# CTF 学习笔记一——RSA 加密

RSA 公开密钥密码体制是一种使用不同的加密密钥与解密密钥，“由已知加密密钥推导出解密密钥在计算上是不可行的”密码体制。

在公开密钥密码体制中，加密密钥（即公开密钥）PK 是公开信息，而解密密钥（即秘密密钥）SK 是需要保密的。加密算法 E 和解密算法 D 也都是公开的。虽然解密密钥 SK 是由公开密钥 PK 决定的，但却不能根据 PK 计算出 SK。

正是基于这种理论，1978 年出现了著名的 RSA 算法，它通常是先生成一对 RSA 密钥，其中之一是保密密钥，由用户保存；另一个为公开密钥，可对外公开，甚至可在网络服务器中注册。为提高保密强度，RSA 密钥至少为 500 位长。这就使加密的计算量很大。为减少计算量，在传送信息时，常采用传统加密方法与公开密钥加密方法相结合的方式，即信息采用改进的 DES 或 IDEA 对话密钥加密，然后使用 RSA 密钥加密对话密钥和信息摘要。对方收到信息后，用不同的密钥解密并可核对信息摘要。

RSA 算法的具体描述如下：

（1）任意选取两个不同的大素数 p 和 q 计算乘积$n=pq$，$\varphi(n)=(p-1)(q-1)$；

（2）任意选取一个大整数 e，满足$gcd\big(e,\varphi(n)\big)=1$，也就是$e$和$\varphi(n)$互质，整数 e 用做加密钥（注意：e 的选取是很容易的，例如，所有大于 p 和 q 的素数都可用）；

（3）确定的解密钥 d，满足$(de)mod\varphi(n)=1$，即$de=k\varphi(n)+1,k\geq 1$是一个任意的整数；所以，若知道 e 和$\varphi(n)$，则很容易计算出 d；

（4）公开整数 n 和 e，秘密保存 d；

（5）将明文 m（m<n 是一个整数）加密成密文 c，加密算法为：

$$c=E(m)=m^{e}modn$$

（6）将密文 c 解密为明文 m，解密算法为：

$$m=D(c)=c^{d}modn$$

然而只根据 n 和 e（注意：不是 p 和 q）要计算出 d 是不可能的。因此，任何人都可对明文进行加密，但只有授权用户（知道 d）才可对密文解密。

解。

在证明上述公式之前，先了解一下欧拉代数定理。也就是若 n，a 为正整数，且 n，a 互质，则:$\boldsymbol{a^{\varphi(n)} mod n = 1}$，$\boldsymbol{\varphi(n)}$为欧拉函数，表示小于 n 且和 n 互质的数的个数。特别的，当 n 为素数时，$\boldsymbol{\varphi(n) = n-1}$，当 p，q 为素数时 $\boldsymbol{\varphi(pq) = (p-1)(q-1)}$。

所以有

$$\boldsymbol{c^d mod n = (m^e mod n)^d mod n = m^{ed} mod n = m^{k\varphi(n)+1} mod n，k \in Z}$$

$$\boldsymbol{= \left((m\ mod\ n) \times \left(m^{k\varphi(n)}\ mod\ n\right)\right) mod n，k \in Z}$$

$$\boldsymbol{= \left(m \times \left(m^{\varphi(n)}\ mod\ n\right)^k mod\ n\right) mod n，k \in Z = m}$$

CTF 中，对于 RSA 题型的解题思路大体有如下几种：

## 1、 直接分解模数 N

直接分解模数 N 是最直接的攻击方法，也是最困难的方法。具体的解析同上 RSA 安全性分析。

通常可以尝试利用在线网站 factordb.com，这一类在线网站进行 n 的分解。

例题：[竞赛平台（vidar.club）]

```python
from Crypto.Util.number import *

flag = open('flag.txt', 'rb').read()

p = getPrime(512)
q = getPrime(512)
n=p*q
e = 65537
m = bytes_to_long(flag)
c = pow(m, e, n)
print(f"c={c}")
print(f"n={n}")

"""
c=110674792674017748243232351185896019660434718342001686906527789876264976328686
n=135127138348299757374196447062640858416920350098320099993115949719051354213545
"""
```

图 1  题目源码

```
c = 110674792674017748243232351185896019660434718342001686906527789876264976328
6861341019721254939384349927870029155625004754806932973608676810000927255832846
1635354342238848920811454500713860654367804079865183602743338328217708103415158
9935024292017207209056829250152219183518400364871109559825679273502274955582
n = 135127138348299757374196447062640858416920350098320099993115949719051354213
5455966432167395554539461960781108347263754759817912230694513640241819528180568
0208956706492651029412459417447812321651660036833476384920694294282471153133423
9106807454086389211139153023662266125937481669520771879355089997671125020789
```

题目只给了 n 和 c 的值，但 n 的值不是很大，可以爆破分解

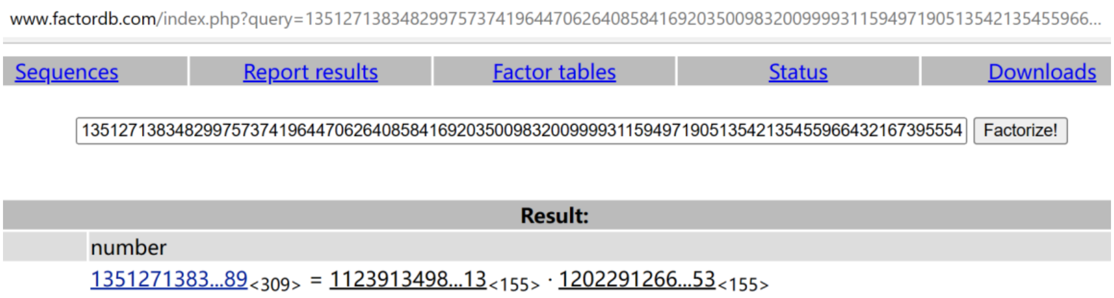


图 2  大数质因数分解

知道了 p,q 的值，那么密文很快就能求出来了，python 代码如下：

```
import gmpy2
import binascii
c = 1106747926740177482432323511858960196604347183420016869065277898762649763286861341019721254939384349927870029155625004754806932973608676810000927255832846163535434223884892081145450071386065436780407986518360274333832821770810341515899350242920172072090568292501522191835184003648711095598256792735022749555 82
n = 1351271383482997573741964470626408584169203500983200999931159497190513542135455966432167395554539461960781108347263754759817912230694513640241819528180568020895670649265102941245941744781232165166003683347638492069429428247115313342391068074540863892111391530236622661259374816695207718793550899976711250207 89
e = 65537
p = 1123913498780499358676355902818724505765255021951520176864477073386908818532074093845017881613839484432972331143354989949979577565592126166408799709729481
3
q = 1202291266142094159256975173180263937508842746343016225211308261961783701091300251545022365694283637804112216383335909791093563842346400625281426695912895
3
# invert 是求乘法逆元
d = gmpy2.invert(e,(p-1)*(q-1))
jiemi = hex(pow(c,d,n))[2:]
print(binascii.unhexlify(jiemi))
```

得到 flag 的值

b'hgame{factordb.com_is_strong!}'

## 2、 利用公约数分解 N

识别此类题目，通常会发现题目给了多个 n，均不相同，并且都是 2048bit，4096bit 级别，无法正面硬杠，并且明文都没什么联系，e 也一般取 65537。

这种题目一般可以直接 gcd(n1,n2)求出一个因数。

例题：[攻防世界 (xctf.org.cn)](xctf.org.cn)


attach2.txt - 记事本
文件 编辑 查看

n:
226427390169433097171847948980179501865204673483173221775564198301951640798277828906603857341133965076403924617908992
493298996586202505068457405316990238542069473310216057460783589678858529897865350939144591206297472401794258384859740
082091405979471352953043823185704544910649380824233093634526658861416043284353666464269179280236081084703821967532926
568285136815620774688461051228120847652577990707544056381495081074632336333504621387517589130363731696688288882133234
296563448120144809629160886959101777638393939547307323122241007184311461335488970310605540055929303472265265619399226
60855047026581292571487960929911
e: 65537

c:
205131086708239384052076298353953500871272874949635534217973517262332217505263559852530694877531509780113401151730422
102849655212151287993690830657963563952859051542607092631971958287653971892678663489461886527520764721721557559402826
152122283703670424352035841593260782389215021510837689087424807567812773583577345456949175919211501275402860877702291
123836058588218116409354758599363192497577547220935513703920837364856372250527388647429471378903631357097964100088455
769852976969226810436499166505993493208189015128350070504254608726758579740699278466209059813748691662028969056003432
23640296138423898703137236463508

attach1.txt - 记事本
文件 编辑 查看

n:
232206198396426241272088043293290792892734979273515640119852920262549143948336915425528908105117512396563616860736282
733093903148816045802044297084615875125006361581613034199162592710781738648002670635405269431811737081083244718157829
856267231981446432564327749848848806985943645839494857495754673181730344678461433805741454551951527937426117171696022
379692865800286627210654953801928151750579454201827423667916614168226239155238685907103876359351798762751470563960185
272604884593330511327205589531429840386352237939926516377081504949647854750654045688440399833814039093413020987735333
25080910057845573898984314246089
e: 65537

c:
970061474841350329126096623186356211750209628461621670744527635527486908661979652761847321342250999684343029652659411
357267584055934507734441909890081870957764232490040558249968360478698114409987802178456754065404083391206314170991365
341639488876628146520068285237879447880132925122480100682092585850727313050423656382212083852074627028073112144283941
225839719196303604055353969784653503884154120905050306100107090972580657423009024604189148650698093929424553725261094
479957392084423522109695639109571611162999859407576250734543094552349277591579082807800045370532078348674473499421302
84764469228158700533119738444961

图 3 题目描述

Python 代码如下：

```python
import gmpy2
n1 = 23220619839642624127208804329329079289273497927351564011985292026254914394
8336915425528908105117512396563616860736282733093903148816045802044297084615875
1250063615816130341991625927107817386480026706354052694318117370810832447181578
2985626723198144643256432774984884880698594364583949485749575467318173034467846
1433805741454551951527937426117171696022379692865800286627210654953801928151750
5794542018274236679166141682262391552386859071038763593517987627514705639601852
7260488459333051132720558953142984038635223793992651637708150494964785475065404
5688440399833814039093413020987735333325080910057845573898984314246089
n2 = 22642739016943309717184794898017950186520467348317322177556419830195164079
82777828906603857341133965076403924617908992493298996586202505068457405316990238
5420694733102160574607835896788585298978653509391445912062974724017942583848597
4008209140597947135295304382318570454491064938082423309363452665886141604328435
3666464269179280236081084703821967532926568285136815620774688461051228120847652
5779907075440563814950810746323363335046213875175891303637316966882888821332342
9656344812014480962916088695910177763839393954730732312224100718431146133548897
031060554005592930347226526561939922660855047026581292571487960929911
p = gmpy2.gcd(n1, n2)
print('gcd(n1, n2):n', p)
q1 = n1// p
q2 = n2// p
```

```
print('q1 is:n', q1)
print('q2 is:n', q2)
```

求出了 p 和 q 的值，那么问题也就迎刃而解了，最后的 flag 答案为：

flag{336BB5172ADE227FE68BAA44FDA73F3B}。

## 3、 构造平方数分解模数 N

识别此类题目，通常会发现 N 的两个质因数 p 和 q 挨的非常近，这个时候就可以将 N 加一个比较小的数 k，令 N+k 为平方数，从而解决此类题目。

例题：[攻防世界 (xctf.org.cn)](xctf.org.cn)

```
from Crypto.Util.number import getPrime, isPrime, getRandomNBitInteger, bytes_to
from gmpy2 import powmod,invert,gcd
from flag import flag
import sympy

q = getPrime(1024)
p = sympy.nextprime(q)
N = p * q
e = 0x10001
flag = flag.ljust(80)
m = bytes_to_long(flag)
c = pow(m,e,N)

print('N = ',N)
print('e = ',e)
print('c = ',c)

'''
N =  121944200738153928809890316115452968541452416753201303148213948434369473733
e =  65537
c =  450481133311187720953900166551639156703810999288427108953730222630439543434
'''
```

图 4 题目代码

设 $p=q+2k$，k 是一个相对比较小的数，因为 $n=pq$，所以 $n=q^2+2kq$，所以 $n+k^2=(q+k)^2$，我们只要爆破 k，使得 $n+k^2$ 是一个平方数，那么 p 和 q 就都解出来了。代码如下：

```
n = 121944200738153928809890316115452968541452416753201303148213948434369473733
3108091178717673720294067680967454313880702473945443208909679453201679724644132
5729856528664071322968428804098069997196490382286126389331179054971927655320978
2989797942453790003366357954902420275196692177844333670215782473401546477628004
0214032102265927238308754447617880202595176801542397218204540546644843155762520
1012332239774962902750073900383993300146193300485117217319794356652729502100167
6684390079250047691180701053246643791416238162568959339592113811141727785352964
09639317535751005960540737044457986793503218555306862743329296169569
def is_square(n):
    low, high, ans = 0, n, -1
    while low <= high:
        mid = (low + high) // 2
        if mid * mid <= n:
            ans = mid
            low = mid + 1
        else:
            high = mid - 1
    if ans**2 == n:
        return True
    else:
        return False
for i in range(1000):
    if is_square(n+i**2):
        print(i)
        break
```

得到 k 的值为 716，所以可以解出 pq 的值，从而 flag 就出来了，代码如下：

```
import gmpy2
import binascii
c = 450481133311187720953900166551639156703810999288427108953730222630439543434
3112574404626060854962818378560852067621253927330725244984869198505556722509058
```

```
0986600830547151466707676871205870492888610632026175072628712798192112312331980
7057453884516162980693254183220704111278633644197508735187353735020346964219899
9219863581040927505152110051313011073115724502567261524181865883874517555848163
0262402018562076262378596656072557407904040390984444521582169077523750780546158
0261306622976634371431755047207922469479855288675910366834927068284391630765221
38109478146188107069973393027348275716351796846525595128733816720 63
n = 1219442007381539288098903161154529685414524167532013031482139484343694737 33
3108091178717673720294067680967454313880702473945443208909679453201679724644132
5729856528664071322968428804098069997196490382286126389331179054971927655320978
2989797942453790003366357954902420275196692177844333670215782473401546477628004
0214032102265927238308754447617880202595176801542397218204540546644843155762520
1012332239774962902750073900383993300146193300485117217319794356652729502100167
6684390079250047691180701053246643791416238162568959339592113811141727785352964
096393175357510059605407370444579867935032185553068627433292961 69569
e = 65537
p = 1104283481440132422349070080833559748342669170272287247497303851040870252 49
3523459461649803610821785323136697674852702543264047239481539129106881181406217
1292264964439673349997269548299186629385786431155768671031746216513136081981349
352445761538320450450522403012995323086687799052976920576959270925454247 2051
q = 1104283481440132422349070080833559748342669170272287247497303851040870252 49
3523459461649803610821785323136697674852702543264047239481539129106881181406217
1292264964439673349997269548299186629385786431155768671031746216513136081981349
352445761538320450450522403012995323086687799052976920576959270925454247 0619
# invert 是求乘法逆元
d = gmpy2.invert(e,(p-1)*(q-1))
jiemi = hex(pow(c,d,n))[2:]
print(binascii.unhexlify(jiemi))
```

flag{5c9c885c361541e0b261f58b61db8cec}

## 4、 共模攻击

共模攻击，也称同模攻击。

同模攻击利用的大前提就是，RSA 体系在生成密钥的过程中使用了相同的模数 n。

在 CTF 题目中，就是同一明文，同一 n, 不同 e，进行加密。

m，n 相同；e，c 不同，且 e1 和 e2 互质

例题：[攻防世界 (xctf.org.cn)]

题目给了四个文件，先用代码看一下题目信息：

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import *
f1 = open('publickey1.pem',"rb").read()
f2 = open('publickey2.pem',"rb").read()
c1 = open('cipher1.txt',"rb").read()
c2 = open('cipher2.txt',"rb").read()
pub1 = RSA.importKey(f1)
pub2 = RSA.importKey(f2)
n1 = pub1.n
e1 = pub1.e
n2 = pub2.n
e2 = pub2.e
c1 = bytes_to_long(c1)
c2 = bytes_to_long(c2)
print("n1 =",n1)
print("e1 =",e1)
print("c1 =",c1)
print("n2 =",n2)
print("e2 =",e2)
print("c2 =",c2)
```

得到：

n1

13060424286033164731705267935214411273739909173486948413518022752

30531386223816659321477269879348776187525103042351699351971421530

68086777241046924741992151193877257419060715534378402567862204845

82884693286140537492541093086953005486704542435188521724013251087

88735140994618450129522474481962193732246914077124538008166356015

01331626921744986424745881684441675336212598246405995300528278785

58481036155222733986179487577693360697390152370901746112653758338

45608344087872600722930783003780868105030299041123866672760825345

25736969040831338660937919855651180327428932470769474807668379413
19251901579605233916076425572961
e1 = 117
c1
12847007370626420814721007824489512747227554004777043129889885590
16832730634421625318082255809846676001464087074828701652382826189
02622108836133367047681828610750143683786094142559821797696865823
65219477657474948548886794807999952780840981021935733984348055642
00311638693901400462091427384004806179606341364193675452537479095
11946172456272132193029589680182277017949877477172997529865004968
48787979475798026065928167197152995841747840050028417539459383280
73512422978995285943448074662357324106146555030300847873014089874
07459990355635991346677087534572117619698062780001264629187884577
07098665612496454640616155477050
n2
13060424286033164731705267935214411273739909173486948413518022752
30531386223816659321477269879348776187525103042351699351971421530
68086777241046924741992151193877257419060715534378402567862204845
82884693286140537492541093086953005486704542435188521724013251087
88735140994618450129522474481962193732246914077124538008166356015
01331626921744986424745881684441675336212598246405995300528278785
58481036155222733986179487577693360697390152370901746112653758338
45608344087872600722930783003780868105030299041123866672760825345
25736969040831338660937919855651180327428932470769474807668379413
19251901579605233916076425572961
e2 = 65537
c2
68308576617031565989734336170550458032770042742873009976346488004
48233655756498070693597839856021431269237565020303935757530559600

1523061543767784378325034657440846331647678649973030808521537572111723949039408632259811425028881269289820094939720760134867584608944167101228112499033224377422412696819345512374316681870061764181249344887755058165447339292419279003929248866494209436993563142782556834849983596634046112360566641497256440513009509884955491645171401590419073290626555742208696120722898496796130241964484462244068894845783105122326655711883516215855282555015469413327824464481440339970679179847191030685l9

发现n1和n2的值是相等的，那么就可以通过共模攻击解决此题，原理如

下：

共模攻击即用两个及以上的公钥(n,e)来加密同一条信息m

已知有密文：

$$m^{e1} mod n = c1$$

$$m^{e2} mod n = c2$$

条件：

当e1，e2互质，则有gcd(e1,e2)=1

根据扩展欧几里德算法，对于不完全为 0 的整数 a，b，gcd（a，b）。

那么一定存在整数 x，y 使得 gcd（a，b）=ax+by

带入本题，则得到：

$\mathbf{e1} \times \mathbf{s1} + \mathbf{e2} \times \mathbf{s2} = \mathbf{1}$，$\boldsymbol{s1}$，$\boldsymbol{s2}$为变量。

因为 e1 和 e2 为正整数，又由于 s1、s2 皆为整数，但是一正一负，此时

假设 s1 为正数, s2 为负数。

这里需要用到两条幂运算的性质：

$(\mathbf{a} \times \mathbf{b}) \ \mathbf{mod} \ \mathbf{p} = (\mathbf{a} \ \mathbf{mod} \ \mathbf{p} \times \mathbf{b} \ \mathbf{mod} \ \mathbf{p}) \ \mathbf{mod} \ \mathbf{p}$公式一

$\boldsymbol{a^b}$mod p = $((\boldsymbol{amodp})^{\boldsymbol{b}})$ mod p 公式二

因为 c1 = $\boldsymbol{m^{e1}}$mod n，c2 = $\boldsymbol{m^{e2}}$mod n，需要证明 m=$(\boldsymbol{c1^{s1}} \times \boldsymbol{c2^{s2}})$mod n

先代入公式一可得：

$(\boldsymbol{c1^{s1}} \times \boldsymbol{c2^{s2}})$mod n = $((\boldsymbol{m^{e1}} \mathbf{mod\ n})^{\boldsymbol{s1}} \times (\boldsymbol{m^{e2}} \mathbf{mod\ n})^{\boldsymbol{s2}})$mod n

=$((\boldsymbol{m^{e1 \times s1}} \mathbf{mod\ n}) \times (\boldsymbol{m^{e2 \times s2}} \mathbf{mod\ n}))$mod n　//代入公式二

=$(\boldsymbol{m^{e1 \times s1}} \times \boldsymbol{m^{e2 \times s2}})$mod n　//消掉 mod n

=$(\boldsymbol{m^{e1 \times s1 + e2 \times s2}})$mod n　　//同底数幂相乘，底数不变，指数相加

又因为 m<n，所以$(\boldsymbol{c1^{s1}} \times \boldsymbol{c2^{s2}})$mod n =m mod n=m

以下为求 s1，s2 的 python 代码：

```
e1 = 117
e2 = 65537
```

```python
def ext_gcd(a, b): #扩展欧几里得算法
    if b == 0:
        return 1, 0, a
    else:
        x, y, gcd = ext_gcd(b, a % b) #递归直至余数等于0(需多递归一层用来判
断)
        x, y = y, (x - (a // b) * y) #辗转相除法反向推导每层a、b的因子使得
gcd(a,b)=ax+by成立
        return x, y, gcd
print(ext_gcd(e1,e2))
```

得到 s1=30808，s2=-55，带入上面公式，得到 flag:

```python
import binascii
c1 = 1284700737062642081472100782448951274722755400477704312988988559016832730б
3442162531808225580984667600146408707482870165238282618902622108836133367047681
8286107501436837860941425598217976968658236521947765747494854888679480799995278
0840981021935733984348055642003116386939014004620914273840048061796063413641936
7545253747909511946172456272132193029589680182277017949877477172997529865004968
4878797947579802606592816719715299584174784005002841753945938328073512422978995
2859434480746623573241061465550303008478730140898740745999035563599134667708753
457211761969806278000126462918788457707098665612496454640616155477050
n = 1306042428603316473170526793521441127373990917348694841351802275230531386222
3816659321477269879348776187525103042351699351971421530680867772410469247419921
5119387725741906071553437840256786220484582884693286140537492541093086953005486
7045424351885217240132510878873514099461845012952247448196219373224691407712453
8008166356015013316269217449864247458816844416753362125982464059953005282787855
8481036155222733986179487577693360697390152370901746112653758338456083440878726
0072293078300378086810503029904112386667276082534525736969040831338660937919855
651180327428932470769474807668379413192519015796052339160764255729б1
c2 = 6830857661703156598973433617055045803277004274287300997634648800448233655 7
5649807069359783985602143126923756502030393575753055960015230615437677843783250
3465744084633164767864997303080852153757211172394903940863225981142502888126928
9820094939720760134867584608944167101228112499033224377422412696819345512374316
6818700617641812493448877550581654473392924192790039292488664942094369935631427
8255683484998359663404611236056664149725644051300950988495549164517140159041907
3290626555742208696120722898496796130241964484462244068894845783105122326655711
883516215855282555015469413327824464481440339970679179847191030685 19
s1 = 30808
s2 = -55
jiemi = hex(pow(pow(c1,s1,n)*pow(c2,s2,n),1,n))[2:]
print(binascii.unhexlify(jiemi))
```

flag{interesting_rsa}

## 5、 低指数攻击

加密指数指的是 e，e 一般选取 65535，当 e 很小，可直接破解。这类攻击

在 CTF 题中，一般是 e=3

如果$e=3$，且$m^e<n$，$c$开3次根式，得到$m$。

如果$e=3$，且$m^e>n$，那么设$k$，有：$c=m^e+kn$

爆破$k$，如果$c-kn$能开三次根式，得到$m$.

图 5 低指数攻击原理

例题：[BUUCTF 在线评测 (buuoj.cn)]

```
#n:
0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27
1e9e94188eaee3d5cd6f752406a43fbecb53e80836ff1e185d3ccd7782ea
986666e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ec
ecfec25c50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f302
742114073 2fedacac97b8fe50999117d27943c953f18c4ff4f8c258d8397
91e0ff5563b31a39e6374d0d41c8c46921c25e5904a817ef8e39e5c9b712
e3218fc5e5a1e8412ba16e588b3d6ac536dce39fcdfce81eec79979ea687
#e:  0x3
#c:0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa8
978347bb232d63e7289283871efab83d84ff5a7b64a94a79d34cfbd4ef12
f83f6f01492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e99523
41f32365
so,how to get the message?
```

图 6 题目描述

从题目发现，e 的值非常的小，为 3。由于$c = m^e \bmod n$，因而$m^3 = kn + c$，$k \in Z$，只要爆破 k，使得$kn + c$是一个立方数即可。Python 代码如下：

```
import gmpy2
from Crypto.Util.number import *
n = 0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c85
1e9e94188eaee3d5cd6f752406a43fbecb53e80836ff1e185d3ccd7782ea846c2e91a7b08089866
66e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ecc23580f4d2e3cc1ecfec25c
50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f3024e11461c294f29d7421140732fe
dacac97b8fe50999117d27943c953f18c4ff4f8c258d839764078d4b6ef6e8591e0ff5563b31a39
e6374d0d41c8c46921c25e5904a817ef8e39e5c9b71225a83269693e0b7e3218fc5e5a1e8412ba1
6e588b3d6ac536dce39fcdfce81eec79979ea6872793
e = 3
c = 0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978
347bb232d63e7289283871efab83d84ff5a7b64a94a79d34cfbd4ef121723ba1f663e514f83f6f0
1492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e995237985a596908adc741f32365
def de(c, e, n):
    k = 0
    while True:
        m = c + n*k
        result, flag = gmpy2.iroot(m, e)
        if True == flag:
            return result
        k += 1
m = de(c,e,n)
print(long_to_bytes(m))
```

得到 flag 的值 b’flag{25df8caf006ee5db94d48144c33b2c3b}’

## 6、 低指数广播攻击

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。

在 CTF 中，n、c 不同，明文 m，e 相同，其 e 比较小。使用中国剩余定理求解。

例题：[攻防世界 (xctf.org.cn)]()

File Edit Format Run Options Window Help

```python
#!/usr/bin/env python3.9
# -*- coding: utf-8 -*-
import gmpy2
from Crypto.Util.number import getPrime, isPrime, bytes_to_long
from secret import FLAG, E1, E2, P, Q1, Q2


def next_prime(num: int) -> int:
    num = num + 2 if num % 2 else num + 1
    while not isPrime(num):
        num += 2
    return num


p = getPrime(1024)
q = next_prime(getPrime(16) * p + 38219)
n = p * q
c = pow(E1, 65537, n)
print(f'n = {n}')
print(f'c = {c}')
# n = 160524760072475259879825463922421570617150635965496135732442802799
# c = 751639057610677013264061431434189083017589908118307247217007533393

assert E2.bit_length() == 69
ns = [getPrime(1024) * getPrime(1024) for _ in range(3)]
cs = [pow(E2, 89, n) for n in ns]
print(f'ns = {ns}')
print(f'cs = {cs}')
# ns = [158632305865006849113563847421234041202136990520180485886503920
# cs = [383309560783086294807909732325487278958657695331767109975208326

qq = getPrime(1024)
nn = P * qq
qqq = qq >> 460 << 460
print(f'nn = {nn}')
print(f'qqq = {qqq}')
# nn = 168517357977711996596259367972791585263797412986923397860494943 2
# qqq = 121042531930820997492656296084544616958724191434895945419858099

assert len(FLAG) == 42
n1 = P * Q1
n2 = P * Q2
c1 = pow(bytes_to_long(FLAG), E1, n1)
c2 = pow(bytes_to_long(FLAG), E2, n2)
print(f'n1 = {n1}')
print(f'n2 = {n2}')
print(f'c1 = {c1}')
print(f'c2 = {c2}')
# n1 = 216556178383580378955346051623587843264952514624472184851021559
# n2 = 246230163386985799674317816802000757062410143840662506603609496
# c1 = 261572234286037390583349192569246589970522937378577362211874627
# c2 = 676930175007028536623523794090427637531831917410050718485529352
```

图 7 题目描述

这道题在求 E2 的时候用到了低指数广播攻击。由于题目告知了 E2 的 89 次方分别模三个不同的数得到的不同的结果，因而可以用中国剩余定理进行解决。原理也就是先解决同余方程：

$$m^e \equiv c1(modn1)$$

$$m^e \equiv c2(modn2)$$

$$\dots\dots$$

$$m^e \equiv cx(modnx)$$

找到其中的一个解$mx$，那么就一定有$m^e = mx + k \times lcm(c1，c2，c3\dots\dots cx)$，$k \in Z$，接着按照低指数攻击的方法进行求解。

本题中给了三组 c 和 n。

代码如下：

```
import gmpy2
import sympy
from functools import reduce
def chinese_remainder(n, a):
    sum = 0
    prod = reduce(lambda a, b: a * b, n)

    for n_i, a_i in zip(n, a):
```

```
        p = prod // n_i
        sum += a_i * sympy.invert(p, n_i) * p
    return int(sum % prod)
ns=[158632305865006849113563847421234041202136990520180485886503920099275653696
8549725634468215018992313100958632364050777370699770486089868294630803102036130
2334248895233255911348365179153799197341744863134926804603973507415697810440916
3050923951803822397295508336078475240053911374744978490770975744521153793684635
4008717280090221082214368701481363136636065258321626913811678548948577243787052
8892032119729929607857459621078790511144060710035933887337208301078892163837203
4120811145101434060138923936079325969213088890589095445846196763807664854931148
14753878272881866907210235681877689493671668534251778397658670518117, 141440984
6943861935868265282850774438169729355667071768555358571966500244047625600847123
5313826051740009083510860714991201047915737216102220242621674841600987122005914
5420619636182722759868359286739203757682723909127787415026559092813909486064678
4711837764135754793147258883672633975857603827382047087963755545844624340124815
1675266602656677360819563744765522495640821496694918515669243614141704744848980
7461015697854397285851448416556659593894605126288007827427641477731504305528593
3126966762694299339210189766171987137572114324027021182126926095038094467019586
3016621594387236339317938305273510719419578308449465183, 2756382287959350393837
7821960427219022565215631856333510782568496016547757945464794632272818101891677
7052564717148052176065036521329951362557206390884245760036506282112710256481836
0063514589552846619906864009447007852641332470802857828994924128882854214320376
9199399500669311878391255837977932634772778594526940501234736059441483897017015
3247652667873999506997325183475185911679320310313202651361583044601996540088950
9527475491815377356682493144034252568874128923515388269946154952342516984626659
7156773535163599640189457171272058311480951820887261040891344076039474315985825
984444520336790670313179493074014037981261]
cs=[383309560783086294807909732325487278958657695331767109975208326194961660875
9231291050566542764984974722790226120399722937104503590740358249900089784508490
8303795316327521697779492007185670330185771846581770194049038179200244689237154
4135540467244300772352575076843089542537612467922571568738238011462810305831217
6343693900115638265002657622618744666247132114654135429040069316368839938881716
5549015930319012729929402004844604361936991755003763684567069985640646938200087
7890034435774569165287550081044714708871528958135150187601204461199097252157025
3106671158207677490849249612002954497927762168699886110455354481924, 1502420121
1772111560916342582596349777090238942787927556944737561630844311237741015128663
1698991792205202316840116721228421990727252811702467044369899023824303022111700
4372456475521502350404137469088570170885409265567084376069256924135270283335242
1331633035992391814179499802929442032042965981881756327239687796729940907885853
4330247344238986545939814263410433174351738458920078933148939437560480195199483
1647339839112698394141328178967516636452592385248135340133712522135715943787590
1723347438932596219095324562813628682905564619079367742311669369156698165093784
19892149164552548131776979706381641477878931403040942, 899220406371390849221425
6291861339175525948946919629972908439132005643626148678347198381531633907182877
```

```
1527280779583455190834066374469720793871617269672958864477916131665773912338665
8335479384212190223464483064005018113038199608308935091122403715479825929112410
4894554037604500881250119806371348673833105103600782286898276354573884788251542
2114341434767743914575878857723799901048351871046199224426138606827924703894908
0422805067112449592553602457110494411239714329949950850491789014093943889189145
3283594000764399193028606955089853654071198909973555844004685149713774167524224
1004879378991264805456815655816739588854]
res = chinese_remainder(ns,cs)
print(gmpy2.iroot(res,89))
```

得到 E2 = 561236991551738188085

## 7、 拆解 e 转化

正常的 RSA 应该有 e 与 φ(n)互素,但是有的题目并不满足，但通常会给两组关于 e 与 φ(n)的等式。这个时候就需要将 e 进行质因数分解，构造出新的 RSA 求解。

例题和 6 一样，6 中求出了 E2 的值，而 E1 可以通过爆破求解得出，代码如下：

```
from gmpy2 import invert,iroot
from Crypto.Util.number import getPrime, isPrime, bytes_to_long
def next_prime(num: int) -> int:
    num = num + 2 if num % 2 else num + 1
    while not isPrime(num):
        num += 2
    return num
n = 1605247600724752598798254639224215706171506359654961357324428027985787942008
8103766562745464838961569081446916113769517713344420113584254259000172572811154
2321073394809036722519921919974584699050644236188883360886523525408825768269883
5578315923797104377013262834479893735315093007130934797280411895281444757620706
6147031238749098842662046825743988208813903138796789940911515825517078554074496
4748191287898353096368043251326025570928477464547863870675995107693820785216916
```

```
0997032052853127047409171347704034389726990348944141006259273230240285403541543
807865668880690535049582533458453334544809133556579209189018567319042406

e = 65537
c = 7516390576106770132640614314341890830175899081183072472170075339384352294310
1585878322216791177284889301551860722928058998571101076645939698923207251231459
4917029375221335361209036112742388866873824163350886610514973038316512032459352
0531584177054060314663324403788719271747319757945798949129999366411630638983651
3478853738916237818544809027939771783197780328448074361239359161428497298143574
9362255654561121758163485884075260156288337176713756471879489767416836868661153
6931577927331427656718877923031813766208645063868208268663409075930806545214987
664210564746526523370371218812071880331087468909982085824068260101218 61

for i in range(2 ** 16,2**15,-1):
    print('\r'+str((65536-i)/32768*100)+"%",end='')
    if isPrime(i):
        q = next_prime(i * iroot(n // i, 2)[0] + 38219)
        if n % q == 0:
            print(q)
            break
p = n // q
phi = (p - 1) * (q - 1)
d = invert(e, phi)
E1 = pow(c, d, n)
print(E1)
```

得到 E1 的值 E1 = 377312346502536339265

接着通过观察，发现 n1 和 n2 有共同的因数 P，所以可以辗转相除法求出

P。

Python 代码如下：

```
import gmpy2
n1 = 21655617838358037895534605162358784326495251462447218485102155997156394132
443891540203860915433559917314267455046844360743623050975083617915806922096697 3
0460387813429596465043039337522579278180472629246092370889072282743655220901636
8047420993613497196059326374616217655625810171080545267058266278112647715784756
4338958097579170704018956131689101668125665455934053629534878078405394253831233
6984274182126052300520847936148489176271474972168383475460159679670766971808434
```

```
3845276793153649005628590896279281956588607062999398889314240295073524688108299345609307659091936270255367762936542565961639163236594456862919813549
n2 = 24623016338698579967431781680200075706241014384066250660360949684385831604822817314457973559632215801205780786144608311361063622813017396858888436529116737754653067203843306015767091585697803364656624926853551997229897087731298797904208292585562517602132663331748784390752958757661484560335406769204491939879324079089140420467301773366050084810282369044622442784113688062220370531522036512803461607049619641336524486507388232280683726065679295742456158606213294533956580462863488082028563360006966912264908424680686577344549034033470952036766850596897062924137344079889301948258438680545785139118107899367307031396309
p = gmpy2.gcd(n1, n2)
print('gcd(n1, n2):\n', p)
q1 = n1// p
q2 = n2// p
print('q1 is:\n', q1)
print('q2 is:\n', q2)
```

## 得到结果：

```
gcd(n1, n2):

13922160689271116331186150216572077968504099114623681977107731147326651993194760578257190002796305588677308609145272452766473815939878249467782426851561675469574980525326061635234831170249776259344985568675527862394653437170150947836869132073518219409311180128931469597871185033476336585646820347139844842399
q1 is:

155547822796260230301163493572328276759754179923840369685187766807125087369928975444183346813644731672051277851645460274588975060909127179689654378234675963719955065971621191295992778117297548246126322013897580863954772277036417493420548297592968260640347856809237210752134250598888189729496548294692404268851
q2 is:
176862032325728733112232812799746628539676281475101885729648695725069625447450093015182103211896449720721008446169371711893103335768209077906398522734998474760674095210567056451011572994691734256964295917711714584736577060817163508416373914379207885134617634676468095190710307036248746843930480925386278062091
```

接着就是本题的题型了，这道题目中 q1-1 和 q2-1 都是 5 的倍数，p-1 是

7 的倍数，而恰巧 e1 和 e2 都是 35 的倍数，不满足 e1，e2 和(q1-1)和

( p-1)，(q2-1)和( p-1)互质，原来的方法失效。所以我们要尝试进行转化，构造一个新的 RSA。

我们先假设$\mathbf{gcd}\big(\boldsymbol{e},\boldsymbol{\varphi}(\boldsymbol{n})\big)=\boldsymbol{a}$，那么就可以将 e 写成$\boldsymbol{e}=\boldsymbol{E}\times\boldsymbol{a}$的形式。再构造 d 使得$\boldsymbol{dEmod\varphi}(\boldsymbol{n})=\mathbf{1}$。由于 RSA 的加密算法为$\boldsymbol{m^e modn}=\boldsymbol{c}$，解密算法为$\boldsymbol{c^d modn}=\boldsymbol{m}$其中$\boldsymbol{demod\varphi}(\boldsymbol{n})=\mathbf{1}$。

那么变形之后可得到：

$\boldsymbol{c^d modn}=(\boldsymbol{m^e modn})^{\boldsymbol{d}}\boldsymbol{modn}=\boldsymbol{m^{ed} modn}=\boldsymbol{m^{aEd} modn}=\boldsymbol{m^{Ed^a} modn}=(\boldsymbol{m^{Ed} modn})^{\boldsymbol{a}}\boldsymbol{modn}=(\boldsymbol{m^{k\varphi(n)+1} modn})^{\boldsymbol{a}}\boldsymbol{modn}$，$\boldsymbol{k}\in\boldsymbol{Z}=\Big(\big(\boldsymbol{m^{k\varphi(n)} modn}\big)*(\boldsymbol{mmodn})\Big)^{\boldsymbol{a}}\boldsymbol{modn}$，$\boldsymbol{k}\in\boldsymbol{Z}=\boldsymbol{m^a modn}$。

带入本题，可以知道$\boldsymbol{a}=\mathbf{35}$，$\boldsymbol{c_1}^{\boldsymbol{d_1}}\boldsymbol{modn_1}=\boldsymbol{m^a modn_1}$，$\boldsymbol{c_2}^{\boldsymbol{d_2}}\boldsymbol{modn_2}=\boldsymbol{m^a modn_2}$。其中$\boldsymbol{d_1E_1mod\varphi}(\boldsymbol{n_1})=\mathbf{1}$，$\boldsymbol{d_2E_2mod\varphi}(\boldsymbol{n_2})=\mathbf{1}$，$\boldsymbol{e_1}=\boldsymbol{aE_1}$，$\boldsymbol{e_2}=\boldsymbol{aE_2}$。

再根据公式若$\boldsymbol{n}=\boldsymbol{pq}$，$\boldsymbol{wmodn}=\boldsymbol{c}$则$\boldsymbol{wmodp}=\boldsymbol{cmodp}$，$\boldsymbol{wmodq}=\boldsymbol{cmodq}$。

因而代入本题，有${c_1}^{d_1} mod q_1 = (m^a mod q_1)$，${c_2}^{d_2} mod q_2 = (m^a mod q_2)$。

为了方便，这里设${c_1}^{d_1} mod q_1 = w_1$，${c_2}^{d_2} mod q_2 = w_2$，那么有 $m^a mod q_1 = w_1$，$m^a mod q_2 = w_2$，根据中国剩余定理，可以很快求出一个数 k，使得$k mod q_1 = w_1$，$k mod q_2 = w_2$，$m^a mod q_1 q_2 = k$，于是一个新的 RSA 就构造好了。由于在本题中$q_1 - 1$，$q_2 - 1$都是 5 的倍数，而 a 也是 5 的倍数，并没有满足互质的前提，因而需要变形一下。由于$a = 35 = 5 \times 7$。所以${m^5}^7 mod q_1 q_2 = k$，7 和$q_1 - 1$，$q_2 - 1$是互质的，满足 RSA 的构造条件，因而可以求出$m^5$的值，最后开五次方就可以得到答案了。Python 代码如下：

```
from Crypto.Util.number import long_to_bytes
import gmpy2
import sympy
from functools import reduce
n1 = 21655617838358037895534605162358784326495251462447218485102155997156394132
44389154020386091543355991731426745504684436074362305097508361791580692209669730
46038781342959646504303933752257927818047262924609237088907228274365522090163
68047420993613497196059326374616217655625810171080545267058266278112647715784756
43389580975791707040189561316891016681256654559340536295348780784053942538312336
98427418212605230052084793614848917627147497216838347546015967967076697180843438
45276793153649005628590896279281956588607062999398889314240295073524688108299345
609307659091936270255367762936542565961639163236594456862919813549
n2 = 24623016338698579967431781680200075706241014384066250660360949684385831604
822817314457973559632215801205780786144608311361063622813017396858884365291167
```

```
37754653067203843306015767091585697803364656624926853551997229897087731298797904208292585562517602132663331748784390752958757661484560335406769204491939879324079089140420467301773366050084810282369044622442784113688062220370531522036512803461607049619641336524486507388232280683726065679295742456158606213294533956580462863488082028563360006966912264908424680686577344549034033470952036766850596897062924137344079889301948258438680545785139118107899367307031396309
c1 = 2615722342860373905833491925692465899705229373785773622118746270300793647098821993550686581418882518204094299812033719020077509270290007615866572202192731169538843513634106977827187688709725198643481375562114294032637211892276591506759075653224150064709644522873824736707734614347484224826380423111005274801291329132431269949575630918992520949095837680436317128676927389692790957195674310219740918585437793016218702207192925330821165126647260859644876583452851011163136097317885847756944279214149072452930036614703451352331567857453770020626414948005358547089607480508274005888648569717750523094342973767148059329557
c2 = 6769301750070285366235237940904276375318319174100507184855293529277737253672792851212185236735819718282816927603167670154115730023644681563602020732801002035524276894497009910595468459369997765552682404281557968383413458466181053253824257764740656801662020120125474240770889092605770532420770257017137747744565202144183642972714927894809373657977142884508230107940618969817885214454558667008383628769508472963039551067432579488899853537410634175220583489733111861415444811663313479382343954977022383996370428051605169520337142916079300674356082855978456798812661535740008277913769809112114364617214398154457094899399
E1 = 377312346502536339265
E2 = 561236991551738188085
P = gmpy2.gcd(n1,n2)
Q2 = n2//P
Q1 = n1//P
c = [pow(c1, gmpy2.invert(E1 // 35, (P - 1) * (Q1 - 1)), n1),
     pow(c2, gmpy2.invert(E2 // 35, (P - 1) * (Q2 - 1)), n2)]
w2 = c[1] %Q2
w1 = c[0] %Q1
def chinese_remainder(n, a):
    sum = 0
    prod = reduce(lambda a, b: a * b, n)
    for n_i, a_i in zip(n, a):
        p = prod // n_i
        sum += a_i * sympy.invert(p, n_i) * p
    return int(sum % prod)
result = chinese_remainder([Q1,Q2],[w1,w2])
d = gmpy2.invert(7,(Q1-1)*(Q2-1))
n = Q1*Q2
m = pow(result,d,n)
flag = gmpy2.iroot(m,5)[0]
print(long_to_bytes(flag))
```

得到 flag 的值 flag{27dab675-9e9b-4c1f-99ab-dd9fe49c190a}

## 8、 多素数分解

这类题目的 n 并不是两个大素数相乘，往往是多个素数的乘积，但解法是类似的。

例题：[攻防世界 (xctf.org.cn)](xctf.org.cn)

```python
import libnum
from Crypto.Util import number
from functools import reduce
from secret import flag

n = 5
size = 64
while True:
    ps = [number.getPrime(size) for _ in range(n)]
    if len(set(ps)) == n:
        break

e = 65537
n = reduce(lambda x, y: x*y, ps)
m = libnum.s2n(flag)
c = pow(m, e, n)

print('n = %d' % n)
print('c = %d' % c)
```

图 8 题目描述

本题的 n 就是 5 个大素数的乘积，先将其进行分解。

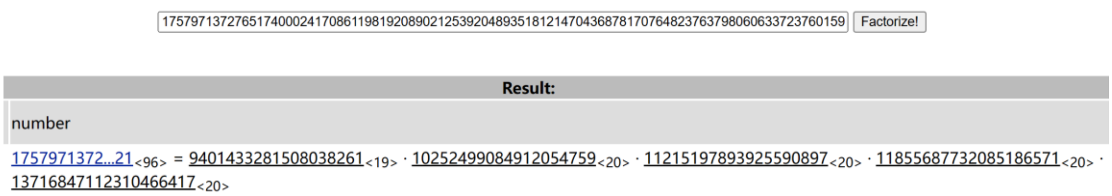


图 9 分解结果图

接着欧拉函数就要相应的进行变化，两个素数的时候，欧拉函数为$\varphi(n) = (p-1)(q-1)$；相应的，多个素数时，欧拉函数就是每个素数减一再乘起来，代码如下：

```
import gmpy2
import binascii
c = 1440092217811723536363399888969109120477262607591088472575660194123820838535987358178699332021 68
n = 1757971372765174000241708611981920890212539204893518121470436878170764823763798060633723760159 21
E = 0x10001
data=[9401433281508038261,10252499084912054759,11215197893925590897,11855687732085186571,13716847112310466417]
phi = 1
for p in data:
    phi = phi * (p-1)
# invert 是求乘法逆元
d = gmpy2.invert(E,phi)
jiemi = hex(pow(c,d,n))[2:]
print(binascii.unhexlify(jiemi))
```

得到 flag：HSCTF{@Tv0_br3ad5_c1ip_cHe3se_!@}

本文地址：[TLearning（caodong0225.github.io）]