Week_2 做题记录

- **Crypto**
  > - RSA Attack
  > - RSA Attack 2

Crypto

RSA参考资料：

> RSA攻击方法总结
> RSA大数分解工具

RSA Attack

- **思路**

  从output可以看出n为大数，普通的分离肯定是不行的了

  打开task.py文件，来分析一下下它的具体加密情况

```
m = s2n(flag)# 用libnum库的s2n对flag转化为整数，反过来用n2s即可
e = 65537
p = getPrime(80)
q = getPrime(80)
n = p * q
c = pow(m, e, n)
print("e =", e)
print("n =", n)
print("c =", c)
```

  标标准准的RSA加密，已知c，n，e，爆破私钥d，首先需要知道n的欧拉函数，需要p，q，采用工具factordb.com来分解出n的质因数p，q

```
p = 715800347513314032483037# http://factordb.com
q = 978782023871716954857211
oula_n = (p - 1) * (q - 1)
```

上代码

- **代码**

```
"""
user:鸢柒
purpose:rsa attack
time:2022.01.29__11:53
"""
import math # 测试用
from Crypto.Util.number import inverse
import libnum
e = 65537
n = 700612512827159827368074182577656505408114629807
c = 122622425510870177715177368049049966519567512708
p = 715800347513314032483037# http://factordb.com
q = 978782023871716954857211
oula_n = (p - 1) * (q - 1)
private_key = inverse(e, oula_n)  # 536622767389183848122360417472562479020563323833
print(private_key)
m=pow(c, private_key , n)
print(m)
flag = libnum.n2s(m)
print(flag)
```

得到：b'hgame{SHorTesT!fLAg}'

RSA Attack 2

- **思路**

根据题目提示这是三种有缺陷的RSA加密，先来看第一个：
有两段密文c1，c2，两段n：n1，n2，同一个e

再来看一下task.py的第一段加密：

```
flag_parts = list(map(s2n, re.findall(rf".{{,{ceil(len(flag) / 3)}}}", flag)))
```

flag被分成了三段，用s2n转化成整数来加密

第一段

```
print("# task1")
m = flag_parts[0]
e = 65537
p = getPrime(1024)
q = getPrime(1024)
r = getPrime(1024)
# n1，n2有相同的公约数q
n1 = p * q
c1 = pow(m, e, n1)
n2 = r * q
c2 = pow(m, e, n2)
print("e =", e)
print("n1 =", n1)
print("c1 =", c1)
print("n2 =", n2)
print("c2 =", c2)
```

工具真香

```
# http://factordb.com
p = 1181061717095186131903373801207216390961094338717585514817505596286078415251999333964010458
q = 1237153435219706840001287998760710428305707232181169311514672202447650558894176268065548681
r = 1692391430929639222133436869246773630889634856330270916455011513884825654902333237968896916
```

然后根据加密过程用乘法逆元求解私钥d

- **代码**

```
e = 65537
n_1 = 14611545605107950827581005165327694782823188603151768169731431418361306231114985037775917
n_2 = 20937478725109983803079185450449616567464596961348727453817249035110047585580142823551289
c_1 = 96507580355493298866427181643918380232881201369420374132076310537603691258499503164767234
c_2 = 11536506945313747180442473461658912307154460869003392732178457643224057969838224601059836
# http://factordb.com
p = 118106171709518613190337380120721639096109433871758551481750559628607841525199933396401045
q = 123715343521970684000128799876071042830570723218116931151467220244765055889417626806554868
r = 169239143092963922213343686924677363088963485633027091645501151388482565490233323796889691
oula_n_1 = (p - 1) * (q - 1)
oula_n_2 = (r - 1) * (q - 1)
private_key_1 = inverse(e, oula_n_1)
private_key_2 = inverse(e, oula_n_2)
print(private_key_1)
print(private_key_2)
m_1 = pow(c_1, private_key_1, n_1)
m_2 = pow(c_2, private_key_2, n_2)
print(m_1)
print(m_2)
flag_1 = libnum.n2s(m_1)
flag_2 = libnum.n2s(m_2)
print(flag_1)
print(flag_2)
```

得到第一部分：

b'hgame{RsA@hAS!a&VArIETY?of.'

b'hgame{RsA@hAS!a&VArIETY?of.'

第二段

查看task.py文件：

```
print("# task2")
m = flag_parts[1]
e = 7 #发现了，加密指数e这么小
p = getPrime(1024)
q = getPrime(1024)
n = p * q
c = pow(m, e, n)
print("e =", e)
print("n =", n)
print("c =", c)
```

查看output文件发现n贼大，c也贼大

低加密指数：当e=7时，如果明文过小，导致明文的7次方仍然小于n，那么通过直接对密文7次开方，即可得到明文。
原理：加密为c=m^e mod n
解密为m=c^d mod n
分析：如果e过小（m$^{e<n或者m}$e<2*n*）
就可以直接对c开方。
*m^e=k*n+c

---

csdn
原文链接：https://blog.csdn.net/cpongo3/article/details/89708313

- **代码**

```
e = 7
n = 1415787849225534630099334965381301810599188457752990952255555146837430794209621496460417273
c = 1026287102051911640631267468523836402353665784103475157284457098375029590949214910150086980
# http://factordb.com
print(gmpy2.iroot(c, e))
# 开方后n2s解码
print(libnum.n2s(26926584401348540331333678102939069838976561137078484378892509505))
```

得到：
b'AttacK^mEThodS^whAT:other!A'

第三段

查看task.py文件：

```
print("# task3")
m = flag_parts[2]
p = getPrime(1024)
q = getPrime(1024)
n = p * q
# 使用了相同的模n对相同的明文m进行了加密
e1 = getPrime(32)
e2 = getPrime(32)
c1 = pow(m, e1, n)
c2 = pow(m, e2, n)
print("n =", n)
print("e1 =", e1)
print("c1 =", c1)
print("e2 =", e2)
print("c2 =", c2)
```

使用了相同的模n对相同的明文m进行了加密，那么就可以在不分解n的情况下还原出明文m的值。(e_1和e_2互素)

---

c_1=m^(e_1)mod n

c_2=m^(e_2)mod n

需要：密文c_1,c_2 并且n相同.

结果:

c_1^{s_1} * c_2^{s_2}=m mod n

s_1 * e_1+s_2 * e_2=1

证明：gcd (e_1,e_2)=1

即存在s_1 ， s_2使得:

s_1 * e_1 + s_2 * e_2=1

又因为： c_1 = m^{e_1} mod n

c_2 = m^{e_2} mod n

通过代入化简可以得出:

c_1^{s_1} * c_2^{s_2}= m mod n

---

csdn

原文链接：https://blog.csdn.net/cpongo3/article/details/89708313

- **代码**

```
# 欧几里得算法
# 参考https://www.cnblogs.com/gwind/p/8013154.html
def egcd(a, b):
  if a == 0:
    return (b, 0, 1)
  else:
    g, y, x = egcd(b % a, a)
    return (g, x - (b // a) * y, y)

n = 18819509188106230363444813350468162056164434642729404632983082518225388069544777374544142317
e1 = 2519901323
c1 = 3230779726225544872531441169009307072073754578761888387983403206364548451496736513905460381
e2 = 3676335737
c2 = 94081859562227916143983671964170784679029465088879982233500738585416673645928312943#476906
s = egcd(e1, e2)
s1 = s[1]
s2 = s[2]
if s1 < 0:
    s1 = - s1
    c1 = inverse(c1, n)
elif s2 < 0:
    s2 = - s2
    c2 = inverse(c2, n)

m = pow(c1, s1, n) * pow(c2, s2, n) % n
print(libnum.n2s(m))
```

得到:

b'ttACK|METHOdS~do@you_KNOW}'

综合三段得到:

```
flag='hgame{RsA@hAS!a&VArIETY?of.AttacK^mEThodS^whAT:other!AttACK|METHOdS~do@you_KNOW}'
```