




BUUCTF RSA (一)

原创

路由()生  于 2021-08-06 15:26:40 发布  684  收藏 7

分类专栏: [crypto](#)

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/qq_52193383/article/details/119428246

版权



[crypto](#) 专栏收录该内容

35 篇文章 3 订阅

订阅专栏

这里写目录标题

- 1.RSA
- 2.rsarsa
- 3.RSA1
- 4.RSA2
- 5.RSA3
- 6.RSA
- 7.RSAROLL
- 8.Dangerous RSA
9. [HDCTF2019]basic rsa
- 10.[GUET-CTF2019]BabyRSA

1.RSA

在一次RSA密钥对生成中, 假设 $p=473398607161$, $q=4511491$, $e=17$, 求解出 d 作为flag提交.

```
import gmpy2
p = 473398607161
q = 4511491
e = 17
d = int(gmpy2.invert(e, (p-1)*(q-1)))
print(d)
```

2.rsarsa

```

import gmpy2
p = 964842302901051567659055174001042653494573763923573980064398935203985250729849139956103500916342705037010757
0733633350911691280297777160200625281665378483
q = 118748438379802970320924058486536568527609101545433809076500401907042833589092085782510630477324439922306479
03887510065547947313543299303261986053486569407
e = 65537
c = 832082989951746041747735902982036393605400248712561268928896613457424033149298619391004926666056473166465764
8652621745700637684228086972858172674640158370589994176821413874225968933484073563355305388764184765117377625182
0293087212885670180367406807406765923638973161375817392737747832762751690104423869019034
n = p*q
f = (p-1)*(q-1)
d = int(gmpy2.invert(e,f))
m = pow(c,d,n) #pow(c,d,n)的内部运算
print('明文:',m)

```

pow(c,d,n)的底层实现原理

3.RSA1

已知p,q,dp,dq,c求明文:

首先有如下公式:

$dp \equiv d \pmod{p-1}$, $dq \equiv d \pmod{q-1}$, $m \equiv c^d \pmod{n}$, $n=pq$

因为 $m \equiv c^d \pmod{n}$, 推出 $m = c^d + kn = c^d + kpq$, 分别取余 p 和 q , 得 $m_1 \equiv c^d \pmod{p}$, $m_2 \equiv c^d \pmod{q}$, 即 $c^d = m_1 + tp$ (或 $c^d = m_2 + tq$), 代入 $m_2 \equiv c^d \pmod{q}$ 得 $m_2 \equiv (m_1 + tp) \pmod{q} \implies m_2 - m_1 \equiv t * p \pmod{q} \implies (m_2 - m_1) * p^{-1} \equiv t \pmod{q}$, (p^{-1} 为 p 的逆元), 从而 $t = (m_2 - m_1) p^{-1} \pmod{q}$, 有因为 $c^d = m_1 + tp$, 所以 $c^d = m_1 + ((m_2 - m_1) p^{-1} \pmod{q}) * p$, 又 $m \equiv c^d \pmod{n} \implies m = (m_1 + ((m_2 - m_1) p^{-1} \pmod{q}) * p) \pmod{n}$. 而 $m_1 \equiv c^d \pmod{p}$, $m_2 \equiv c^d \pmod{q}$, $dp \equiv d \pmod{p-1}$, $dq \equiv d \pmod{q-1}$, 从而 $m_1 \equiv c^{dp+k_1(p-1)} \pmod{p}$, $m_2 \equiv c^{dq+k_2(q-1)} \pmod{q}$, 由费马小定理可知 $c^{p-1} \equiv 1 \pmod{p}$, $c^{q-1} \equiv 1 \pmod{q}$, 所以 $m_1 \equiv c^{dp} \pmod{p}$, $m_2 \equiv c^{dq} \pmod{q}$.

因为 $(p,q)=1$, 则存在 s,t , 使得 $sp+ tq=1$, 即 $sp \equiv 1 \pmod{q}$, 又 $p^{-1} * p \equiv 1 \pmod{q}$, 即 $p^{-1} * p \equiv 1 \pmod{q}$, 所以 $s=p^{-1}$, 可以用扩展欧几里得算法求解

```

import libnum

p = 863763376725700856709965348654109117132049150943361544753916243791124417588566780639841179052408355344515811
3502227745206205327690939504032994699902053229
q = 126406749739964727691760479371708834209270508214800105815931371353724738805956137373376306297525773461470392
84030082593490776630572584959954205336880228469
dp = 65007957022168346211090423511932615306500438410562529309309496633586250168818328407280660261502646930761093
54874099841380454881716097778307268116910582929
dq = 78347226367355344901953258038647067238057403355130388913791176043888168367455609809825679567351220196300217
5438762767516968043599582527539160811120550041
c = 247223054038873820735673164676490806626315529059602293990791079956021544181760563358006388875276141640735304
3765708507967615735020535194522298935131607648657359957604197833987226592506276431853608900731027027852615967893
7431903862892400747915525118983959970607934142974736675784325993445942031372107342103852

n = p*q

def ext_euclid(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = ext_euclid(b, a % b) # q = gcd(a, b) = gcd(b, a%b)
        x, y = y, (x - (a // b) * y)
        return x, y, q

def mod_inv(a, b):
    return ext_euclid(a, b)[0] % b

pn = mod_inv(p,q)
m1 = pow(c,dp,p)
m2 = pow(c,dq,q)
m = (((m1-m2)*pn%q)*p+m1)%n
print(libnum.n2s(m)) #数字转字符串

```

4.RSA2

已知 $e,n,dp/(dq),c$ 求明文:

首先有如下公式:

$$dp \equiv d \pmod{p-1}, \quad ed \equiv 1 \pmod{\varphi(n)}, \quad n=pq, \quad \varphi(n)=(p-1)(q-1)$$

$$\therefore e \cdot d \equiv 1 \pmod{\varphi(n)}$$

$$\therefore d \equiv e^{-1} \pmod{\varphi(n)}$$

$$\text{即 } d = e^{-1} + k\varphi(n) \quad k \in \mathbb{Z}$$

$$\text{任取 } d_p \equiv d \pmod{p-1}$$

$$\begin{aligned} \text{得 } d_p &= e^{-1} + k\varphi(n) + t(p-1) \\ &= e^{-1} + [k\varphi(n) + t](p-1) \end{aligned}$$

$$\therefore d_p \equiv e^{-1} \pmod{p-1}$$

$$\Rightarrow e \cdot d_p \equiv 1 \pmod{p-1}$$

$$\Rightarrow (p-1) \mid (e \cdot d_p - 1)$$

$$\therefore (e, \varphi(n)) = 1$$

$$\therefore (e, p-1) = 1$$

\therefore 存在唯一整数 $e' \in [1, p-1]$, 使得 $e \cdot e' \equiv 1 \pmod{p-1}$

$$\text{即 } ~~d_p = e' \in [1, p-1]~~$$

$$d_p = e' < p-1$$

~~由~~ 由 $(p-1) \mid (e \cdot d_p - 1)$ 得 $(p-1) \cdot x = e \cdot d_p - 1 \approx e \cdot d_p \quad (x \in \mathbb{Z})$

$$\therefore d_p < p-1$$

$$\therefore x < e$$

$$\therefore x \in [1, e)$$

所以根据 x 的范围求出 $p-1$, 从而求出 $p, \varphi, \varphi(n), d$, 最后求出明文 m .

在求 p 时, p 满足两个约束条件:

$$\textcircled{1} (e \cdot d_p - 1) \% (p-1) = 0$$

$$\textcircled{2} n \% p = 0$$

```

import libnum
e = 65537
n=24825400785152624117772152669890180298583276617622160961225887737162058006043310153832803030521991869764361981
420093067961210988553380133534844502375167047843707305544724280684733298051599167660303645183146161497485358633
681492129668802402065797789905550489547645118787266601929429724133167768465309665906113
dp=9050744980523469046430251328795183306919251745730540046218772533186826750554219709435520166955285603648344463
03196939207056642927148093290374440210503657
c=14042367097625269680753367358620940057566428210068411978420352712452118899640382659743688376604187906749428095
7410201958935737360380801845453829293997433414188838725751796261702622028587211560353362847191060306578510511380
965162133472698713063592621028959167072781482562673683090590521214218071160287665180751
pd = e*dp-1

def ext_euclid(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = ext_euclid(b, a % b)
        x, y = y, (x - (a // b) * y)
        return x, y, q

def mod_inv(a, b):
    return ext_euclid(a, b)[0] % b #函数返回的第一个数%b

for i in range(1,e):
    if pd%i == 0:
        if n%(pd//i+1) == 0:
            p = pd//i+1
            q = n//p
            fn = (p-1)*(q-1)
            d = mod_inv(e,fn)
            m = pow(c,d,n)
            print(libnum.n2s(m))

```

5.RSA3

共模攻击

前提：有两组及以上的RSA加密过程，而且其中两次的m和n都是相同的，那么就可以在不计算出d而直接计算出m的值。

设模数为n，两个用户的公钥分别为e1和e2，且e1和e2互素，明文为m，密文分别为c1和c2。在已知e1,e2,n,c1,c2的情况下，推理过程如下：

$$c1 \equiv m^{e1} \pmod{n}, c2 \equiv m^{e2} \pmod{n}$$

从而 $c1 * c2 \equiv m^{(e1+e2)} \pmod{n}$ ，这里求出的m需要开根号，并不好。注意到e1和e2互素，即 $(e1,e2)=1$ ，所以存在整数s,t，使得 $s * e1+t * e2=1$ ，这样就可以把m的次方化为1，利用扩展欧几里得算法求出s,t，由此可得 $c1^s * c2^t \equiv m^{(s * e1+t * e2)} \pmod{n} \equiv m \pmod{n}$ ，推出 $m \equiv c1^s * c2^t \pmod{n}$

```
#共模攻击
```

```
import libnum
e1 = 11187289
c1=2232203527566323704164689377045193350932470191348430333807621060354261275895626286964082248647012114942448557
1361007421293675516338822195280313794991136048140918842471219840263536338886250492682739436410013436651161720725
8554848666900847887213495556620198790815011132229961233055330093259643777988927031615218528059568112195638833128
9633015629862167468435391954755812792092570684280891476219901105495581653497767526739500957534782038707348392842
5066536361482774892370969520740304287456555508933372782327506569010772537497541764311429052216291198932092617792
645253901478910801592878203564861118912045464959832566051361
e2 = 9647291
c2=1870201004518701555654869164239498283566926214723021273130993867522645855521042597242941844927341053538798593
1036711854265623905066805665751803269106880746769003478900791099590239513925449748814075904017471585572848473556
4905654500626647064491284158347879619472662597897859629222387011340797204142284140661930714953046123410529874556
1593002353682380149926977335718608745274750084064041936501155442118303750565346128673274098370274082267114804561
9497667184586123657285604061875653909567822328914065337797733444640351518775487649819978262363617265797982843179
630888729407238496650987720428708217115257989007867331698397
n=22708078815885011462462049064339185898712439277226831073457888403129378547350292420267016551819052430779004755
8466490440010241414852832864831307026160572746984736111495087988697063475019315831176327107007872280164801276773
9364992953041659868602735421642256593445901516192761360790283154285797785961259628235367932777330372700440726219
7231586324599181983572622404590354084541788062262164510140605868122410388090174420147752408554129789760902300898
0462739090078528184740307706996476473630151021189567376739413542176926960449696953085064365731425655734875835070
37356944848039864382339216266670673567488871508925311154801

def ext_euclid(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = ext_euclid(b, a % b) # q = gcd(a, b) = gcd(b, a%b)
        x, y = y, (x - (a // b) * y)
        return x, y, q
r,s,q = ext_euclid(e1,e2)
m = (pow(c1,r,n)*pow(c2,s,n))%n
print(libnum.n2s(m))
```

6.RSA

证书公钥解析

RSA公钥文件解密密文的原理分析

首先得到两个文件，将pub.key文件的内容进行公钥解析

[公钥解析网站](#)

得到e和n,对n进行分解因数求p,q，使用yafu工具或网站[factordb](#)

代码：


```

import rsa
import gmpy2
c = 1854183526100811878807183372982532818560316522978821358738967769534081571682
p = 285960468890451637935629440372639283459
q = 304008741604601924494328155975272418463
e = 65537
n = 86934482296048119190666062003494800588905656017203025617216654058378322103517
fn = (p-1)*(q-1)

d = int(gmpy2.invert(e,fn))
key = rsa.PrivateKey(n,e,d,q,p)
with open(r'c:\111\flag.enc','rb') as f: #文件路径
    f = f.read()
    print(rsa.decrypt(f,key))

```

7.RSAROLL

给出两个文件，data.txt文件中第一行的数据的形式有没有像公钥(e,n)的形式，猜测两个数据为e和n。下面的数据应该是密文。

```

data.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
{920139713,19}

704796792
752211152
274704164
18414022
368270835
483295235
263072905
459788476
483295235
459788476
663551792
475206804

```

但我在写的过程中把下面的数据整合到一块了，结果解出个b'\x05E\xba!'

.....猜错了！根据每个数据的位数（8位或9位）和n的位数（9位）几乎一样，可能是每一行数据进行解密最后整合到一块。第一、二、三行数据分别解出：

b'f', b'l', b'a'

说明猜想是对的。

代码：

```

import libnum, gmpy2

e = 19
n = 920139713
p = 18443
q = 49891
d = int(gmpy2.invert(e, (p-1)*(q-1)))

f = open(r'c:\111\data.txt', 'rb')
next(f) #跳过文件中的第一行
next(f) #跳过文件中的第二行
for i in f: #行读取
    m = pow(int(i), d, n)
    print(libnum.n2s(m).decode(), end = '') #Libnum解出的是bytes类型, 转换成字符串类型用.decode('utf-8')

```

8. Dangerous RSA

将文件中的16进制数据转成10进制数据（n的16进制数后面有个'L'是数据类型标识），这里的 $e = 3$, $\text{len}(c) = 274$, $\text{len}(n) = 617$, e 和 c 都很小, n 很大，应该属于小指数明文爆破攻击。

小指数明文爆破攻击：

当 e 和 m 都很小，而 n 很大时，会出现两种情况。

1. $m^e < n$ ，此时 $c = m^e$ ，直接开 e 次根号即可得到 m 。

2. $m^e > n$ 但并没有超过 n 太多，且 k 是可以爆破的大小时，根据 $c \equiv m^e \pmod{n}$ ，有 $m^e = c + k * n$ ，通过列举 k ，对 $c + k * n$ 开 e 次根，找到满足条件的 m 。

```

import libnum
import gmpy2

n = 104563359048381699143496468528300829321521306245331798554377007299864309163599109680353713551281520167500752
7116112974497925460592299103075361657084921193198911294127712168236379071064650334552050593141835021909803656993
2546893477983585713668853372819392130314661542626399462820378837771792293945490048339575869129479058678981790418
1128874032927217756445335407694678895127733824948782915742621427551863745702458136953906647022622262816014568891
7619192027065881175369608108252853926319147719223633611214770556479643580015261436611385434961510419105280764712
8834113146535639155857819697492608839941056356828288393881491
e = 3
c = 221734475079829609119323039422158289465790964317493441684258833587129815259836870148402883240728974621838778
3855373449002121088413603751014125921242419602155087438902181522441026460003722677539409576093794862185483713606
547386172606576925933695952279401957552813065318376293
k = 0

while True:
    pd = gmpy2.iroot(c + k*n, e) #大整数开根
    if pd[1] == True:
        m = int(pd[0])
        print(libnum.n2s(m))
        break
    else:
        k+=1

```

9. [HDCTF2019]basic rsa

代码里已经给出了 e, n, p, q, c ，没什么好说的，直接上代码：


```

import gmpy2
import libnum

p = 262248800182277040650192055439906580479
q = 262854994239322828547925595487519915551
e = 65533
n = p*q
f = (p-1)*(q-1)
c = 27565231154623519221597938803435789010285480123476977081867877272451638645710

d = int(gmpy2.invert(e,f))
m = pow(c,d,n)
print(libnum.n2s(m))

```

10.[GUET-CTF2019]BabyRSA

文件中给出了 $p+q$ 和 $(p+1)*(q+1)$ ，根据
 $\varphi(n) = (p-1)*(q-1) = (p+1)*(q+1) - 2*(p+q)$
 $n = p*q = (p+1)*(q+1) - (p+q) - 1$
 求出 $\varphi(n)$ ， n 后，直接上代码

```

import libnum
import gmpy2

e = int('0xe6b1bee47bd63f615c7d0a43c529d219',16)
#a = p+q    b = (p+1)*(q+1)
a = int('0x1232fecb92adead91613e7d9ae5e36fe6bb765317d6ed38ad890b4073539a6231a6620584cea5730b5af83a3e80cf30141282c97be4400e33307573af6b25e2ea',16)
b = int('0x5248becf1d925d45705a7302700d6a0ffe5877fddf9451a9c1181c4d82365806085fd86fbaab08b6fc66a967b2566d743c626547203b34ea3fdb1bc06dd3bb765fd8b919e3bd2cb15bc175c9498f9d9a0e216c2dde64d81255fa4c05a1ee619fc1fc505285a239e7bc655ec6605d9693078b800ee80931a7a0c84f33c851740',16)
n = b-a-1
f = b-2*a
d = int('0x2dde7fbaed477f6d62838d55b0d0964868cf6efb2c282a5f13e6008ce7317a24cb57aec49ef0d738919f47cdcd9677cd52ac2293ec5938aa198f962678b5cd0da344453f521a69b2ac03647cdd8339f4e38cec452d54e60698833d67f9315c02ddaa4c79ebaa902c605d7bda32ce970541b2d9a17d62b52df813b2fb0c5ab1a5',16)
c = int('0x50ae00623211ba6089ddfcae21e204ab616f6c9d294e913550af3d66e85d0c0693ed53ed55c46d8cca1d7c2ad44839030df26b70f22a8567171a759b76fe5f07b3c5a6ec89117ed0a36c0950956b9cde880c575737f779143f921d745ac3bb0e379c05d9a3cc6bf0bea8aa91e4d5e752c7eb46b2e023edbc07d24a7c460a34a9a',16)

m = pow(c,d,n)
print(libnum.n2s(m))

```



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)