Hgame Week2 WP

Crypto

notRC4

萌新瑟瑟发抖......先 rename 一下毒瘤变量名

```
class notRC4_class:
    def __init__(self):
        self.sBox = [0] * 256
        self.number_i = 0
        self.number_j = 0
        self.key_list_r32 = [0] * 256
        for i in range(256):
            self.sBox[i] = i

    def init_notRC4(self, key_list):
        l = len(key_list)  # 8
        for i in range(256):
            self.key_list_r32[i] = key_list[i % l]  # 32 个arg_list
        for i in range(256):  # 依据key 打乱S 盒
            self.number_j = (self.number_j + self.sBox[i] + self.key_list_r32[i]) % 256
            self.sBox[i], self.sBox[self.number_j] = self.sBox[self.number_j], \
                                                     self.sBox[i]
        self.number_i = self.number_j = 0

    def notRC4encode(self, length):
        Output = []
        for _ in range(length):
            self.number_i = (self.number_i + 1) % 256
            self.number_j = (self.number_j + self.sBox[self.number_i]) % 256
            self.sBox[self.number_i], self.sBox[self.number_j] = self.sBox[
                                                                     self.number_j], \
                                                                 self.sBox[
                                                                     self.number_i]
            t = (self.sBox[self.number_i] + self.sBox[self.number_j]) % 256
            Output.append(self.sBox[t])
        print(self.sBox)
        return Output
```

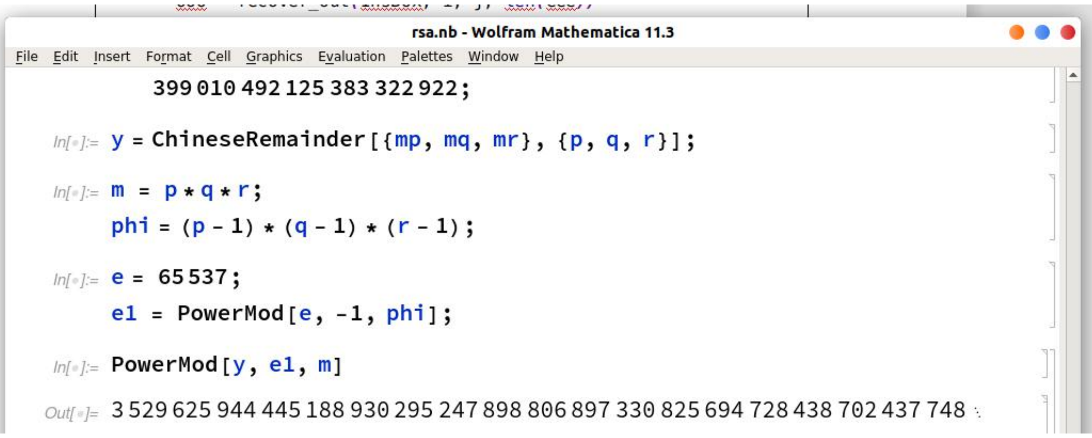
问题的核心是恢复 S 盒和 Out，爆破最终指标 i 和 j 即可

```python
def recover_out(sBox, number_i, number_j, length):
    Output = []
    for _ in range(length):
        t = (sBox[number_i] + sBox[number_j]) % 256
        Output.append(sBox[t])
        sBox[number_i], sBox[number_j] = sBox[
                                            number_j], \
                                         sBox[
                                            number_i]
        number_j = (number_j - sBox[number_i]) % 256
        number_i = (number_i - 1) % 256
    Output.reverse()
    return Output


for i in range(256):
    for j in range(256):
        InsBox = sBox.copy()
        ooo = recover_out(InsBox, i, j, len(eee))
        t = xor(eee, ooo)
        if b'hgame' in t:
            print(t)
```

Remainder

利用中国剩余定理求解三个线性同余方程联立成的方程组，考虑其通解形式，问题归结于模 pqr 最小非负完全剩余系上的离散对数问题。直接开高次模根计算上是不可行的，这是 RSA 加密算法的安全保障。依据欧拉定理，求幂指数模 EularPhi(pqr)的乘法逆元即可。



Inv

这题一开始看见 sBox 以为是对称加密，后来发现好像不是，实在是不知道和什么密码有关系，当成代数题来做了……注意到 S 在运算 Mul 操作下总是$S_e = (0,1,2,3...255)$经过某个置换得到的，我们猜测 S 和定义在 S 上的运算 Mul 构成有限群 S-Mul,这里给出一个不严格的说明：

存在唯一单位元$S_e = (0,1,2,3...255)$有

$$\forall s\ Mul(s, S_e) = Mul(S_e, s) = s$$

容易知道 Mul 运算满足结合律

$$Mul(Mul(a,b),c) = Mul(a, Mul(b,c))$$

可以用爆破的办法求逆元，且逆元必定唯一

```
def cul_inv(e):
    e_1 = []
    for i in range(256):
        for j in range(256):
            if e[j] == i:
                e_1.append(j)
    return e_1
```

题中所给数据相当于给出了两个 sBox 的高次幂s^{739} s^{595}利用这两个值和他们的逆元进行变换即可得到初始的 s，进而计算 flag

```
e_739_1 = cul_inv(e_739)
e_595_1 = cul_inv(e_595)
e_144 = Mul(e_739,e_595_1)
e_144_1 = cul_inv(e_144)
e_19 = Mul(e_595,e_144_1)
for _ in range(3):
    e_19 = Mul(e_19,e_144_1)

e_19_1 = cul_inv(e_19)
e_11 = Mul(e_144,e_19_1)
for _ in range(6):
    e_11 = Mul(e_11,e_19_1)

e_11_1 = cul_inv(e_11)
sBox0 = Mul(e_144,e_11_1)
for _ in range(12):
    sBox0 = Mul(sBox0,e_11_1)

print(sBox0)
```

```python
flagList = []

for i in range(len(list(e_flag))):
    for t in range(256):
        if list(sBox0)[t] == list(e_flag)[i]:
            flagList.append(t)
print(flagList)
print(bytes(flagList))
```