xctf攻防世界 CRYPTO高手进阶区 OldDriver

原创

l8947943 于 2022-02-14 10:41:41 发布 100 收藏 1

分类专栏： 攻防世界crypto之路 文章标签： 算法 图论 CRYPTO

本文链接：https://blog.csdn.net/l8947943/article/details/122919325

1. 进入环境，下载附件

这个题我感觉还是挺难的，看了两天终于能理解了一部分！附件给了个enc.txt文件，打开看了一眼，n和c都很大，而e较小且固定，猜测与rsa有关，看完wp才知道是 低加密指数广播攻击

查阅相关博客，什么是低加密指数广播攻击？

- **加密指数e非常小**
- **一份明文使用不同的模数n，相同的加密指数e进行多次加密**
- **可以拿到每一份加密后的密文和对应的模数n、加密指数e**
- **可以根据中国剩余剩余定理求得m^e，最后得到明文**

2. 问题分析

1. **解题思路**

   参考大佬的链接https://www.cnblogs.com/lingxuer/p/15018183.html

- **观察附件，发现有e,n,c三个参数名，考虑本题目会涉及到RSA**
- **观察附件，所有的e均为相同的值，且该值并不是很大**
- **本题目可能由明文m通过多个公钥(n,e)加密且得到多个密文n**
- **考虑使用低加密指数广播攻击获取明文**
- **通过判断，所有的密文n均互素，使用中国剩余定理进行求解**
- **将得到的值开e次方，得到的值转为16进制通过base16转化获得flag**

2. **什么是中国余数定理？**

- 参考链接中国剩余定理介绍
- 百度百科关于余数定理的计算孙子定理

在《孙子算经》中有这样一个问题：“今有物不知其数，三三数之剩二（除以3余2），五五数之剩三（除以5余3），七七数之剩二（除以7余2），问物几何？”这个问题称为“孙子问题”，该问题的一般解法国际上称为“中国剩余定理”。

在《孙子歌诀》中给出了解决这个问题的解法：**三人同行七十稀，五树梅花廿一支，七子团圆正半月，除百零五便得知。**很是朗朗上口，但这是什么意思呢？

具体解法分三步：

找出三个数：

1.从3和5的公倍数中找出被7除余1的最小数15，从3和7的公倍数中找出被5除余1 的最小数21，最后从5和7的公倍数中找出除3余1的最小数70。

2.用15乘以2（2为最终结果除以7的余数），用21乘以3（3为最终结果除以5的余数），同理，用70乘以2（2为最终结果除以3的余数），然后把三个乘积相加（15*2+21*3+70*2）得到和233。

3.用233除以3，5，7三个数的最小公倍数105，得到余数23，即233%105=23。这个余数23就是符合条件的最小数。

本题的核心就是求解余数定理的结果，用其结果进行解密，那么先模仿写一份余数定理代码：

```python
import gmpy2


def crt(b, m):
    # 判断是否互素
    for i in range(len(m) - 1):
        for j in range(i + 1, len(m)):
            # 或者math.gcd()也可以
            if gmpy2.gcd(m[i], m[j]) != 1:
                print('模数中存在不互质的数! ')
                return -1

    # 乘积计算
    M = 1
    for i in range(len(m)):
        M *= m[i]

    Mm = []
    # 求余数 M/m[i]
    for i in range(len(m)):
        Mm.append(M // m[i])

    # 求Mm[i]的乘法逆元
    Mm_ = []
    for i in range(len(m)):
        t, a, _ = gmpy2.gcdext(Mm[i], m[i])
        Mm_.append(int(a % m[i]))

    # 求的累加
    y = 0
    for i in range(len(m)):
        y += (Mm[i] * Mm_[i] * b[i])
    y = y % M
    return y

r = crt(b = [2, 3, 2], m = [3, 5, 7])
```

3. **利用n和c与余数定理函数进行求解**

   代码如下：

```python
from Crypto.Util.number import long_to_bytes


cip_keys = [
    {
        "c": 736606757474117146172206513324291608049550591366325033008274746538367689397041147655074839484143741
81053123539710950034243226796169403711230289821895020 42,
        "e": 10,
        "n": 2516250705233971442183968887373459617775112403672383100330095976113781149071520574294173840654815 02
40861779301784133652165908227917415483137585388986274803},
    {
        "c": 2196282532330046915179592028988688656279094277154685850084217980656643576710380397888514877213930 54
84319688249368999503784441507383476095946258011317951461,
        "e": 10,
        "n": 2397685958990441979832081209768185865232547379189123271043199720289781958063493707090062521321809 53
30766877190212418023297341732808839488308551126409983193},
    {
        "c": 6569689420274066957835983390583585286570087619048110141187700584193792695235405077811544355169290 38
23571493741070764060861541033518978907935989976870539 83
```

```
        "e": 10,
        "n": 185037828368585400439745580356016546109489155056452198201502510623051201487455459065675486501918320
908234828526043464783353537845010767619226053618487036 23},
    {
        "c": 450824616804451351845249388271353639063674154155180582179033897379761597127186724858437981311412547
819528469269592866894655362548317963326605712296754705 2,
        "e": 10,
        "n": 233830874785455122187131579329347461107217068190774234180602200836577134285035828019098071428026473
679942897750155951005411683670830975061938094513650107 23},
    {
        "c": 229661056702912823355888430182441615527644863731179428659669040761911223374355425532767439388176867
295547143154948189227538801989458972224221372684276116 72,
        "e": 10,
        "n": 317756490898614286710579090761441528707967225281125804794420733650539160125072734330284517554369870
547224960577497317584759583011640827550031956320053084 93},
    {
        "c": 179633130634050457429681369162198383521355617853895343812629792645853978968444708790236865085403551
609985331229702392610720206892171531266493908256467120 87,
        "e": 10,
        "n": 222463420229434328206961904441556652899283786538411726322832278881744954022486330610106155726421265
845911037503389192139456460748338239055216430258790539 49},
    {
        "c": 165241753470902945038057065397370532098611767959756387302268314080050748256048294831013154094822779
704550539033314619158674926924954816824731640407401463 9,
        "e": 10,
        "n": 253954611426706312681561061360283257443933584366175286779672493473535249246550011518495440222017725
000332808223726613443526074347386960517790957365478130 43},
    {
        "c": 155857717344883510394566313940404977595686794295106192197661917808076753617418592904907324511126487
766481267797593684282051946847215164970262909817862393 52,
        "e": 10,
        "n": 320565088927441849012894132877280398913038323115486081410882278763267536741541247751327769284819353
781847567567851075407816325702953304867382681731678090 47},
    {
        "c": 896512342163769405004421684452337916334747802912481503283281322505073255852423966064874628488414074
678882368188601057734225484101459457006746790568235979 7,
        "e": 10,
        "n": 528497662695418274742281894288206485741625395959853959922616498099074357422630205510500642688903333
928771735728116915998412531504602199868179644619707365 53},
    {
        "c": 135609457565430230085293881084469408471378530384370952445730358885312885773708290656663200693978983
948484848470303210189156383818339355809583427199889782 47,
        "e": 10,
        "n": 304159848003075789329463999875590889683556383543448233593972044191912418027217724994866156616990809
985024399015855739508890479185379066878407250054962386 21}
]

cips = []
keys = []

for cip_key in cip_keys:
    cips.append(cip_key['c'])
    keys.append(cip_key['n'])

r = crt(cips, keys)

res, _ = gmpy2.iroot(r, 10)
print(res)
```

```
print(long_to_bytes(res).decode())
```

得到最终的结果：`flag{wo0_th3_tr4in_i5_leav1ng_g3t_on_it}`

4. **再次复习RSA过程**

> RSA是一种公钥加密算法，RSA算法的具体描述如下：

1. 任意选取两个不同的大素数p和q计算乘积

$$n = pq, \varphi(n) = (p-1)(q-1)$$

2. 任意选取一个大整数e，满足

$$gcd(e, \varphi(n)) = 1$$

，整数e用做加密钥（注意：e的选取是很容易的，例如，所有大于p和q的素数都可用）

3. 确定的解密钥d，满足

$$(de) mod \varphi(n) = 1$$

，即

$$de = k\varphi(n) + 1, k \geq 1$$

是一个任意的整数；所以，若知道e和

$$\varphi(n)$$

，则很容易计算出d；

4. 公开整数n和e，秘密保存d
5. 将明文m（m<n是一个整数）加密成密文c，加密算法为

$$c = E(m) = m^e modn$$

6. 将密文c解密为明文m，解密算法为

$$m = D(c) = c^d modn$$

7. 然而只根据n和e（注意：不是p和q）要计算出d是不可能的。因此，任何人都可对明文进行加密，但只有授权用户（知道d）才可对密文解密。

学无止境，干他丫的！！！