xctf攻防世界 CRYPTO高手进阶区 best_rsa

原创

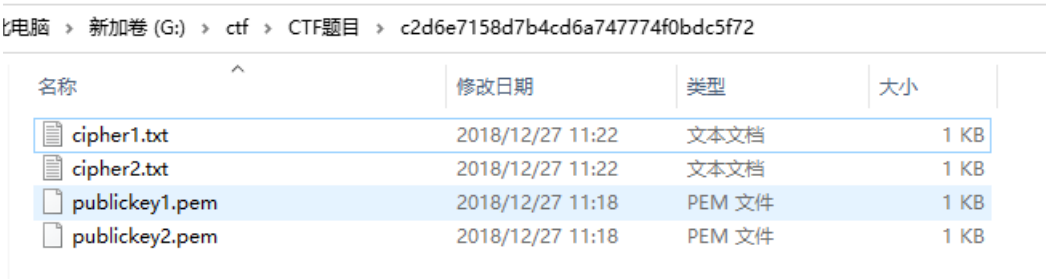
l8947943 已于 2022-02-15 10:44:34 修改 2411 收藏 1

分类专栏： 攻防世界crypto之路 文章标签： 安全 https 网络协议 crypto

于 2022-02-14 19:07:26 首次发布

1. 进入环境，下载附件

题目给的压缩包，包括4个文件，如图：



给出了2个公钥文件和和2个密文文件，用常规的RSA解密方式分别解密，解密失败（n为2048位难以分解）

2. 问题分析

1. 继续复习RSA

- 明文为m，密文为c，模数n = p * q
- 使用欧拉函数，φ(n) = (p - 1) * (q- 1)
- 选取一个大整数e，使得gcd(e, φ(n)) = 1，e用来加密秘钥
- 私钥d可以由欧拉函数值计算出来，满足ed mod φ(n) ≡ 1
- 将明文m加密成密文c：m^e ≡ c (mod n)
- 将密文c解密成明文m：c^d ≡ m (mod n)

2. 共模攻击

猜测应该是同一个明文，使用了2个不同的公钥加密得到了不同的密文，对同一明文的多次加密使用相同的模数和不同的公钥指数可能导致共模攻击。

翻看大佬的wp后：https://blog.csdn.net/weixin_44795952/article/details/108933406，明白了什么是共模攻击

所谓共模，就是明文m相同，模n相同，用两个公钥e1,e2加密得到两个私钥d1,d2和两个密文c1,c2
共模攻击，即当n不变的情况下，知道n,e1,e2,c1,c2 。可以在不知道d1,d2的情况下，解出m。
这里有个条件，即

```
gcd(e1,e2)=1
```

攻击原理

此处引用博主的分析：https://blog.csdn.net/weixin_44795952/article/details/108933406

有整数 s_1，s_2（一正一负），满足：

$e_1 * s_1 +$

根据扩展欧几里德算法，我们可以得到该式子的一组解（s1,s2），假设s1为正数,s2为负数。

同时需要了解欧几里得算法：
其实就是辗转相除法，得到最大公约数：

```
d = gcd(b,a mod b) //这里假设a>b
# 一次辗转相除如下
gcd(a,b) = gcd(b,a mod b)
# 一直辗转相除下去，可得：
gcd(a,b) = gcd(b,a mod b) = ... = gcd(m,0)
其中m为最大公约数
```

扩展欧几里得算法

```
对于不完全为 0 的非负整数 a，b，有gcd（a，b）
必然存在整数对 x，y ，使得 gcd（a，b）=a*x+b*y。
```

代码如下：

```
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
```

证明：

$c = m \quad \%n$

那么则有：

$s2$

两式相乘，化简则有：

$s2$

又因为有：

$e_1 * s_1 +$

最终则有：

$s2$

需要注意的是：
一个数的负次幂，在数论模运算中，要求一个数的负数次幂，与常规方法并不一样。比如此处要求c2的s2次幂，就要先计算c2的模反元素c2r，然后求c2r的-s2次幂

所以我们有以下脚本：

4. **最终解题脚本**

```python
from Crypto.Util.number import long_to_bytes, bytes_to_long
from Crypto.PublicKey import RSA
from gmpy2 import gcd, invert


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


with open('pic/publickey1.pem', 'rb') as f:
    f1 = f.read()
pub1 = RSA.importKey(f1)
# 拿到共模数
n = int(pub1.n)
# 拿到公钥e1
e1 = int(pub1.e)

with open('pic/publickey2.pem', 'rb') as f:
    f2 = f.read()
pub2 = RSA.importKey(f2)

# 拿到公钥e2
e2 = int(pub2.e)

with open('pic/cipher1.txt', 'rb') as f:
    c1 = f.read()
# 得到密文c1
c1 = bytes_to_long(c1)
print(c1)

with open('pic/cipher2.txt', 'rb') as f:
    c2 = f.read()
# 得到密文c2
c2 = bytes_to_long(c2)
print(c2)

# 得到扩展欧几里得中的计算模系数
s = egcd(e1, e2)

s1 = s[1]
s2 = s[2]

# 避免负指数运算
if s1 < 0:
    s1 = -s1
    c1 = invert(c1, n)
elif s2 < 0:
    s2 = -s2
    c2 = invert(c2, n)

#
m = pow(c1, s1, n) * pow(c2, s2, n) % n
print(m)
print(long_to_bytes(m).decode())
```

最终答案为：flag{interesting_rsa}