

# BugkuCTF-Crypto题rsa

原创

彬彬有礼am\_03



于 2021-08-31 15:02:50 发布



129



收藏

分类专栏: # BugkuCTF-Crypto 文章标签: python

版权声明: 本文为博主原创文章, 遵循CC 4.0 BY-SA 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/am\\_03/article/details/120017116](https://blog.csdn.net/am_03/article/details/120017116)

版权



[BugkuCTF-Crypto 专栏收录该内容](#)

20 篇文章 1 订阅

订阅专栏

## 解题流程

N:  
44805702308429948986937295038344724551111702946293887440291374827113042722959807783009991191361738959980001362153608653831326536296324364014116083801800299924808808408124059575326277994453386540170514591485388877700729895720113708042397  
1895034217532738794541303284914147124733996722397  
  
394011184413817522963720141073212909030454383621815713893089381279463916544695635894245413896307834463779449532371811251083461472159591679153289162126336461216660254300880823524385860323483274771221625783024389646880946881528380124049684071183425397737  
08625380487318613668621867232657614641549327629  
  
enc:  
3813899111622899503821567126649138106004463012581737646532396805615621562148167644578852889498107751567159111346665315625711136056999659173862949964539280811820482705386021545912379121  
1863815912025463777582942833563335693449424289125

n, e已经给出, 可以看出e特别大, 在e特别大的情况下, 可以使用wiener attack的方法进行破解, 正好工具RsaCtfTool集成了wiener attack的方法, 所以可以直接使用RsaCtfTool计算私钥。

典型的rsa...

密钥的产生:

选两个满足需要的大素数p和q, 计算 $n = p \times q$ ,  $\phi(n) = (p - 1) \times (q - 1)$ , 其里 $\phi(n)$ 是n的欧拉函数值。

选一个整数e, 满足 $1 < e < \phi(n)$ 且 $\text{gcd}(\phi(n), e) = 1$ 。通过 $d \times e \equiv 1 \pmod{\phi(n)}$ , 计算得出d。

以{e,n}为公开密钥, {d, n}为私钥。

加解密都为模幂运算

加密:  $c = m^e \pmod{n}$ , 即得到密文 c

解密:  $m = c^d \pmod{n}$ , 即得到明文m

## 分析

题里给了 n即由两个大素数 $p \times q$ 得到的, e, 以及密文enc, 也就是给了{e,n}公开密钥以及密文, 我们需要的目标是要获得明文。通过解密算法我们知道, 对于 $m = c^d \pmod{n}$ , m是我们想要获取的, c为密文, n也是已知的, 然而d是不知道的, 目标就是要获取d, 该题因为n和e是同数量级的, 证明d会很小。

## 解决方案

这里推荐一个工具RsaCtfTool

下面可直接利用这个工具获得 p, q和d。

```
python RsaCtfTool.py --createpub -n
```

```
python RsaCtfTool.py --publickey test.pem --private > test.key
```

```
python RsaCtfTool.py --key test.key --dumpkey
```

注意：

生成公钥或者私钥文件可能有乱码，运行出现格式错误，删除即可

```
root@kali: /media/root/RECC4F0E3CC4FC3E5/CTF/CTF_Tool/Crypto/RsaCtfTool# ./RsaCtfTool.py --createpub -n 46665781388428960986372850554417248531811792624626389974432
02374957838137059248304321738388823362826892385139401145186308780829924808811466148595875326277899645338694977974591083089776
0291939577838137059248304321738388823362826892385139401145186308780829924808811466148595875326277899645338694977974591083089776
4535925878137059248304321738388823362826892385139401145186308780829924808811466148595875326277899645338694977974591083089776
7009257810782711862119182621897232837761496754782759
private argument is not set, the private key will not be displayed, even if recovered.
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhliQwIBAQEFAACQADAM118cAKBqkP/S3TSvXgN8Set/5ekmP9
akNhxElLxLoSafFcFwMEDEHfYgWnCt18rCQ8hzKuNxqjDkceJ2eyltGkTrzMoF
IMKyUwOcyYqk6VlunWekcs5f3Bh8nW-GMS1Dj2Mu513idHViufZ221pBnEt3MT
<5fmICcpd0tbe1qfB610jXK8gQh+6QQB537nWGLxq1pAdgQzYjNKZ81z8aP0
kIBAxQF3W7ccMfPOWhzyt565c8MwAb1s107111e0Q1c56c0LQ5FgWtPx60
RMTeGhjtB8PMuP6TG/KMwv9Eh7sY2Dyv59Wv5KXgqoUldab8aKt2LNugC2
2087vowee
-----END PUBLIC KEY-----
root@kali: /media/root/RECC4F0E3CC4FC3E5/CTF/CTF_Tool/Crypto/RsaCtfTool# ./RsaCtfTool.py --publickey flag1.pem --private
[*] Testing key flag1.pem.
[*] Performing factordb attack on flag1.pem.
[*] Attack success with factordb method 1

Results for flag1.pem:

Private key :
-----BEGIN RSA PRIVATE KEY-----
MIICODAAKgBqK9/53T3vKqG5ct/SelawPPawWnxBELxLo5aFcxFmEDFWYGMbC
t18IC08hZxUNuqjDnC5132eyltGkTr7uKwFTKkVmlmQcYqoKvV3mWeke5F3F
Bh0dr9MS1Dj2Mu513idHViufZ221pBnEt3MT+SfHCcpd0tbe1qfB610jXK8gQh
+0Q085337aM6Lxq1p5aQ6QzYjNKZ81z8aP0k18aX9QFw7mtqcDp0KfHzyt565c8
BmMaB8107h1leQ01C56m0105FgWtpe6QRT0tEoIjB8PfNkxP6TG/KMwv9Eh7
H7sY2Dyv59Wv5KXgqoUldab8aKt2LNugC22089vowig5W15Wnra1ZBcJt+CW
QP1k0c9t8kYvtaXKq4u0du8a8NzLXkgcC22089vowig5W15Wnra1ZBcJt+CW
eB+GRDGlQqM0#87Av+HvYs1r8j5neheqodmby7ow2-ZY/AkEC2f/G9B8y1ZAM
Zj39k250uUlcgm1EiV2s9mR0xouj5s145k09m0X7BfJOpzSp9PmFwvTp17
t21NQXGdrf1A6xj1MKwv9Eh7sY2Dyv59Wv5KXgqoUldab8aKt2LNugC2
t21NQXGdrf1A6xj1MKwv9Eh7sY2Dyv59Wv5KXgqoUldab8aKt2LNugC2
Mhugly1tjy50L3hv/208ew27yy18nJB6214W72V8punauFv1f8TtScQXq0
-----END RSA PRIVATE KEY-----
root@kali: /media/root/RECC4F0E3CC4FC3E5/CTF/CTF_Tool/Crypto/RsaCtfTool# ./RsaCtfTool.py --publickey flag1.pem --private > flag1.key
CSDN @彬彬有礼Lam_03
```

```
root@kali: /media/root/RECC4F0E3CC4FC3E5/CTF/CTF_Tool/Crypto/RsaCtfTool# ./RsaCtfTool.py --key flag1.key --dumpkey
private argument is not set, the private key will not be displayed, even if recovered.
n: 46665781388428960986372850554417248531811792624626389974432
02992488091848612485950752627099645338694977974591685308987780072396957281019760942397169652427755227187061418202849911479124793990722597
e: 354611241307572056572181827925899198345352287537931089327546391654445626894245415096107834-6577849532373187125318594614725993017915289162128393681210660355
41008808261534505680238576771227162578520428964688046880323800124496804771053025193773709257107827116821391826210977328377614667547827619
d: 82646679722942750129339772371783321488221494719763422108239489363691895
p: 1591364697093213322076269015607493280632766403446402334810735192490663709160906409262190793688455104440314032229147771682961132420481897362843199
q: 2886579177126024594865942729284368667835441296247148207862836646578497353436182070981639817872873685697884725213463556733429935676008950745444027083
-----END RSA PRIVATE KEY-----
```

```
#!/usr/bin/python
# -*- coding=utf8 -*-
import gmpy2
from libnum import n2s

n = '46065781388428960989637205658554417248531811702624626389974432923749270182062721955600778820059011913617389
5989001382151536006853823326382892363143604314518686388786002989248800814861248595075326277099645338694977097459
168530898776007293695728101976069423971696524237755227187061418202849911479124793990722597'

e = '35461110244130757205657218182792589919834535022875373093108939327546391654445662689424541509610783446577840
9532373187125318554614722599301791528916212839368121066035541008808261534500586023652767712271625785204280964688
004680328300124849680477105302519377370092578107827116821391826210972320377614967547827619'

enc = '382309913162293996518235675906923010600446204121917377646323846805462562284515182388429652213947118483378
3245944384444688946836215418821484073674465788585894381017767587199111146665315825719113960569991634730829499566
4530280816850482740530602254559123759121106338359220242637775919026933563326069449424391192'

p = '15991846970993213322072626901560749932686325766403404864023341810735319249066370916090640926219079368845510
444031400322229147771682961132420481897362843199'

q = '28805791771260259486856902729020438686670354441296247148207862836064657849735343618207098163901787287368569
768472521344635567334299356760080507454640207003'

d = '8264667972294275017293339772371783322168822149471976834221082393409363691895'

n1 = gmpy2.mpz(n)
enc1 = gmpy2.mpz(enc)
d1 = gmpy2.mpz(d)

r = gmpy2.powmod(enc1, d1, n1)
print (r)
s = n2s(r)
print (s)
```

结果为：

```
root@kali:~# python Rsa.py
flag{Wien3r_4tt@ck_1s_3AsY}
```

flag{Wien3r\_4tt@ck\_1s\_3AsY}