攻防世界crypto高手题之RSA_gcd

原创

沐一 · 林 于 2022-01-17 10:31:47 发布 584 收藏

分类专栏： CTF 密码学 文章标签： ctf

本文链接：https://blog.csdn.net/xiao__1bai/article/details/122534237

CTF 同时被 2 个专栏收录

167 篇文章 6 订阅

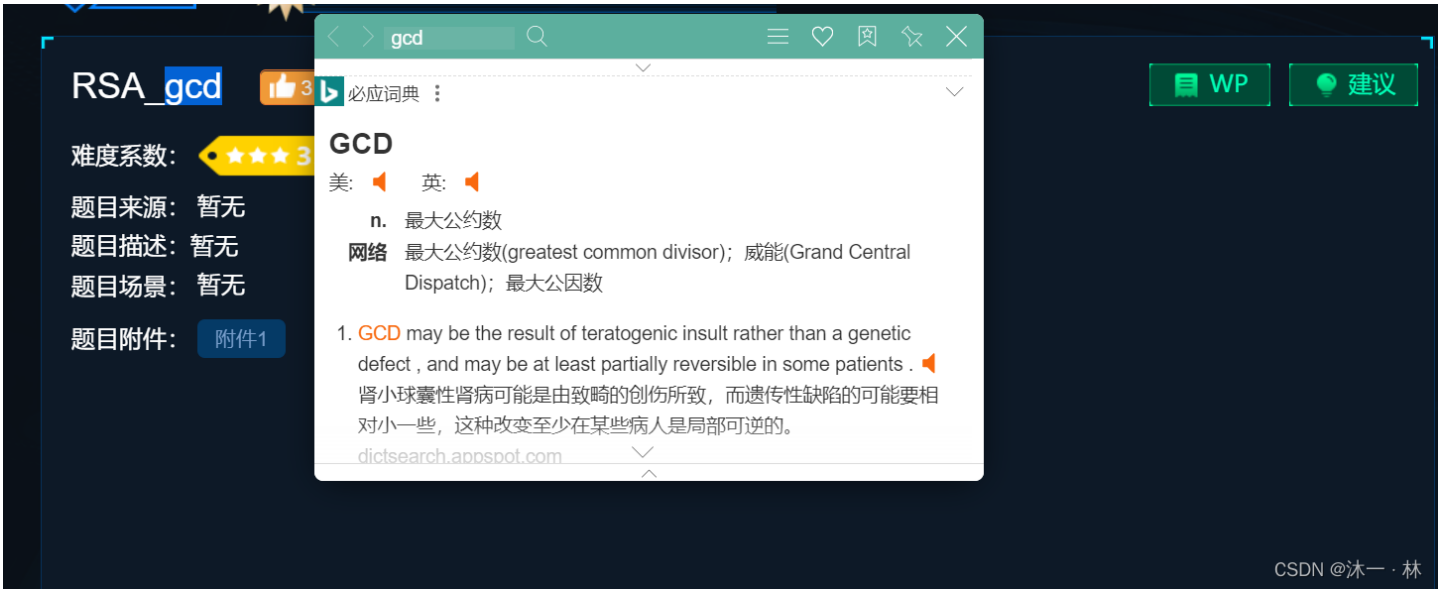
订阅专栏

密码学

51 篇文章 1 订阅

订阅专栏

攻防世界crypto高手题之RSA_gcd



.

.

最近由于在备考，所以有一段时间没刷题了，当然现在也在备考。~~只是在摸鱼而已。~~

.

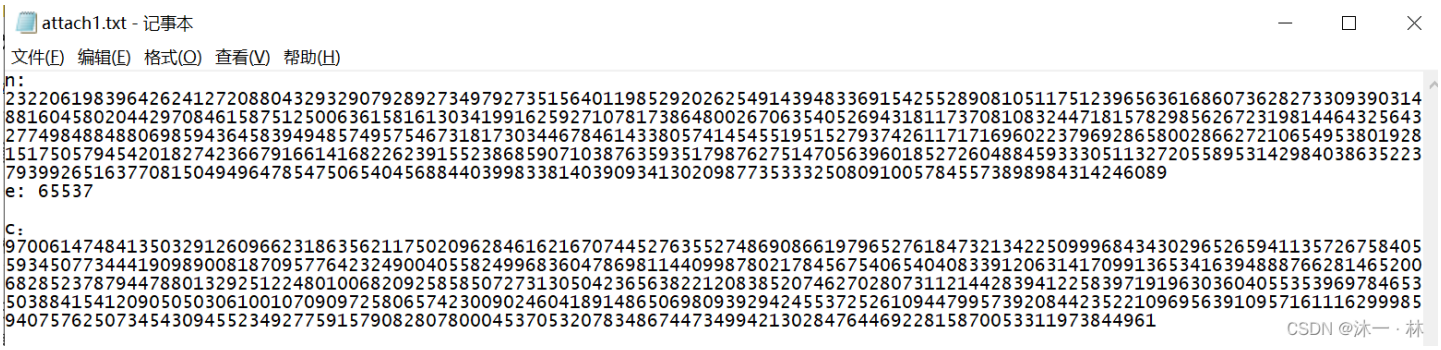
.

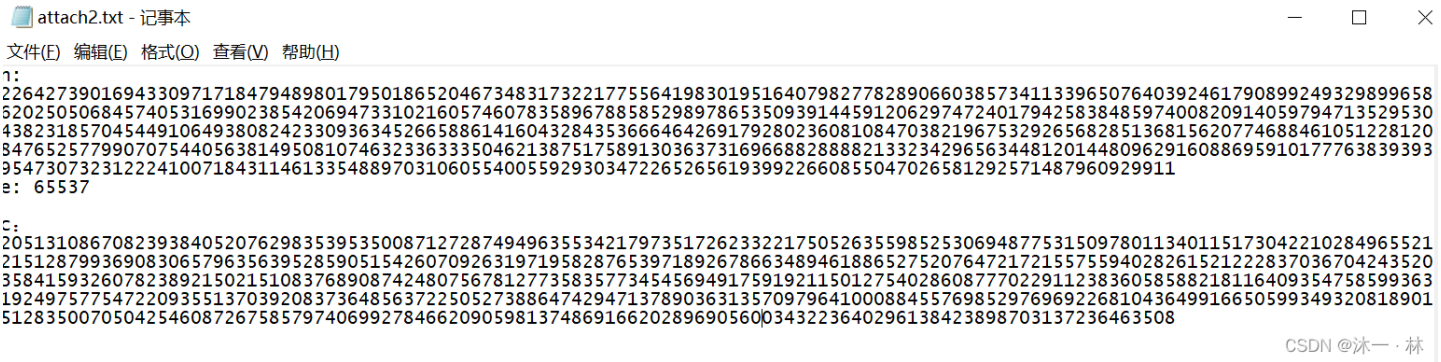
首先看看题目暗示 `GCD` 是最大公约数的意思，根据笔记猜想是给了`多个模 n`，且都是 `2048bit 4096bit` 等无法正面分解的数，需要使用欧几里得算法求取模之间的公约数。

.

.

那么照例下载附件，两个TXT，的确是多个模 N：





.

.

这个题目类型对应 `CTF-RSA-tool工具` 的一个例子

这个题目类型对应 CTF-RSA-tool工具 的一个例子：

```
n = 49717352238903813258167965634872644122363546007764600124211711250852904300466263993456901629465
e = 13720370251305502198453722303188629715020031854527043028194234587721186750392338568840741980913
c = 33995928221963087827338803441589901571528069870645123081701004401546919289265619397264329787933

n = 49717352238903813258167965634872644122363546007764600124211711250852904300466263993456901629465
e = 13521927979417175825463063347222803267672338126236919097685639546419640649137631879086526055040
c = 20663727752331650736213993392793715555002283733124450484396596147661401938360252713595785342791
```

CTF-RSA-tool工具的官方例子

.

.

那么照例摆放整齐，看看能不能直接用脚本跑出：

```
n =
 23220619839642624127208804329329079289273497927351564011985292026254914394833691542552890810511751239656361686073628273309390314881604580204429-
 70846158751250063615816130341991625927107817386480026706354057495754673181730348467846143380574145455195152793742611717169602237969286-
 86627210654953801928151750579454201827423667916614168226239155238868590710387635935179876275147056396-
 72604884593330511327205589531429840386352237939926516377081504949647854750654045688440399833814039093412-
 20987735333250809100578455738989843142460895
e = 65537
c =
 97006147484135032912609662318635621175020962846162167074452765355274869086619796527618473213422509996843430296526594113572675840559345077344419-
 98900818709577642324900405582499683604786981144099878021784567540654040833912063141709913653416394888766281465200682852378794478801329251224801-
 00682092585850727313050423656382212083852074627028073112144283941225839719196303604055353969784653503884154120905050306100107090972580657423009-
 02460418914865069809392942455372526109447995739208442352210969563910957161116299985940757625073454309455234927759157908280780004537053207834867-
 44734994213028476446922815870053311973844961
n =
 22642739016943309717184794898017950186520467348317322177556419830195164079827782890660385734113396507640392461790899249329899658620250506845740-
 53169902385420694733102160574607835896788585298978653509391445912062974724017942583848597400820914059794713529530438231857045449106493808242330-
 93634526658861416043284353666464269179280236081084703821967532926568285136815620774688461051228120847652577990707544056381495081074632336333504-
 62138751758913036373169668828888213323429656344812014480962916088695910177763839393954730732312224100718431146133548897031060554005592930347226-
 52656193992266085504702658129257148796092991 1
e = 65537
c =
 20513108670823938405207629835395350087127287494963553421797351726233221750526355985253069487753150978011340115173042210284965521215128799369083-
 06579635639528590515426070926319719582876539718926786348946188652752076472172155755940282615212222837036704243520358415932607823892150215108376-
 89087424807567812773583577345456949175919211501275402860877022911238360585882181164093547585993631924975775472209355137039208373648563722505 27-
 38864742947137890363135709796410008845576985297696922681043649916650599349320818901512835007050425460872675857974069927846620905981374869166202-
 89690560034322364029613842389870313723646350 8
```



```
┌──(wdnmd㉿kali)-[~/桌面/CTF-RSA-tool]
└─$ python2 solve.py --verbose  -i /home/wdnmd/桌面/1.txt
ERROR: wrong json file, check examples
```

.

.

说是格式错了，单独求解排查一下，发现单个又能解出来喔：（直接给我整不会了，一个N都能求,那还用什么 共因子 啊，搞得我都不知道哪里错了，感觉是不用公因子就 不要摆成 求公因子的格式吧。）

```
n =
 23220619839642624127208804329329079289273497927351564011985
 70846158751250063615816130341991625927107817386480026706354
 57495754673181730344678461433805741454551951527937426117171
 68590710387635935179876275147056396018527260488459333051132
 30209877353332508091005784557389898431424608 9
e = 65537
c =
 97006147484135032912609662318635621175020962846162167074452
 98900818709577642324900405582499683604786981144099878021784
 00682092585850727313050423656382212083852074627028073112144
 02460418914865069809392942455372526109447995739208442352210
 44734994213028476446922815870053311973844961
4
```

删掉下半部分单独求解竟然解得出来喔

```
文件  动作  编辑  查看  帮助
┌──(wdnmd㉿kali)-[~/桌面/CTF-RSA-tool]
└─$ python2 solve.py --verbose  -i /home/wdnmd/桌面/1.txt
DEBUG: factor N: try past ctf primes
DEBUG: factor N: try Gimmicky Primes method
DEBUG: factor N: try Wiener's attack
DEBUG: Starting new HTTP connection (1): www.factordb.com:80
DEBUG: http://www.factordb.com:80 "GET /index.php?query=232206198396426241272088043293290792892734971564011985292026254914394833691542552890810511751239656361686073628273309390314881604580204429708461
25006361581613034199162592710781738648002670635405269431811737081083244718157829856267231981446432564327749848848806985943645839494
86627210654953801928151750579454201827423667916614168226239155238868590710387635935179876275147056396
72604884593330511327205589531429840386352237939926516377081504949647854750654045688440399833814039093
20987735333250809100578455738989843142460895 HTTP/1.1" 200 1295
DEBUG: http://www.factordb.com:80 "GET /index.php?id=1100000001427734934 HTTP/1.1" 200 1087
DEBUG: http://www.factordb.com:80 "GET /index.php?id=1100000001494523587 HTTP/1.1" 200 1084
DEBUG: d = 0xa9e3bc39bc831bea110ea4899c749f407f9a4b00ff94699328bbdfa99b9e7196fb0c9a0ede1a512ee5ed831
b26f08b2fe2a52988d9e57b04604b96c3fa7c606e3c6dbe9aa80e2a6228129f7b0bbfea9f53c07e4c6b0f4abc051c2d9e57a
39bf36fe0af6a5d3e1a54c535078806a1a8884234e805d1876aa637e2b5afa1a690920ed0e6b755d03b3ce692c210af75dc5
3d1a06ac31f9de0e04cb354f38862b9e8ec01b1dc9e76c7e8af40e42551c02da63781f10c36373613325e7f9573338ad040e
fa26329d1b7c330dca65aa68df40e1bff7d0cebf351eb7cbdf7f478a332869dd8a275b9b30b841a6d67a1d5904eb2fc1287e
L
INFO: flag{336BB5172ADE227
```



```
n =
```

```
文件  动作  编辑  查看  帮助
```

```
226427390169433097171847948980179501865204673483173221775564
531699023854206947331021605746078358967885852989786535093914
936345266588614160432843536664642691792802360810847038219675
621387517589130363731696688288882133234296563448120144809629
526561939922660855047026581292571487960929911
e = 65537
c =
205131086708239384052076298353953500871272874949635534217973
065796356395285905154260709263197195828765397189267866348946
890874248075678127735835773454569491759192115012754028608777
388647429471378903631357097964100088455769852976969226810436
896905600343223640296138423898703137236463508
```

下半部分也是如此。

```
└─$ python2 solve.py --verbose  -i /home/wdnmd/桌面/share_N.txt
DEBUG: factor N: try past ctf primes
DEBUG: factor N: try Gimmicky Primes method
DEBUG: factor N: try Wiener's attack
DEBUG: Starting new HTTP connection (1): www.factordb.com:80
DEBUG: http://www.factordb.com:80 "GET /index.php?query=226427390169433097171847948980179501865204673
732217755641983019516407982778289066038573411339650764039246179089924932989965862025050684574053169
420694733102160574607835896788585298978653509391445912062974724017942583848597400820914059794713529
231857045449106493808242330936345266588614160432843536664642691792802360810847038219675329265682851
207746884610512281208476525779907075440563814950810746323363335046213875175891303637316966882888821
965634481201448096291608869591017776383939395473073231222410071843114613354889703106055400559293034
656193992266085504702658129257148796092991 HTTP/1.1" 200 1301
DEBUG: http://www.factordb.com:80 "GET /index.php?id=1100000001427734934 HTTP/1.1" 200 1087
DEBUG: http://www.factordb.com:80 "GET /index.php?id=1100000001504814000 HTTP/1.1" 200 1081
DEBUG: d = 0x65340b5cb6b1ebc1204012dd1a540d7ef0942e84dc28c71bd7d16c8b8f6ee106894943b26d86869c4df8f5
89352dacc604a6a6d71601b335181aa61057ddea462fed3b9fbcb10b96ef2f6295bbbc8a3bc9a2a5de11d3dd1437349263f
739b7ff795a9c7935304614f3d7ff7f4c2c38b19be57704e0b3c101957577d4dab850d61c0a9dc34b8e4139eaa402227a4d
d12481425a7b3f33f06cc32b944f4b56653d85d58eb0d58412632c4af953796d55f2cd75135cbb8acf2cb86e44719da136a
90360279d72e38a76c79066d48399baa3ccc5499bfe0181b15c4af9c09724e4108adcb86a4da62c85c52a79c78a8c6d05e2
L
INFO: FE68BAA44FDA73F3B}
```


.

.

提交完 `flag` 之后自己写一个脚本出来，这里两个 `N`，两个 `C`。一开始还不知道是一个 `N` 加密出一个 `C` 还是 `N1` 加密的 `C1` 再加密出 `C2`，毕竟以前做过这种 `多层加密` 的题。

想了一下感觉是第一种，那就附上脚本开搞：（参照的是以前做过的 `2021年9月广州羊城杯，CRYPTO的BigRsa` ）

```python
import gmpy2
def share_factor(n1, n2, e, c1,c2):
    p1 = gmpy2.gcd(n1, n2)
    q1 = n1 / p1
    q2 = n2 / p1
    d1 = gmpy2.invert(e, (p1 - 1) * (q1 - 1))
    plain = gmpy2.powmod(c1, d1, n1)
    d2 = gmpy2.invert(e, (p1 - 1) * (q2 - 1))
    plain2 = gmpy2.powmod(c2, d2, n2)
    plain = hex(plain)[2:]+hex(plain2)[2:]  #由于hex之后是字符串，所以可以直接用+来拼接。但是如果在前面用plain+plain2的话由于数字相加就乱码了。
    if len(plain) % 2 != 0:
        plain = '0' + plain
    print('Here are your plain text: \n' + plain.decode('hex'))
if __name__ == "__main__":
    n1 = 23220619839642624127208804329329079289273497927351564011985292026254914394833691542552890810511751239656361686073628273309390314881604580204429708461587512500636158161303419916259271078173864800267063540526943181173708108324471815782985626723198144643256432774984884880698594364583949485749575467318173034467846143380574145455195152793742611717169602237969286580028662721065495380192815175057945420182742366791661416822623915523868590710387635935179876275147056396018527260488459333051132720558953142984038635223793992651637708150494964785475065404568844039983381403909341302098773533325080910057845573898984314246089
    n2 = 22642739016943309717184794898017950186520467348317322177556419830195164079827782890660385734113396507640392461790899249329899658620250506845740531699023854206947331021605746078358967885852989786535093914459120629747240179425838485974008209140597947135295304382318570454491064938082423309363452665886141604328435366646426917928023608108470382196753292656828513681562077468846105122812084765257799070754405638149508107463233633350462138751758913036373169668828888213323429656344812014480962916088695910177763839393954730732312224100718431146133548897031060554005592930347226526561939922660855047026581292571487960929911
    e = 65537
    c1 = 97006147484135032912609662318635621175020962846162167074452763552748690866197965276184732134225099968434302965265941135726758405593450773444190989008187095776423249004055824996836047869811440998780217845675406540408339120631417099136534163948887662814652006828523787944788013292512248010068209258585072731305042365638221208385207462702807311214428394122583971919630360405535396978465350388415412090505030610010709097258065742300902460418914865069809392942455372526109447995739208442352210969563910957161116299985940757625073454309455234927759157908280780004537053207834867447349942130284764469228158700533119738449613401151730422102849655212151287993690830657963563952859051542607092631971958287653971892678663489461886527520764721721557559402826152122283703670424352035841593260782389215021510837689087424807567812773583577345456949175919211501275402860877702291123836058588218116409354758599363192497577547220935513703920837364856372250527388647429471378903631357097964100088455769852976969226810436499166505993493208189015128350070504254608726758579740699278466209059813748691662028969056003432236402961384238987031372364635083
    share_factor(n1, n2, e, c1,c2)
```

.

.

结果：(python2运行)

```
└─$ python2 9.py
Here are your plain text:
flag{336BB5172ADE227FE68BAA44FDA73F3B}
```

.

.

.

解毕！

敬礼！