CTF-CRYPTO-RSA-partial

原创

大熊何在 于 2020-11-21 19:47:23 发布 130 收藏

分类专栏： CRYPTO

本文链接：https://blog.csdn.net/zippo1234/article/details/109881959

CTF-CRYPTO-RSA-partial

每天一题，只能多不能少

RSA-partial

题目分析

1. 私钥格式
2. 私钥恢复

开始

1.题目

```
题目介绍
这里有一张RSA私钥上半部分被挡住的截图，你能恢复私钥并解密出flag的内容吗？flag格式为0ctf{字符串}。
-----BEGIN RSA PRIVATE KEY-----
****隐藏的部分****
Os9mhOQRdqW2cwVrnNI72DLcAXpXUJ1HGwJBANWiJcDUGxZpnERxVw7s0913WXNt
V4GqdxCzG0pG5EHThtoTRbyX0aqRP4U/hQ9tRoSoDmBn+3HPITsnbCy67VkCQBM4
xZPTtUKM6Xi+16VTUnFVs9E4rqwIQCDAxn9UuVMBXlX2Cl0xOGUF4C5hItrX2woF
7LVS5EizR63CyRcPovMCQQDVyNbcWD7N88MhZjujKuSrHJot7WcCaRmTGEIJ6TkU
8NWt9BVjR4jVkZ2EqNd0KZWdQPukeynPcLlDEkIXyaQx
-----END RSA PRIVATE KEY-----
```

给出一个图片、题目.txt和flag.enc。图片的尾部隐写的就是被隐藏了部分内容的私钥。

2.分析

明显就是要通过恢复私钥来解密flag.enc。

（1）已给出部分解密

先把已给出部分进行base64解密

```python
#!python2
# -*- coding: utf-8 -*-
# @Time : 2020/11/20 23:15
# @Author : A.James
# @FileName: test.py
import base64
import binascii
a = """Os9mhOQRdqW2cwVrnNI72DLcAXpXUJ1HGwJBANWiJcDUGxZpnERxVw7s0913WXNt
V4GqdxCzG0pG5EHThtoTRbyX0aqRP4U/hQ9tRoSoDmBn+3HPITsnbCy67VkCQBM4
xZPTtUKM6Xi+16VTUnFVs9E4rqwIQCDAxn9UuVMBXlX2Cl0xOGUF4C5hItrX2woF
7LVS5EizR63CyRcPovMCQQDVyNbcWD7N88MhZjujKuSrHJot7WcCaRmTGEIJ6TkU
8NWt9BVjR4jVkZ2EqNd0KZWdQPukeynPcLlDEkIXyaQx"""
print binascii.hexlify(base64.b64decode(a))
```

得到

```
3acf6684e41176a5b673056b9cd23bd832dc017a57509d471b024100d5a225c0d41b16699c4471570eecd3dd7759736d5781aa7710b31b4a
46e441d386da1345bc97d1aa913f853f850f6d4684a80e6067fb71cf213b276c2cbaed5902401338c593d3b5428ce978bed7a553527155b3
d138aeac084020c0c67f54b953015e55f60a5d31386505e02e6122dad7db0a05ecb552e448b347adc2c9170fa2f3024100d5c8d6dc583ecd
f3c321663ba32ae4ab1c9a2ded6702691993184209e93914f0d5adf415634788d5919d84a8d77429959d40fba47b29cf70b943124217c9a4
31
```

（2）rsa私钥格式解析

参考：
OPENSSL中RSA私钥文件（PEM格式）解析
可以得知：

```
标签头  3082025c（4 bytes）    类型为SEQUENCE    后接 604 bytes
020100   INTEGER           长度为0                                内容为：VERSION
028181   INTEGER           长度为129 bytes              内容为：   n（modulus）
0203     INTEGER               长度为3 bytes                内容为：   e（publicExponent）
028180   INTEGER             长度为128 bytes             内容为：  d（privateExponent）
0241     INTEGER                 长度为65 bytes                内容为：     p（prime1）
0241      INTEGER                长度为65 bytes                 内容为：    q（prime2）
0240    INTEGER                  长度为64 bytes                内容为：    d mod（p-1） exponent1
0240      INTEGER                长度为 64 bytes               内容为：       d mod (q-1)  exponent2
0241      INTEGER                 长度为65 bytes               内容为：       q -1 mod p    coefficient
```

那么根据关键的标签头进行划分之后，可以得到

```
3acf6684e41176a5b673056b9cd23bd832dc017a57509d471b
0241//d mod（p-1） exponent1
00d5a225c0d41b16699c4471570eecd3dd7759736d5781aa7710b31b4a46e441d386da1345bc97d1aa913f853f850f6d4684a80e6067fb71
cf213b276c2cbaed59
0240//d mod (q-1)  exponent2
1338c593d3b5428ce978bed7a553527155b3d138aeac084020c0c67f54b953015e55f60a5d31386505e02e6122dad7db0a05ecb552e448b3
47adc2c9170fa2f3
0241//q -1 mod p    coefficient
00d5c8d6dc583ecdf3c321663ba32ae4ab1c9a2ded6702691993184209e93914f0d5adf415634788d5919d84a8d77429959d40fba47b29cf
70b943124217c9a431
```

（3）理论

因为：

```
e * dp == 1 (mod (p-1)) = d mod (p-1)
e * dq == 1 (mod (q-1)) = d mod (q-1)
q * qi == 1 (mod p) = q^-1 mod p
```

所以：

```
(e * dp -1)/k +1 == (p)
(e * dq -1)/j +1 == (q)
(q * qi -1)/l == (p)
```

3.上脚本破解pq

```python
#!python2
# -*- coding: utf-8 -*-
# @Time : 2020/11/21 0:09
# @Author : A.James
# @FileName: pq.py
import gmpy2
d_p = 0xd5a225c0d41b16699c4471570eecd3dd7759736d5781aa7710b31b4a46e441d386da1345bc97d1aa913f853f850f6d4684a80e60
67fb71cf213b276c2cbaed59
d_q = 0x1338c593d3b5428ce978bed7a553527155b3d138aeac084020c0c67f54b953015e55f60a5d31386505e02e6122dad7db0a05ecb5
52e448b347adc2c9170fa2f3
e = 65537
for k_p in range(1, e):
    if (e*d_p - 1) % k_p == 0:
        p = (e*d_p - 1) / k_p + 1
        if gmpy2.is_prime(p):
            print '[p] {}'.format(p)
            break
for k_q in range(1, e):
    if (e*d_q - 1) % k_q == 0:
        q = (e*d_q - 1) / k_q + 1
        if gmpy2.is_prime(q):
            print '[q] {}'.format(q)
            break
```

得到

```
[p] 128834299396391004790030585185232484938216882076971384178346312186380275645623066202148639884476813006665382
129185724721287329437847115270132247774747072569
[q] 125028936349231615998244651464070698822285137769477072954768059973117768558790240022895935986579497839370419
296684431152244773691360895579114640461181273 87
```

4.常规套路

```
#!python2
# -*- coding: utf-8 -*-
# @Time : 2020/11/20 23:07
# @Author : A.James
# @FileName: tt.py
from Crypto.Util.number import *
import gmpy2
import libnum
p = 12883429939639100479003058518523248493821688207697138417834631218638027564562306620214863988447681300666538212918572472128732943784711527013224777474072569
q = 12502893634923161599824465146407069882228513776947707295476805997311776855879024002289593598657949783937041929668443115224477369136089557911464046118127387
n = p * q
e = 65537
phi = (p - 1) * (q - 1)
d = gmpy2.invert(e, phi)
with open('flag.enc','rb') as f:
    flag = f.read()
c = bytes_to_long(flag)
m = pow(c, d, n)
print libnum.n2s(m)
```

得到

```
��K ���f�z��d^5�*������h\����ȟ3�}�v?�n���j'�!EF�~�K:/6!<�}=�& 0ctf{Keep_ca1m_and_s01ve_the_RSA_Eeeequati0n!!!}
```

5.get flag

```
0ctf{Keep_ca1m_and_s01ve_the_RSA_Eeeequati0n!!!}
```

结语

理论基础。