CTF-Crypto-RSA基本原理及常见攻击方法

原创

昭z_ysa2022 于 2020-10-04 13:14:42 发布 1196 收藏 10

分类专栏： 笔记 文章标签： 密码学 加密解密

本文链接：https://blog.csdn.net/lSHobbyst/article/details/108918079

0X00 RSA简介：

> 1977年，三位数学家Rivest、Shamir 和 Adleman
> 设计了一种算法，可以实现非对称加密。这种算法用他们三个人的名字命名，叫做RSA算法。
> 从那时直到现在，RSA算法一直是最广为使用的"非对称加密算法"。

0X01 数学背景：

> 互质
>
> > 从小学开始，我们就了解了什么是质数。互质是针对多个数字而言的，如果两个正整数，除了1以外，没有其他公因子，那么就称这两个数是互质关系（注意，这里并没有说这两个数一定是质数或有一个为质数。比如15跟4就是互质关系）。以下有一些关于质数与互质的性质：
>
> 1:质数只能被1和它自身整除
> 2:任意两个质数都是互质关系
> 3:如果两个数之中，较大的那个数是质数，则两者构成互质关系
> 4:如果两个数之中，较小的那个数是质数，且较大数不为较小数的整数倍，则两者构成互质关系 1和任意一个自然数是都是互质关系
> 5:p是大于1的整数，则p和p-1构成互质关系 p是大于1的奇数，则p和p-2构成互质关系

> 欧拉函数
>
> > 欧拉函数是求小于x并且和x互质的数的个数。其通式为：`ϕ(x) = x(1-1/p1)(1-1/p2)(1-1/p3)(1-1/p4)…..(1-1/pn)。`
> > 其中p1, p2……pn为x的所有质因数，x是不为0的整数。看到这里是不是有一些头疼，太理论的东西的确不够具象。我们且不去理会后面公式计算与论证，因为已经超出本文的范围了。就前一句来说说吧，欧拉函数是求小于x并且和x互质的数的个数。这里我可以列举一个例子：
> > 令x = 16，那么x的所有质因数为：`ϕ(16) = 16 * (1 - 1/2) = 8`
> > 我们也可以枚举出所有比16小，且与16互质的数：1, 3, 5, 7, 9, 11, 13, 15
>
> 欧拉函数的性质具体可以百度

> 模反元素
>
> > 定义：如果两个正整数a和n互质，那么一定可以找到整数b，使得 ab-1 被n整除，或者说ab被n除的余数是1。
>
> 关于模反元素的求解，使用的是朴素的解法。如果想要更进一步了解的话，请自行搜索其他解法（比如：辗转相除法、欧几里德算法）。

0X03 RSA原理

> RSA是一种算法，并且广泛应用于现代，用于保密通信。
>
> > RSA算法涉及三个参数,n,e,d，其中分为私钥和公钥，私钥是n,d，公钥是n,e

n是两个素数的乘积，一般这两个素数在RSA中用字母p,q表示,e是一个素数， d是e模varphi(n)
的逆元,d是由e,p,q可以求解出的

> > > 一般CTF就是把我们想要获得的flag作为明文，RSA中表示为m,然后通过RSA加密，得到密文，RSA中表示为C。

加密过程

c=m^e mod n

```
c=pow(m,e,n)
```

解密过程

m=c^d mod n

```
m=pow(c,d,n)
```

求解私钥d

```
d = gmpy2.invert(e, (p-1)*(q-1))
```

一般来说，n，e是公开的，但是由于n一般是两个大素数的乘积，所以我们很难求解出d，所以RSA加密就是利用现代无法快速实现大素数的分解，所存在的一种安全的非对称加密。

0X04 CTF中常见的类型：

1，已知 p ，q，e 求 d

```
（ed 除以 (q-1)(p-1) 的 余数 为 1 ）
```

```
import gmpy2
p = 38456719616722997
q = 44106885765559411
e = 65537
s = (p-1)*(q-1)
d = gmpy2.invert(e,s)
print ("dec: " + str(d))
print ("hex: " +  hex(d))
```

2，已知 n（比较小），e 求 d

```
（n = q * p ，ed 除以 (q-1)(p-1) 的 余数 为 1 ）(n往往是一个 1024bit 的超大数，很难分解为两个 质数)
```

n的分解使用yafu 中的 factor（t) 命令 ,简单的也可以使用在线网站

3，已知 公钥（n, e） 和 密文 c 求 明文 m
方法一：（n,e不太大的情况下）
首先将 n 分解为 q 和 p

再利用脚本：

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

c = 0x6cd55a2bbb49dfd2831e34b76cb5bdfad34418a4be96180b618581e9b6319f86
n = 10853984726857399027523402435467243724652508507660551696032000572274158989864 1
#n = int("",16)
e = 65537
#e = int("",16)
q = 333360321402603178263879595968004169219
p = 325593180411801742356727264127253758939

d = libnum.invmod(e, (p - 1) * (q - 1))
m = pow(c, d, n)   # m 的十进制形式
string = long_to_bytes(m)  # m明文
print(string)  # 结果为 b‘ m ’ 的形式
```

方法二：（n , e 比较大的时候）

直接利用 RsaCtfTool进行爆破：

利用 n,e生成公钥文件 test.pem：

```
python RsaCtfTool.py --createpub --n 46065781....  --e 3546111...  > test.pub
```

再使用公钥爆破

4，已知密文文件和 公钥文件求解明文 m

直接用 RsaCtfTool进行破解：

```
python RsaCtfTool.py --publickey key.pem --uncipherfile cipher.bin
```

5，有私钥 private.pem 和密文 flag.enc
方法一：利用 rsactftool。

```
python RsaCtfTool.py --private private.pem  --uncipherfile flag.enc
```

方法二：利用 openssl：

```
openssl rsautl -decrypt -in flag.enc -inkey private.pem
```

6，已知c ，e，n（非常大）和 dp，dq，求解明文m
给了e, n, c,由于特别大，没法直接用质因数分解求得 q， p

但是还给了

```
phint  =  d  %  (p - 1)       ==  phint  = dp`
```

```
qhint   =  q   %  (p - 1)       == qhint  = dq
```

EX：

```
import gmpy2
import libnum
e=65537
n=16969752165509132627630266968748854330340701692125427619559836488350298234735571480353078614975580378467355952
33375531393551651377355216339295265632149026845255660485896689995624210700841055865792434429565193929732800793322
45741660910510032969527598266270511004857674534802203387399678231880894252328431133224653544948661283777645985028
20760952665481664515555891519774506256912458741237871604981404067066507948005564487347075660299338726193956695 8
80629659978294346014158204515097103121121861709128328411857371402926633122732739872426517035264679406870278964598
08081000554937639953511082005247241984680180911018655716212 7
dp=17816257752910288702696852575211080903295430127287054677825469139515376426236217692464411221899486713749909464
05164459867410646825591310622618379116284293794090970292165263334749393009999335413089903796624326168039618287078
78192646490488534062803960418790874890435529393047389228718835244370645215187358081805
c=0x6c78dcee37830f3ec4ab4989d40fbb595060b3fbc395d52ad26defc13372c1a3948c5388f4e450e46e016c7803133d6881e5efc3b90a4789448097c94124590b1e7949f2524d7edccd61a27691c18d090ac1f54643b56314130604541758 1e3b263f4ad2816136a48b106f3058b08e2a810f4ae8ef25916cc110b41ac8158ce69ecbe20fc60c1ddb20154c6646bc5142aefe47abf053a8ac949d5bc057bb18b191ad08070fe9ec5d76b1fceae685514532448c1b388b2d38e7241ac19c296e95e4e021a3a4015d909a1d53a2eb7fa86f6329f4e6c937f958be576c58fab4d9c9126999c99bb28718efc41a6f5db52b47942a2ddf21639f020b5489699cf22b46

for i in range(1,65538):
    if (dp*e-1)%i == 0:
        if n%(((dp*e-1)//i)+1)==0:
            p=((dp*e-1)//i)+1
            q=n//(((dp*e-1)//i)+1)
            phi = (p-1)*(q-1)
            d = gmpy2.invert(e,phi)%phi
            print(libnum.n2s(pow(c,d,n)))
```

7，已知n（非常大）e,d求p,q（无法直接 从n分解）
无法直接分解得到p和q。

```
(python 2)

import random
from md5 import md5

def gcd(a, b):
   if a < b:
     a, b = b, a
   while b != 0:
     temp = a % b
     a = b
     b = temp
   return a

def getpq(n,e,d):
 p = 1
 q = 1
 while p==1 and q==1:
  k = d * e - 1
  g = random.randint ( 0 , n )
  while p==1 and q==1 and k % 2 == 0:
   k /= 2
   y = pow(g,k,n)
   if y!=1 and gcd(y-1,n)>1:
    p = gcd(y-1,n)
    q = n/p
 return p,q

def main():
 n = 16352578963372306131642407541567045533766691177138375676491913897592458965544068296813122740126583082006556
217616296009516413202833698268845634497478988128850373221853516973259086845725813424850548682503827191121548693288763243619033224322698075987667531863213468223654181658012754897588147027437229269098246969811226129883327598021859724836993626315476699384610680857047403431430525708390695622848315322636785398223207468754197643541958599210127261345770914514670199047435085714403641469016212958361993969304545214061560160267760786482163373784437641808292654489343487613446165542988382687729593384887516272690654309
 e = 65537
 d = 94599283799736674301380685280594381390923686253390792532895605779853044350622131213982318758322648944583146295754555534857526856437432666546308299574420087752597763115856540148581653417575472841120618851580068814757405535328265762608394303439607385208223679755286443291726688776962087410076483700455205352980401616754077792393004666816154938926922655422902554086735338530116621349538694326325540082353408648033776103524381462645247707103452734397241070801901829182855474261665618037166440894140783894750721033154326381975781861065766267288690203662140774551945549307255760232749227411159412147896000891667544764764534
 p,q = getpq(n,e,d)
        print p
        print q
        print "Flag: flag{%s}" %md5(str(p + q)).hexdigest()
if __name__ == '__main__':
 main()
```

8，提取私钥中的信息

```
python RsaCtfTool.py --key private.pem  --dumpkey
```

9,利用公钥pub.key/pub.pem文件生成 私钥文件

```
python RsaCtfTool.py --publickey pubkey.pem  --private > private.pem
```

或

```
python2 RsaCtfTool.py --publickey pub.key  --private > private.key
```

10，n分解出多个不同的因子时 ，求明文m
EX：

```
n= 544187306850902797629107353619267427694837163600853983242783
e= 39293
c= 43925489581832041340882702239805368586734326797171233201l972
m=???
```

对n进行质因数分解，得到了3个质因数，（这里知道欧拉公式的性质的话 就很好解)

> $\phi(x * y * z) = \phi(x) * \phi(y) * \phi(z)=(x-1)(y-1)(z-1)$

```
(python2)

import gmpy2
from Crypto.Util.number import long_to_bytes

n= 544187306850902797629107353619267427694837163600853983242783
e= 39293
c= 43925489581832041340882702239805368586734326797171233201l972
p1 = 67724172605733871
p2 = 11571390939636959887
p3 = 694415063702720454699679
phi = (p1-1)*(p2-1)*(p3-1)
d = gmpy2.invert(e, phi)
m = pow(c, d, n)
print long_to_bytes(m)
```

0X05 爆破攻击方法：

1：小明文攻击

小明文攻击是基于低加密指数的，主要分成两种情况。
*明文过小，导致明文的e次方仍然小于n

```
('n=', '0xad03794ef170d81aad370dccb7b92af7d174c10e0ae9ddc99b7dc5f93af6c65b51cc9c40941b002c7633caf8cd50e1b73aa942
c8488d46c0032064306de388151814982b6d35b4e2a62dd647f527b31b4f826c36848dc52999574a8694460e1b59b4e96bda1341d3ba5f99
1f0000a56004d47681ecfd37a5e64bd198617f8dadL')
('e=', '0x3')
('c=', '0x10652cdf6f422470ea251f77L')
```

这种情况直接对密文e次开方，即可得到明文
**明文的三次方虽然比n大，但是大不了多少

```
('n=', '0x9683f5f8073b6cd9df96ee4dbe6629c7965e1edd2854afa113d80c44f5dfcf030a18c1b2ff40575fe8e222230d7bb5b6dd8c41
9c9d4bca1a7e84440a2a87f691e2c0c76caaab61492db143a61132f584ba874a98363c23e93218ac83d1dd715db6711009ceda2a31820bba
caf1b6171bbaa68d1be76fe986e4b4c1b66d10af25L')
('e=', '0x3')
('c=', '0x8541ee560f77d8fe536d48eab425b0505e86178e6ffefa1b0c37ccbfc6cb5f9a7727baeb3916356d6fce3205cd4e586a1cc407
703b3f709e2011d7b66eaaeea9e381e595b4d515c433682eb3906d9870fadbffd0695c0168aa26447f7a049c260456f51e937ce75b74e5c3
c2bd7709b981898016a3a18f15ae99763ff40805aaL')
```

爆破即可，每次加上一个n

2：低加密指数广播攻击

如果选取的加密指数较低，并且使用了相同的加密指数给一个接受者的群发送相同的信息，那么可以进行广播攻击得到明文。
这个识别起来比较简单，一般来说都是给了三组加密的参数和明密文，其中题目很明确地能告诉你这三组的明文都是一样的，并且e都取了一个较小的数字。

3：低解密指数攻击

主要利用的是私钥d很小，表现形式一般是e很大

```
n = 92476066235238477726989531616164556648218671835712180569700997513016822051231157160894867998374473979253088879767759948171759949457602781975279096217934 69
e = 27587468384672288862881213094354358587433516035212531881921186101712498639965289973292625430363076074737388345935775494312333025500409503290686394032069
```

4：共模攻击
识别：若干次加密，e不同，n相同，m相同。就可以在不分解n和求d的前提下，解出明文m。

```
('n=', '0xc42b9d872f8ecf90b4832199771bbd8d9bafb213747d905a644baa42144f316dc224e7914f8a5d361eeab930adf5ea7fbe1416e58b3fae34ca7e6d2a3145e04af02cf5a4f14539fff032bccd7bb9cf85b12d7d36dbc870b57e11aa5704304d08eff685fe4ccd707e308dfac6a1167d79199ffa9396c4f2efb4770256253d1407L')
('e1=', '0xc21000af014a98b2455dec479L')
('e2=', '0x9935842d63b75899ddd81b467L')
('c1=', '0xc0204d515a275954bbc8390d80efa1cca3bb29724ed7ba18f861913e28b6400298603b920d484284ad9c1c175587496300355395cb06b32603e779ec9b97f7eea6bb0de42c54f7f60e6e1171496efef0de8048e6074658084d080bd346db426888084e6dd45cb89b283247443de75328d47f9bd64adbd9be86043c6d13c7ed41L')
('c2=', '0xc4053ed3455c15174e5699ab6eb09b830a98b79e92e7518b713e828faca4d6d02306a65a8ec70893ca8a56943a7074e6de8649f099164cad33b8ca93fce1656f0712b990cce06642250c52a80d19c2afa94a4e158139028ac89c811e6be8d7b6984b6c1edcdd752e4955e3a6f1ab38cf2edb4474a80e03d6c313eb8ebf4e98ccL'
```

推导过程

```
首先，两个加密指数互质：
gcd(e1,e2)=1

即存在s1、s2使得：
s1+*e1+s2*e2=1

又因为：
c1≡m^e1 mod n
c2≡m mod n

代入化简可得：
c1^s1 * c2^s2 ≡ m mod n

即可求出明文
```

5：Stereotyped messages攻击

```
('n=', '0xf85539597ee444f3fcad07142ecf6eaae5320301244a7cedc50b2beed7e60ffa11ccf28c1a590fb81346fb16b0cecd046a1f63f0bf93185c109b8c93068ec02fL')
('e=', '0x3')
('c=', '0xa75c3c8a19ed9c911d851917e442a8e7b425e4b7f92205ca532a2ab0f5abe6cb86d164cc61374877f9e88e7bca606b43c79f1d59deadfcc68c3db52e5fc42f0L')
('m=', '0x666c6167206973203a7465737431323131313131313131313133343536000000000000000000L')
```

给了明文的高位，可以尝试使用Stereotyped messages攻击
我们需要使用sage实现该算法
可以安装SageMath
或者在线网站
https://sagecell.sagemath.org/

简单的RSA原理及基本攻击方式就写到这里，文章如有漏洞请联系E-mail : hl4836@163.com
本文作者:C1heck0ut