 32, 69, 96, 117, 86, 108, 155, 50, 27, 249, 171, 82, 168, 245, 228, 195, 118, 187
, 12, 109, 99, 241,
26, 41, 194, 205, 236, 185, 73, 35, 139, 175, 44, 116, 48, 148, 22, 142, 190, 114, 201, 60, 59, 89, 181, 13, 10,
 197, 151, 219, 87, 198, 191, 93, 152, 230, 34, 124, 173, 146, 226, 40, 101, 165, 83, 212, 206, 229, 16, 95, 56,
 107, 144, 250, 149, 153, 208, 133, 145, 156, 172, 162, 113, 100, 183, 127, 57, 223, 130, 72, 135, 84, 147, 15,
242, 177, 42, 23, 122, 54, 0, 131, 166, 244,
88, 231, 94, 235, 29, 125, 7, 232, 110, 90, 179, 75, 220, 71, 251, 45, 243, 207, 58, 120, 184, 9, 39, 218, 63, 3
8, 254, 167, 140, 53, 213, 214, 80, 85, 128, 8, 64, 174, 132, 233, 65, 200, 36, 115, 215, 192, 74, 203, 11, 134,
 104, 170, 160, 227, 169, 246, 141, 221, 52, 21, 20, 47, 199, 33, 216, 182, 188, 143, 138, 102, 202, 105, 79, 49
, 253, 30, 121, 103, 137, 1, 211, 150, 5, 55, 196, 247, 240, 178, 159, 24, 81, 224, 210, 67, 157, 91, 28, 18, 18
6, 126, 237, 98, 252, 2, 239, 43, 180, 209, 238, 225, 68, 119, 66, 46, 61, 17, 31, 112, 62, 158, 189, 234, 193,
111, 204, 19, 77, 222, 164, 14, 97, 3, 70, 255, 78, 129, 163, 92, 76, 4, 123, 176, 106, 161, 154, 136]

def lock(m):
    r=[]
    for i in m:
        r.append(sbox[i])
    return r

def peak(m):
    r=m[-1:]+m[0:-1]
    return r

def rise(m,k):
    assert len(m)==len(k)
    r=[ ]
    for i in range(len(m)):
        r.append(m[i]^k[i])
    return r
```

```
key=os.urandom(8)

secret=[1,2,3,4,5,6,7,8]
r=lock( secret)
r=peak( r )
print( r)

r=rise(r,key)
r=lock ( r)
r=peak ( r)
r=rise (r,key)
print("lockrise:",r)
print("k0 ^ k7:",key[0]^key[-1])

secret=os.urandom(8)
r=lock(secret)
r=peak(r)



r=rise(r,key)
r=lock(r)
r=peak(r)
r=rise(r,key)



print("encrypted:",r)
d=DES.new(secret,DES.MODE_ECB)

#flag=b"flagxxxxx...xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
#print(d.encrypt(flag).hex())

output:
lockrise: [158, 51, 193, 48, 114, 23, 164, 217]
k0 ^ k7: 160
encrypted: [32, 98, 40, 152, 125, 149, 87, 167]
0304eb66ffd8a1e1ee60b7e923c7823da10bb23fba254ab6748137b204133e2a35759572068c3a0c
```

这道题不难,就是两个过程，通过第一个解密钥k,再用密钥k解密文，将密文作为k用DES解密即可

解密钥k，关键点在于peak操作使得k[i]与k[i+1]产生联系，爆破获得k[0] k[7]之后借由这种联系可以很快算出k的其他位

```
import random
import os
from Crypto.Cipher import DES
sbox=[217, 25, 248, 51, 6, 37, 32, 69, 96, 117, 86, 108, 155, 50, 27, 249, 171, 82, 168, 245, 228, 195, 118, 187
, 12, 109, 99, 241,
26, 41, 194, 205, 236, 185, 73, 35, 139, 175, 44, 116, 48, 148, 22, 142, 190, 114, 201, 60, 59, 89, 181, 13, 10,
 197, 151, 219, 87, 198, 191, 93, 152, 230, 34, 124, 173, 146, 226, 40, 101, 165, 83, 212, 206, 229, 16, 95, 56,
 107, 144, 250, 149, 153, 208, 133, 145, 156, 172, 162, 113, 100, 183, 127, 57, 223, 130, 72, 135, 84, 147, 15,
242, 177, 42, 23, 122, 54, 0, 131, 166, 244,
88, 231, 94, 235, 29, 125, 7, 232, 110, 90, 179, 75, 220, 71, 251, 45, 243, 207, 58, 120, 184, 9, 39, 218, 63, 3
8, 254, 167, 140, 53, 213, 214, 80, 85, 128, 8, 64, 174, 132, 233, 65, 200, 36, 115, 215, 192, 74, 203, 11, 134,
 104, 170, 160, 227, 169, 246, 141, 221, 52, 21, 20, 47, 199, 33, 216, 182, 188, 143, 138, 102, 202, 105, 79, 49
, 253, 30, 121, 103, 137, 1, 211, 150, 5, 55, 196, 247, 240, 178, 159, 24, 81, 224, 210, 67, 157, 91, 28, 18, 18
6, 126, 237, 98, 252, 2, 239, 43, 180, 209, 238, 225, 68, 119, 66, 46, 61, 17, 31, 112, 62, 158, 189, 234, 193,
111, 204, 19, 77, 222, 164, 14, 97, 3, 70, 255, 78, 129, 163, 92, 76, 4, 123, 176, 106, 161, 154, 136]
key=[0,0,0,0,0,0,0,0]

r0=[96,25,248,51,6,37,32,69]
r2=[158, 51, 193, 48, 114, 23, 164, 217]

for i in range(256):
    for j in range(256):
            if((i^j==160)&(r2[0]==i^sbox[r0[7]^j])):
              print(i,j)##k[0] k[7]
              key[0]=i
              key[7]=j

for z in [1,2,3,4,5,6]:
          key[z]=r2[z]^sbox[r0[z-1]^key[z-1]]

print(key)
print(key[0]^key[7])
```

输出

```
[0, 180, 224, 60, 139, 193, 154, 160]
```

知道key之后只需逆向运行一遍加密过程即可

```
import random
import os
from Crypto.Cipher import DES
import numpy as np
from Crypto.Util.number import *
sbox=[217, 25, 248, 51, 6, 37, 32, 69, 96, 117, 86, 108, 155, 50, 27, 249, 171, 82, 168, 245, 228, 195, 118, 187
, 12, 109, 99, 241,
26, 41, 194, 205, 236, 185, 73, 35, 139, 175, 44, 116, 48, 148, 22, 142, 190, 114, 201, 60, 59, 89, 181, 13, 10,
 197, 151, 219, 87, 198, 191, 93, 152, 230, 34, 124, 173, 146, 226, 40, 101, 165, 83, 212, 206, 229, 16, 95, 56,
 107, 144, 250, 149, 153, 208, 133, 145, 156, 172, 162, 113, 100, 183, 127, 57, 223, 130, 72, 135, 84, 147, 15,
242, 177, 42, 23, 122, 54, 0, 131, 166, 244,
88, 231, 94, 235, 29, 125, 7, 232, 110, 90, 179, 75, 220, 71, 251, 45, 243, 207, 58, 120, 184, 9, 39, 218, 63, 3
8, 254, 167, 140, 53, 213, 214, 80, 85, 128, 8, 64, 174, 132, 233, 65, 200, 36, 115, 215, 192, 74, 203, 11, 134,
 104, 170, 160, 227, 169, 246, 141, 221, 52, 21, 20, 47, 199, 33, 216, 182, 188, 143, 138, 102, 202, 105, 79, 49
, 253, 30, 121, 103, 137, 1, 211, 150, 5, 55, 196, 247, 240, 178, 159, 24, 81, 224, 210, 67, 157, 91, 28, 18, 18
6, 126, 237, 98, 252, 2, 239, 43, 180, 209, 238, 225, 68, 119, 66, 46, 61, 17, 31, 112, 62, 158, 189, 234, 193,
111, 204, 19, 77, 222, 164, 14, 97, 3, 70, 255, 78, 129, 163, 92, 76, 4, 123, 176, 106, 161, 154, 136]
key=[0, 180, 224, 60, 139, 193, 154, 160]
r=[0,0,0,0,0,0,0,0]
r2=[32, 98, 40, 152, 125, 149, 87, 167]
```

```
dbox=np.empty(shape=(256),dtype=int)
for i in range(256):
    dbox[sbox[i]]=i

def delock(m):
    r=[]
    for i in m:
        r.append(dbox[i])
    return r

def depeak(m):
    r=m[-7:]+m[0:-7]
    return r

def rise(m,k):
    assert len(m)==len(k)
    r=[]
    for i in range(len(m)):
        r.append(m[i]^k[i])
    return r

r2=rise(r2,key)
r2=depeak(r2)
r2=delock(r2)
r2=rise(r2,key)
r2=depeak(r2)
r2=delock(r2)

print(r2)
secret=b''
for i in r2:
    secret+=long_to_bytes(i)


d=DES.new(secret,DES.MODE_ECB)
c=b'\x03\x04\xeb\x66\xff\xd8\xa1\xe1\xee\x60\xb7\xe9\x23\xc7\x82\x3d\xa1\x0b\xb2\x3f\xba\x25\x4a\xb6\x74\x81\x37
\xb2\x04\x13\x3e\x2a\x35\x75\x95\x72\x06\x8c\x3a\x0c'

s=d.decrypt(c)
print(s)
```

flag

```
b'flag{lock_peak_rise_repeat_2333_7481233}'
```

湖湘杯密码学第一题

```
import numpy as np
from pylfsr import LFSR
from Crypto.Util.number import *
import random
import string
from secret import flag

assert flag[:6] == "DASCTF"

def xor(a,b):
    return str(chr(a^b)).encode('latin1')
```

```
def encode(content,key):
    tmp=b""
    for i in range(len(content)):
        tmp += xor(content[i],key[i%len(key)])
    return tmp

def shuffle_str(s):
    str_list = list(s)
    random.shuffle(str_list)
    return ''.join(chr(i) for i in str_list).encode('latin1')


ran_str = ''.join(chr(random.randint(1,256)) for _ in range(512)).encode()
content = ran_str+flag

L4 = LFSR(fpoly=[4,3],initstate ='random',verbose=True)
data = L4.runFullCycle()
k4 = b""
for _ in range(len(data)):
    a = b''
    for _  in range(8):
        a +=  str(L4.next()).encode()
    k4 += long_to_bytes(int(a,2))

k4 = shuffle_str(k4)

enc1=encode(content,k4)

print(enc1)
#b'\xbb\xd3\x08\x15\xc6:\x08\xb2\xb2\x9f\xe4p\xc7\xec\x7f\xfd)\xf6f\x9c\xe4\xd12\xaeJ\x81\xb1\x88\xab\xa5V\xa9\x88\x14\xdf`~\xf6\xdbJ\xb4\x06S!0\xbb\xe4\x1a\xe6R\x8e\x84X\x19K\x95\x07C\xe8\xb2\'\xa9\x80\x15\xec\x8f\x8dY\nK\x85\x99\xb7!\x134\xa9\xb6\x15\xcf&\r\x9b\xe1\x99\xe4]3h~\xf0\xa9\xa5\x14\xee}\xd19l\x14h\x07v *a0\x12\x14\xfe\x0f\x05\xdem\x1d\xe4s2J\x7f\xc28\xf6RR\x8e\xba\xb2m\x18M\xf1\xef!4\x17\xa8\xb4\x14\xc2\x8f\xb9Y:K\xaa\x06T!\x1b\xbb\xfd\xf6Gv\x8e\x9a\xeb\xd9K\xbb\x06N\x9a\x82c\xa9\xa0\x14\xed!\x04\xdbm\x13\xe5w3B\x7f\xd0\xa9\xbf\xb7\x9c\xe3\xd00\x83K\x86\xab3\x7f\xc1\xbb\xfd\x11\x15\xdf\x8e\x80Y\x07\xd8\xe5]2m\xe9\xbb\xce`\x91o\x8f\x8cY!\x81\xe4J\x92\x8c\xa7T\x16E\x15\xf1WMY(\xb8[\x8e2y~\xcbM\x10\x15\xc7\x1fWY\x0cK\x87\xce\xe5 !b\xa8\x83\x14\xec6\xd1!\xc8\x905\xe52L\xf1\xba\xcf\n\x9d\x9d\xe7u\xadm\x06\xe4n2r\xd8\xba\xed\xf6\x7f\x9d\xd8\xd02m\x12G\x07Y\x89\x7f\xc0\xa8\xa4\x15\xe5\x043Y\x1eJ\xae\x07n\x94\x87\xbb\xcf_\x8d\x9d\xd1\x14Y,\x9e\xe5b\xd7\x8c\x7f\xf7\xa8\x8f\x14\xc7\x8f\xb3\xb6\xf1\x93\xe4O\xdd\xc4\xdb\xba\xf6!\x15\xfd.\xd1\x18\xcf\xf6\x03\xea2E\x7f\xe1\xa9\xa5\xfe\x9d\xc9\xd1;\xd9\xee\x05\x06z\xc8\xb2\xbb\xe2\xf7{JW4\xcdm\x1a\xe5U\x8d \x0f&\x14\x7f\xf6\x9d\xd4E\xbf\xc3\xdb\xe4L\xe1\xf7\x90\xbb\xdaZ\xf4\x9d\xd13\xb8m3\xe2D3o~\xf8H\xf6U*\x07lY\x03K\xab\x07~\xa3\x87\xbb\xc9\xf7sAQ\x08Y6J\x86\x07Y\xec\xf7\xbb\xc6s\x15\xc6\x7fEY\x02J\x95\x07Z \x11\xbb\xc6T\x15\xfc-\xd0\x06\xe6\x9f-\x07^ \x15\xbb\xccz\x14\xf3\x8f\x97\xd4l9t\x85\xe8\x8a\xbe\xbb\xf9\xf6f\x9d\xf2\xd19\xa2K\xb6\xcd\xcf\xf6~\xd5\xa9\xaa\x15\xd8\x8e\xb3\x81m9\xe4f\xb2!\x1e\xba\xd8s\xfd\x11\x08W\xa1L;\x01\x07_!\x11\xbb\xdd\xf6x\x9d\xf0\x17Y\x15\xfe\x02\xc7\xa0!.W\xa9\xa5\x8f\x9c\xe8\xd1\x12m\x04\xe5s3Q~\xdd\xa9\xa3\x15\xdb\x8f\xac\xaf\xec\xbb\x10\xde2_\xba\xba\xe8\xf6f.\x1e\xd1\x17l\x06\xe4U\xdd\xf0\xd6~\x0fA\x14\xcb\x8e\xb0Y\x1fJ\xb2\xe4\xb3!"\xba\xfeU\x14\xedY\xd0>l-~\x06P 1\xbb\xf2\xf6waD\xd1(m\x12`\x06@\xb6~\xfa\xa9\xb1\xb0\x9d\xfb\x18\xfbm&\xe4v2w\xce\xba\xcbo\xd5\x07\x11QX<J\xbd\xb22O\x7f\xd8x>\xc8\x9c\xd3\xd03\x9d\xb5\x1e\xd72S\xf2ry\xf1W\x9c\xc89Y\rK\x8f\xff\x8a\xe0\xb5{\xa9\xae\xb1\x9d\xdd\xd1=\xbeK\xa3\x06e!\x08\xba\xd2\xf6j\x9c\xf6\xd0\x0fl#\xe5o\xf5\xaa~\xc2\xa9\x99\x15\xea6\xd1:\xe7\xa8\xe4n\xbb \nV\xa9\x91\x14\xf9}\xd0!m/\xe5|2o\x81\xba\xf8\r\x14\xeb\tR\xc9\xec\xdd`\xbf\xc6\x81\xdfKXW\xb3o.%\xa9\xcd\xb9\x14\xfd\x97\x83\x8eO\n\x03\xb6iuu\xab\x9d\xbc\x15\xf4\xc3\xd6\xc1'
```

这道题的解题关键就是LFSR生成密钥时，用的初始数据位数太少，k4仅有4位，16种可能性，而且密文M中必定含有DASCTF字符串，即k xor C（明文）='DASCTF'，所以我们可以用k=C^'DASCTF'(已知明文攻击）
求出k的某六位，进而得知k是16种密钥里的哪一种。
有个问题是不知道DASCTF在哪个位置，所以要遍历一下（大概位置是后100位，毕竟flag也不会太长）

```
import numpy as np
```

```
from pylfsr import LFSR
from Crypto.Util.number import *
import random
import string

def xor(a,b):
    return str(chr(a^b)).encode('latin1')


def encode(content,key):
    tmp=b""
    for i in range(len(content)):
        tmp += xor(content[i],key[i%len(key)])
    return tmp

def increase(a,i):
    if(a[i]==0):a[i]=1
    else:
        a[i]=0
        increase(a,i-1)

M=b'\xbb\xd3\x08\x15\xc6:\x08\xb2\xb2\x9f\xe4p\xc7\xec\x7f\xfd)\xf6f\x9c\xe4\xd12\xaeJ\x81\xb1\x88\xab\xa5V\xa9\
x88\x14\xdf`~\xf6\xdbJ\xb4\x06S!0\xbb\xe4\x1a\xe6R\x8e\x84X\x19K\x95\x07C\xe8\xb2\'\xa9\x80\x15\xec\x8f\x8dY\nK\
x85\x99\xb7!\x134\xa9\xb6\x15\xcf&\r\x9b\xe1\x99\xe4]3h~\xf0\xa9\xa5\x14\xee}\xd19l\x14h\x07v *a0\x12\x14\xfe\x0
f\x05\xdem\x1d\xe4s2J\x7f\xc28\xf6RR\x8e\xba\xb2m\x18M\xf1\xef!4\x17\xa8\xb4\x14\xc2\x8f\xb9Y:K\xaa\x06T!\x1b\xb
b\xfd\xf6Gv\x8e\x9a\xeb\xd9K\xbb\x06N\x9a\x82c\xa9\xa0\x14\xed!\x04\xdbm\x13\xe5w3B\x7f\xd0\xa9\xbf\xb7\x9c\xe3\
xd00\x83K\x86\xab3\x7f\xc1\xbb\xfd\x11\x15\xdf\x8e\x80Y\x07\xd8\xe5]2m\xe9\xbb\xce`\x91o\x8f\x8cY!\x81\xe4J\x92\
x8c\xa7T\x16E\x15\xf1WMY(\xb8[\x8e2y~\xcbM\x10\x15\xc7\x1fWY\x0cK\x87\xce\xe5 !b\xa8\x83\x14\xec6\xd1!\xc8\x905\
xe52L\xf1\xba\xcf\n\x9d\x9d\xe7u\xadm\x06\xe4n2r\xd8\xba\xed\xf6\x7f\x9d\xd8\xd02m\x12G\x07Y\x89\x7f\xc0\xa8\xa4
\x15\xe5\x043Y\x1eJ\xae\x07n\x94\x87\xbb\xcf_\x8d\x9d\xd1\x14Y,\x9e\xe5b\xd7\x8c\x7f\xf7\xa8\x8f\x14\xc7\x8f\xb3
\xb6\xf1\x93\xe4O\xdd\xc4\xdb\xba\xf6!\x15\xfd.\xd1\x18\xcf\xf6\x03\xea2E\x7f\xe1\xa9\xa5\xfe\x9d\xc9\xd1;\xd9\x
ee\x05\x06z\xc8\xb2\xbb\xe2\xf7{JW4\xcdm\x1a\xe5U\x8d \x0f&\x14\x7f\xf6\x9d\xd4E\xbf\xc3\xdb\xe4L\xe1\xf7\x90\xb
b\xdaZ\xf4\x9d\xd13\xb8m3\xe2D3o~\xf8H\xf6U*\x07lY\x03K\xab\x07~\xa3\x87\xbb\xc9\xf7sAQ\x08Y6J\x86\x07Y\xec\xf7\
xbb\xc6s\x15\xc6\x7fEY\x02J\x95\x07Z \x11\xbb\xc6T\x15\xfc-\xd0\x06\xe6\x9f-\x07^ \x15\xbb\xccz\x14\xf3\x8f\x97\
xd4l9t\x85\xe8\x8a\xbe\xbb\xf9\xf6f\x9d\xf2\xd19\xa2K\xb6\xcd\xcf\xf6~\xd5\xa9\xaa\x15\xd8\x8e\xb3\x81m9\xe4f\xb
2!\x1e\xba\xd8s\xfd\x11\x08W\xa1l;\x01\x07_!\x11\xbb\xdd\xf6x\x9d\xf0\x17Y\x15\xfe\x02\xc7\xa0!.W\xa9\xa5\x8f\x9
c\xe8\xd1\x12m\x04\xe5s3Q~\xdd\xa9\xa3\x15\xdb\x8f\xac\xaf\xec\xbb\x10\xde2_\xba\xba\xe8\xf6f.\x1e\xd1\x17l\x06\
xe4U\xdd\xf0\xd6~\x0fA\x14\xcb\x8e\xb0Y\x1fJ\xb2\xe4\xb3!"\xba\xfeU\x14\xedY\xd0>l-~\x06P 1\xbb\xf2\xf6waD\xd1(m
\x12`\x06@\xb6~\xfa\xa9\xb1\xb0\x9d\xfb\x18\xfbm&\xe4v2w\xce\xba\xcbo\xd5\x07\x11QX<J\xbd\xb22O\x7f\xd8x>\xc8\x9
c\xd3\xd03\x9d\xb5\x1e\xd72S\xf2ry\xf1W\x9c\xc89Y\rK\x8f\xff\x8a\xe0\xb5{\xa9\xae\xb1\x9d\xdd\xd1=\xbeK\xa3\x06e
!\x08\xba\xd2\xf6j\x9c\xf6\xd0\x0fl#\xe5o\xf5\xaa~\xc2\xa9\x99\x15\xea6\xd1:\xe7\xa8\xe4n\xbb \nV\xa9\x91\x14\xf
9}\xd0!m/\xe5|2o\x81\xba\xf8\r\x14\xeb\tR\xc9\xec\xdd`\xbf\xc6\x81\xdfKXW\xb3o.%\xa9\xcd\xb9\x14\xfd\x97\x83\x8e
O\n\x03\xb6iuu\xab\x9d\xbc\x15\xf4\xc3\xd6\xc1'
s=[0,0,0,0]
c=b'DASCTF'
for i in range(60):
   for j in range(16):
     L4 = LFSR(fpoly=[4,3],initstate=s,verbose=False)
     data = L4.runFullCycle()
     k4 = b""
     for _ in range(len(data)):
       a = b''
       for _  in range(8):
          a +=  str(L4.next()).encode()
       k4 += long_to_bytes(int(a,2))
     increase(s,3)
     sum=0
     p=(encode(M[750+i:805],c))
     for j in range(6):
        for k in range(15):
```

```
            if(p[j]==k4[k]):
                sum=sum+1
    if(sum==6):
        print(750+i)
        print(k4)
```

最后因为k是打乱顺序的，所以我们只知道前六位（DASCTF），第7位（{），第39位（}）也即密钥的第九位
所以剩下七位要用全排列遍历这些可能性，然后筛选一下结果，符合条件的输出

```
import itertools
import numpy as np
from pylfsr import LFSR
from Crypto.Util.number import *
import random
import string
def xor(a,b):
    return str(chr(a^b)).encode('latin1')

def decode(M,key):
    tmp=b""
   # M=M.decode('Latin1')
    for i in range(len(M)):
        tmp += xor(M[i],key[i%len(key)])      ##向量按位异或
    return tmp

m=b'\xbb\xd3\x08\x15\xc6:\x08\xb2\xb2\x9f\xe4p\xc7\xec\x7f\xfd)\xf6f\x9c\xe4\xd12\xaeJ\x81\xb1\x88\xab\xa5V\xa9\
x88\x14\xdf`~\xf6\xdbJ\xb4\x06S!0\xbb\xe4\x1a\xe6R\x8e\x84X\x19K\x95\x07C\xe8\xb2\'\xa9\x80\x15\xec\x8f\x8dY\nK\
x85\x99\xb7!\x134\xa9\xb6\x15\xcf&\r\x9b\xe1\x99\xe4]3h~\xf0\xa9\xa5\x14\xee}\xd19l\x14h\x07v *a0\x12\x14\xfe\x0
f\x05\xdem\x1d\xe4s2J\x7f\xc28\xf6RR\x8e\xba\xb2m\x18M\xf1\xef!4\x17\xa8\xb4\x14\xc2\x8f\xb9Y:K\xaa\x06T!\x1b\xb
b\xfd\xf6Gv\x8e\x9a\xeb\xd9K\xbb\x06N\x9a\x82c\xa9\xa0\x14\xed!\x04\xdbm\x13\xe5w3B\x7f\xd0\xa9\xbf\xb7\x9c\xe3\
xd00\x83K\x86\xab3\x7f\xc1\xbb\xfd\x11\x15\xdf\x8e\x80Y\x07\xd8\xe5]2m\xe9\xbb\xce`\x91o\x8f\x8cY!\x81\xe4J\x92\
x8c\xa7T\x16E\x15\xf1WMY(\xb8[\x8e2y~\xcbM\x10\x15\xc7\x1fWY\x0cK\x87\xce\xe5 !b\xa8\x83\x14\xec6\xd1!\xc8\x905\
xe52L\xf1\xba\xcf\n\x9d\x9d\xe7u\xadm\x06\xe4n2r\xd8\xba\xed\xf6\x7f\x9d\xd8\xd02m\x12G\x07Y\x89\x7f\xc0\xa8\xa4
\x15\xe5\x043Y\x1eJ\xae\x07n\x94\x87\xbb\xcf_\x8d\x9d\xd1\x14Y,\x9e\xe5b\xd7\x8c\x7f\xf7\xa8\x8f\x14\xc7\x8f\xb3
\xb6\xf1\x93\xe4O\xdd\xc4\xdb\xba\xf6!\x15\xfd.\xd1\x18\xcf\xf6\x03\xea2E\x7f\xe1\xa9\xa5\xfe\x9d\xc9\xd1;\xd9\x
ee\x05\x06z\xc8\xb2\xbb\xe2\xf7{JW4\xcdm\x1a\xe5U\x8d \x0f&\x14\x7f\xf6\x9d\xd4E\xbf\xc3\xdb\xe4L\xe1\xf7\x90\xb
b\xdaZ\xf4\x9d\xd13\xb8m3\xe2D3o~\xf8H\xf6U*\x07lY\x03K\xab\x07~\xa3\x87\xbb\xc9\xf7sAQ\x08Y6J\x86\x07Y\xec\xf7\
xbb\xc6s\x15\xc6\x7fEY\x02J\x95\x07Z \x11\xbb\xc6T\x15\xfc-\xd0\x06\xe6\x9f-\x07^ \x15\xbb\xccz\x14\xf3\x8f\x97\
xd4l9t\x85\xe8\x8a\xbe\xbb\xf9\xf6f\x9d\xf2\xd19\xa2K\xb6\xcd\xcf\xf6~\xd5\xa9\xaa\x15\xd8\x8e\xb3\x81m9\xe4f\xb
2!\x1e\xba\xd8s\xfd\x11\x08W\xa1l;\x01\x07_!\x11\xbb\xdd\xf6x\x9d\xf0\x17Y\x15\xfe\x02\xc7\xa0!.W\xa9\xa5\x8f\x9
c\xe8\xd1\x12m\x04\xe5s3Q~\xdd\xa9\xa3\x15\xdb\x8f\xac\xaf\xec\xbb\x10\xde2_\xba\xba\xe8\xf6f.\x1e\xd1\x17l\x06\
xe4U\xdd\xf0\xd6~\x0fA\x14\xcb\x8e\xb0Y\x1fJ\xb2\xe4\xb3!"\xba\xfeU\x14\xedY\xd0>l-~\x06P 1\xbb\xf2\xf6waD\xd1(m
\x12`\x06@\xb6~\xfa\xa9\xb1\xb0\x9d\xfb\x18\xfbm&\xe4v2w\xce\xba\xcbo\xd5\x07\x11QX<J\xbd\xb22O\x7f\xd8x>\xc8\x9
c\xd3\xd03\x9d\xb5\x1e\xd72S\xf2ry\xf1W\x9c\xc89Y\rK\x8f\xff\x8a\xe0\xb5{\xa9\xae\xb1\x9d\xdd\xd1=\xbeK\xa3\x06e
!\x08\xba\xd2\xf6j\x9c\xf6\xd0\x0fl#\xe5o\xf5\xaa~\xc2\xa9\x99\x15\xea6\xd1:\xe7\xa8\xe4n\xbb \nV\xa9\x91\x14\xf
9}\xd0!m/\xe5|2o\x81\xba\xf8\r\x14\xeb\tR\xc9\xec\xdd`\xbf\xc6\x81\xdfKXW\xb3o.%\xa9\xcd\xb9\x14\xfd\x97\x83\x8e
O\n\x03\xb6iuu\xab\x9d\xbc\x15\xf4\xc3\xd6\xc1'
k2=b'\xd7\x89\xaf\x13^&\xbcMx\x9a\xf15\xe2k\xc4'
for i in k2:
    print(i)
print("----------")
k3=b''
k3+=long_to_bytes(77)
k3+=long_to_bytes(19)
k3+=long_to_bytes(154)
k3+=long_to_bytes(175)
k3+=long_to_bytes(137)
k3+=long_to_bytes(38)
```

```
k3+=long_to_bytes(196)
k4=k3
a=[215,94,120,241,188,53,226,107]

p=list(itertools.permutations(a,8))
for i in p:
    for k in range(8):
        k3+=long_to_bytes(i[k])
    flag=decode(m[770:],k3)
    if((flag[7]>0x1f)&(flag[7]<0x7f)&(flag[8]>0x1f)&(flag[8]<0x7f)&(flag[10]>0x1f)&(flag[10]<0x7f)&(flag[11]>0x1
f)&(flag[11]<0x7f)&(flag[12]>0x1f)&(flag[12]<0x7f)&(flag[13]>0x1f)&(flag[13]<0x7f)&(flag[14]>0x1f)&(flag[14]<0x7
f)):
      print(flag)
    k3=k4
```

flag: