Week_1 做题记录

- **Crypto**
  - Multi Prime RSA
  - RSA Attack 3

Multi Prime RSA

- **思路**

根据题目多素数的rsa可以参考这篇文章

> CTF中的RSA算法

参考中国剩余定理

在文章中的实例代码可以看到给出的是三素数的加密及解密过程，用p1，p2相乘得到p4来与p3做一组

在HGAME中output是四位素数可以分出两个组（解出来的两个m是一样的）

- **代码**

文章实例代码

```
import gmpy2
n=2279269
p1=137
p2=131
p3=127
e=19
c=768924
dp1=gmpy2.invert(e,p1-1)
dp2=gmpy2.invert(e,p2-1)
dp3=gmpy2.invert(e,p3-1)
m1=pow(c,dp1,p1)
m2=pow(c,dp2,p2)
m3=pow(c,dp3,p3)

qInv1=gmpy2.invert(p2,p1)
h1=(qInv1*((m1-m2) % p1)) % p1
m4=m2+h1*p2
#构造了一个p4来带入m4解出最后的m
p4=p1*p2

qInv2=gmpy2.invert(p3,p4)
h2=(qInv2*((m4-m3)% p4)) % p4
m=m3+h2*p3
print(m)
```

修改后的代码

```
"""
user:鸢柒
purpose:Multi Prime RSA
time:2022.02.08__16:53
"""
import gmpy2
from libnum import n2s
n=...
p1=...
p2=...
p3=...
p4=...
e=65537
c=...

dp1=gmpy2.invert(e,p1-1)
dp2=gmpy2.invert(e,p2-1)
dp3=gmpy2.invert(e,p3-1)
dp4=gmpy2.invert(e,p4-1)

m1=pow(c,dp1,p1)
m2=pow(c,dp2,p2)
m3=pow(c,dp3,p3)
m4=pow(c,dp4,p4)

qInv1=gmpy2.invert(p2,p1)
h1=(qInv1*((m1-m2) % p1)) % p1
m5=m2+h1*p2
# p4=p1*p2显然我们已经有了p4不需要再组p1，p2了

qInv2=gmpy2.invert(p3,p4)
h2=(qInv2*((m4-m3)% p4)) % p4
m6=m3+h2*p3
print(m5)
print(m6)
print(n2s(37414068822610300895733121671361921539811268381281827443338586128643155689462893891362487514883
```

输出为：

```
37414068822610300895733121671361921539811268381281827443338586128643155689462893891362487514883
37414068822610300895733121671361921539811268381281827443338586128643155689462893891362487514883
b'hgame{EulEr:fUNcTIon;iS.So*IMpORTaNt*In&RsA}'
```

m5，m6是相同的，也就是说解一组p1，p2就出来了，至于为什么.....还没搞懂（dog）

RSA Attack 3

- **思路**

  根据ouput里的长度可知这是一种针对e过大的情况，week2中有出现e过小的情况

  检索得知是维纳攻击，具体原理参考RSA之初学维纳攻击

原理过于复杂（至少对于我来说）所以直接用文中的代码进行解密

- **代码**

```
"""
user:鸢柒
purpose:rsa——3
time:2022.02.10__13:33
"""
from libnum import n2s
from math import isqrt
import gmpy2

# 费马是一开始尝试用来分解因数的，10000000结束了才退出
# 分解出的p*q和n在前几百位都是相同的，后面一两百位是不一样的
# def fermat(n, verbose=True):
#     a = isqrt(n)  # int(ceil(n**0.5))
#     b2 = a * a - n
#     b = isqrt(n)  # int(b2**0.5)
#     count = 0
#     while b * b != b2 and count < 10000000:
#         a = a + 1
#         b2 = a * a - n
#         b = isqrt(b2)  # int(b2**0.5)
#         count += 1
#     p = a + b
#     q = a - b
#     a = [p, q]
#     print(p)
#     print(q)
#     return p * q


with open('F:\\Face_to_the_prison\\Crypto\\HGAME\\Week3\\RSA Attack 3\\output.txt', 'r') as f:
    a = f.readlines()
n = eval(a[0][4:-1])
e = eval(a[1][4:-1])
c = eval(a[2][4:-1])
print(len(str(n)), len(str(e)), len(str(c)))


def transform(x, y):  # 使用辗转相处将分数 x/y 转为连分数的形式
    res = []
    while y:
        res.append(x // y)
        x, y = y, x % y
    return res


def continued_fraction(sub_res):
    numerator, denominator = 1, 0
    for i in sub_res[::-1]:  # 从sublist的后面往前循环
        denominator, numerator = numerator, i * numerator + denominator
```

```
    return denominator, numerator  # 得到渐进分数的分母和分子，并返回


# 求解每个渐进分数
def sub_fraction(x, y):
    res = transform(x, y)
    res = list(map(continued_fraction, (res[0:i] for i in range(1, len(res)))))  # 将连分数的结果
    return res


def get_pq(a, b, c):  # 由p+q和pq的值通过维达定理来求解p和q
    par = gmpy2.isqrt(b * b - 4 * a * c)  # 由上述可得，开根号一定是整数，因为有解
    x1, x2 = (-b + par) // (2 * a), (-b - par) // (2 * a)
    return x1, x2


def wienerAttack(e, n):
    for (d, k) in sub_fraction(e, n):  # 用一个for循环来注意试探e/n的连续函数的渐进分数，直到找到一个
        if k == 0:  # 可能会出现连分数的第一个为0的情况，排除
            continue
        if (e * d - 1) % k != 0:  # ed=1 (mod φ(n)) 因此如果找到了d的话，(ed-1)会整除φ(n),也就是存
            continue

        phi = (e * d - 1) // k  # 这个结果就是 φ(n)
        px, qy = get_pq(1, n - phi + 1, n)
        if px * qy == n:
            p, q = abs(int(px)), abs(int(qy))  # 可能会得到两个负数，负负得正未尝不会出现
            d = gmpy2.invert(e, (p - 1) * (q - 1))  # 求ed=1 (mod  φ(n))的结果，也就是e关于 φ(n)的
            return d
    print("该方法不适用")


d = wienerAttack(e, n)
print("d=", d)
m = pow(c, d, n)
print(m)
flag=n2s(176682944595673240063788601764110191909427842736769909066978679521165615012306986062274
print(flag)
```

得到如下：

```
1233 1232 1233
d= 13094612077654083919
176682944595673240063788601764110191909427842736769909066978679521165615012306986062747971925050
b'hgame{dO|YOU:kNOw!tHE*PRINcIplE*bEhInd%WInNEr#aTTacK}'
```