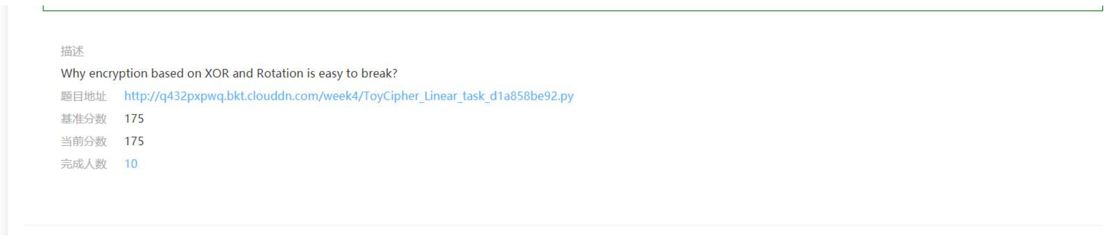
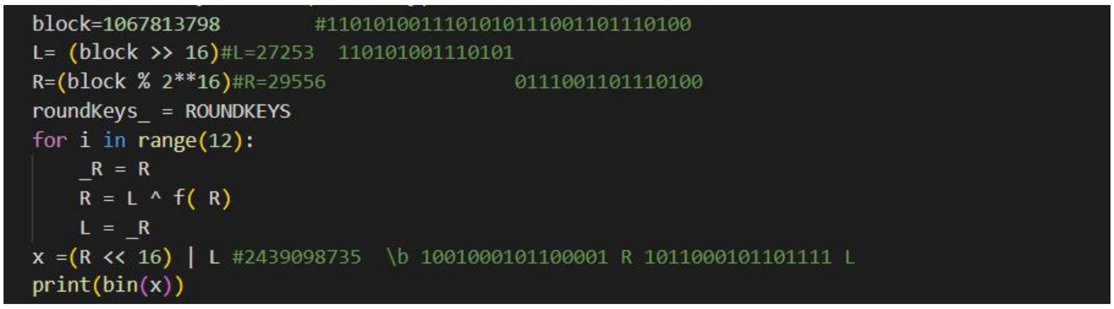
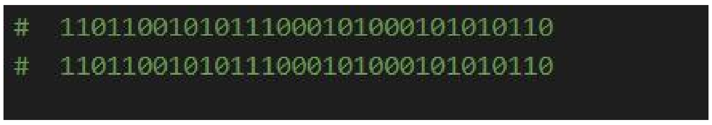

描述
Why encryption based on XOR and Rotation is easy to break?
题目地址 http://q432pxpwq.bkt.clouddn.com/week4/ToyCipher_Linear_task_d1a858be92.py
基准分数 175
当前分数 175
完成人数 10

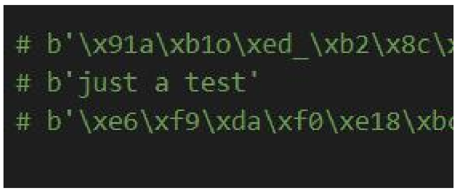
想了好久好久。想了好久好久 在最后一天早上睡觉的时候突然想到了自己一个忽视的点
既然我异或是可以换位拆分，然后 feistel 密码是可逆的然后根据他这个循环我想到我可以把密钥单独放在一边考虑明文加密放在一边
单独考虑加密把　(明文^密钥)加密==明文加密^密钥加密


```
block=1067813798        #11010100111010101110011011110100
L= (block >> 16)#L=27253  110101001110101
R=(block % 2**16)#R=29556             0111001101110100
roundKeys_ = ROUNDKEYS
for i in range(12):
    _R = R
    R = L ^ f( R)
    L = _R
x =(R << 16) | L #2439098735  \b 1001000101100001 R 1011000101101111 L
print(bin(x))
```

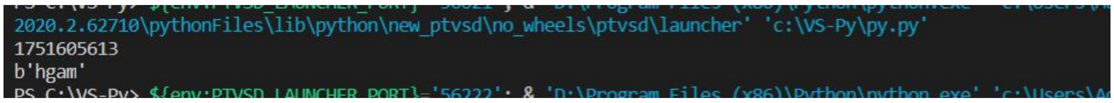
通过 f 中间的密钥删除我可以求出来明文加密
然后根据答案的（明文^密钥）加密可以异或出来密钥加密


```
#  11011001010111000101000101010110
#  11011001010111000101000101010110
```

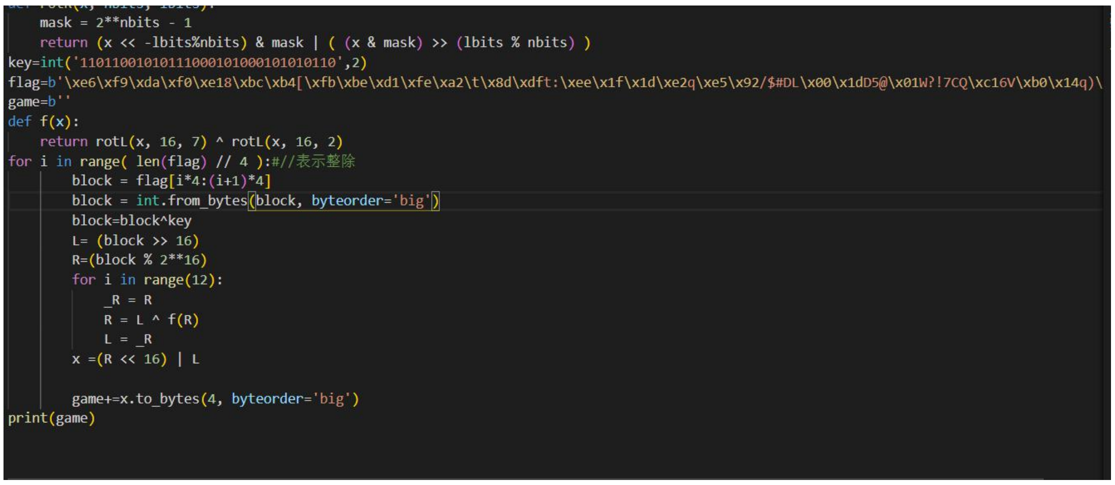
根据 just　a t　est\x01
三次的比较发现我的思路可行密钥求出来这个确实是密钥加密
然后根据思路把


```
# b'\x91a\xb1o\xed_\xb2\x8c\
# b'just a test'
# b'\xe6\xf9\xda\xf0\xe18\xb
```

前四个代入！！！


```
2020.2.62710\pythonFiles\lib\python\new_ptvsd\no_wheels\ptvsd\launcher' 'c:\VS-Py\py.py'
1751605613
b'hgam'
PS C:\VS-Py> ${env:PTVSD_LAUNCHER_PORT}='56222'; & 'D:\Program Files (x86)\Python\python.exe' 'c:\Users\A
```

Hgam！！！
然后确实可行！！然后就是写代码的过程啦



```
    mask = 2**nbits - 1
    return (x << -lbits%nbits) & mask | ( (x & mask) >> (lbits % nbits) )
key=int('11011001010111000101000101010110',2)
flag=b'\xe6\xf9\xda\xf0\xe18\xbc\xb4[\xfb\xbe\xd1\xfe\xa2\t\x8d\xdft:\xee\x1f\x1d\xe2q\xe5\x92/$#DL\x00\x1dD5@\x01W?!7CQ\xc16V\xb0\x14q)\
game=b''
def f(x):
    return rotL(x, 16, 7) ^ rotL(x, 16, 2)
for i in range( len(flag) // 4 ):#//表示整除
        block = flag[i*4:(i+1)*4]
        block = int.from_bytes(block, byteorder='big')
        block=block^key
        L= (block >> 16)
        R=(block % 2**16)
        for i in range(12):
            _R = R
            R = L ^ f(R)
            L = _R
        x =(R << 16) | L

        game+=x.to_bytes(4, byteorder='big')
print(game)
```

```
import os, binascii
def rotL(x, nbits, lbits):                              #
    mask = 2**nbits - 1                                 #2 的 n 次减一   即 n 个 1 二进制
    return (x << lbits%nbits) & mask | ( (x & mask) >> (-lbits % nbits) )    #&与 |或


def rotR(x, nbits, lbits):
    mask = 2**nbits - 1
    return (x << -lbits%nbits) & mask | ( (x & mask) >> (lbits % nbits) )
key=int('11011001010111000101000101010110',2)
flag=b'\xe6\xf9\xda\xf0\xe18\xbc\xb4[\xfb\xbe\xd1\xfe\xa2\t\x8d\xdft:\xee\x1f\x1d\xe2q\xe5\x92
/$#DL\x00\x1dD5@\x01W?!7CQ\xc16V\xb0\x14q)\xaa2'
game=b''
def f(x):
    return rotL(x, 16, 7) ^ rotL(x, 16, 2)
for i in range( len(flag) // 4 ):#//表示整除
        block = flag[i*4:(i+1)*4]
        block = int.from_bytes(block, byteorder='big')
        block=block^key
        L= (block >> 16)
        R=(block % 2**16)
        for i in range(12):
            _R = R
            R = L ^ f(R)
            L = _R
        x =(R << 16) | L

        game+=x.to_bytes(4, byteorder='big')
print(game)
```