# 一、利用二项式定理和费马定理求解pq

题目链接：https://buuoj.cn/plugins/ctfd-matches/matches/9/challenges#RRRRsa

题目来源平台：BUUCTF

题目名称：RRRRsa

题目描述：利用二项式定理和费马定理求解出p1，q1和p2，q2从而得出p，q最后得出flag

题目类型：CRYPTO

题目附件： 

FLAG：GKCTF{f64310b5-d5e6-45cb-ae69-c86600cdf8d8}

# 二、解题工具

|  |  |  |  |
| --- | --- | --- | --- |
| 工具类型 | 工具名称 | 版本 | 备注 |
| 基础环境 | Python | 3.7 |  |
|  |  |  |
| 脚本 | rsa.py | \ | 根据解题式子，自写代码 |
|  |  |  |
|  |  |  |

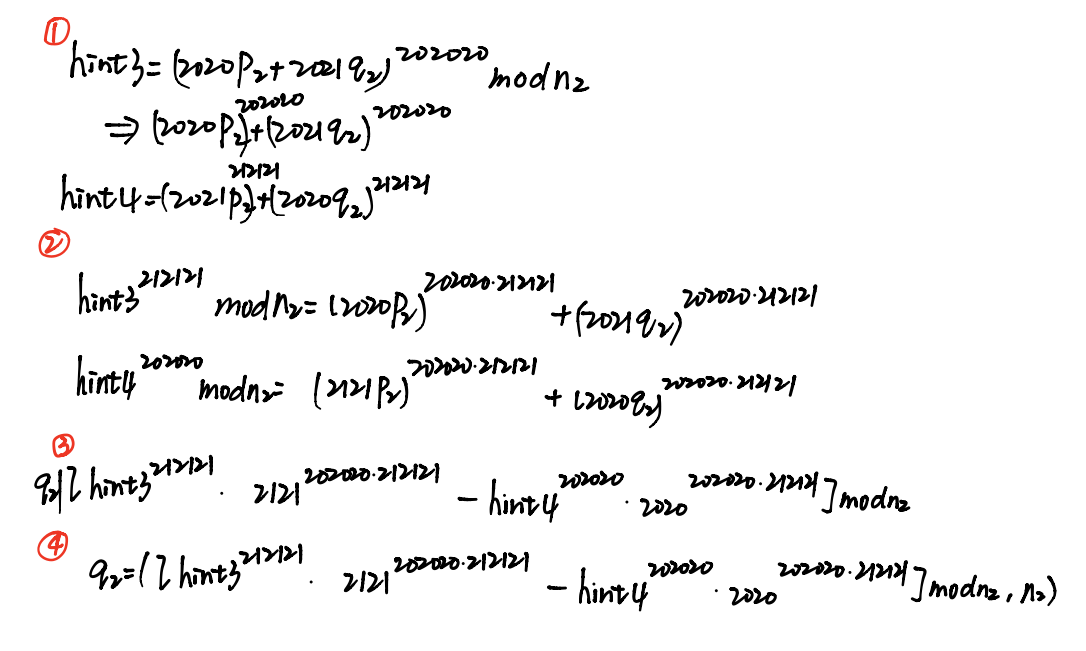
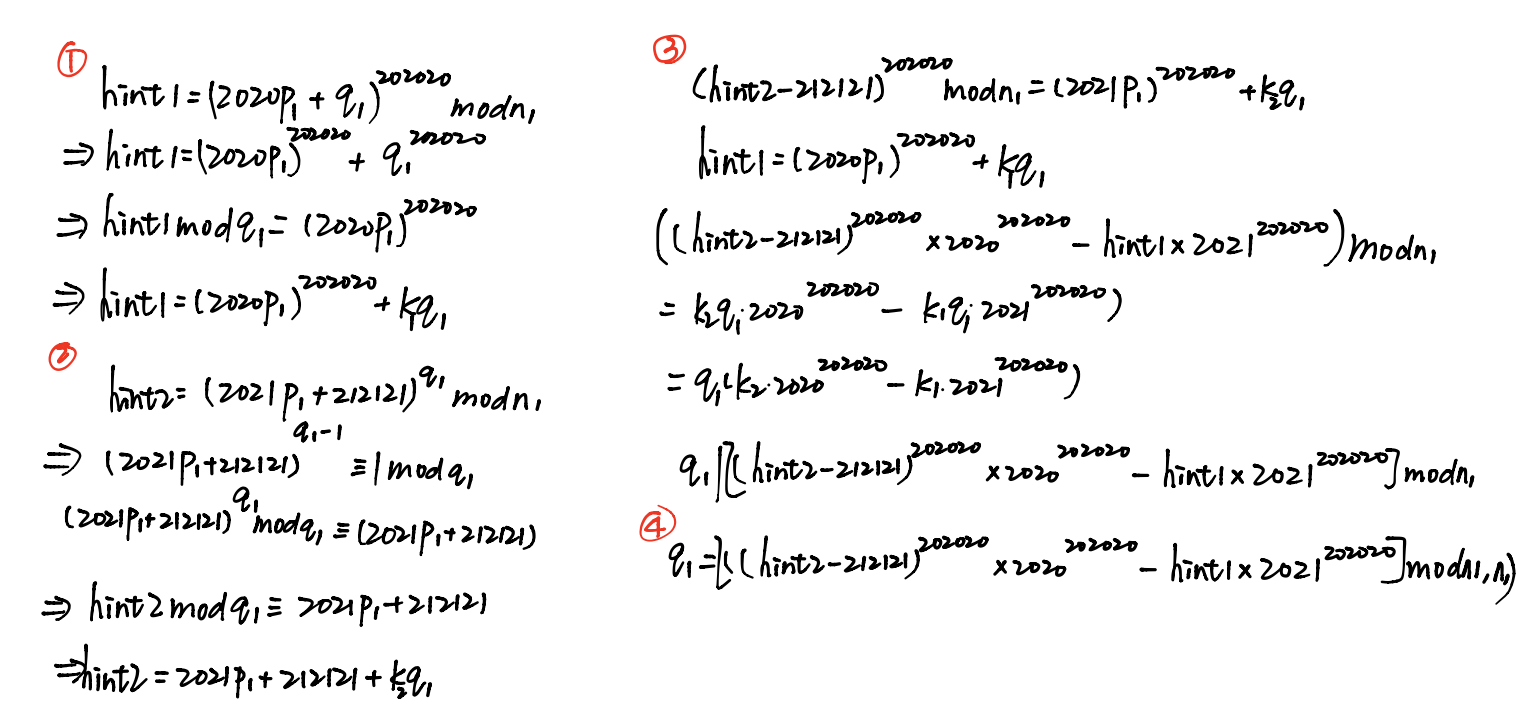
# 三、解题过程

### 步骤1

分析代码，可以发现是由hint1，2，3，4解出p1，q1，p2，q2，然后再得出pq，最后得到flag

### 步骤2

计算q1，q2



### 步骤3

写代码得出flag

|  |
| --- |
| **from Crypto.Util.number import \***  **from gmpy2 import gcd**  **import gmpy2**  **c=13492392717469817866883431475453770951837476241371989714683737558395769731416522300851917887957945766132864151382877462142018129852703437240533684604508379950293643294877725773675505912622208813435625177696614781601216465807569201380151669942605208425645258372134465547452376467465833013387018542999562042758**  **n1=75003557379080252219517825998990183226659117019770735080523409561757225883651040882547519748107588719498261922816865626714101556207649929655822889945870341168644508079317582220034374613066751916750036253423990673764234066999306874078424803774652754587494762629397701664706287999727238636073466137405374927829**  **c1=68111901092027813007099627893896838517426971082877204047110404787823279211508183783468891474661365139933325981191524511345219830693064573462115529345012970089065201176142417462299650761299758078141504126185921304526414911455395289228444974516503526507906721378965227166653195076209418852399008741560796631569**  **hint1=23552090716381769484990784116875558895715552896983313406764042416318710076256166472426553520240265023978449945974218435787929202289208329156594838420190890104226497263852461928474756025539394996288951828172126419569993301524866753797584032740426259804002564701319538183190684075289055345581960776903740881951**  **hint2=52723229698530767897979433914470831153268827008372307239630387100752226850798023362444499211944996778363894528759290565718266340188582253307004810850030833752132728256929572703630431232622151200855160886614350000115704689605102500273815157636476901150408355565958834764444192860513855376978491299658773170270**  **n2=114535923043375970380117920548097404729043079895540320742847840364455024050473125998926311644172960176471193602850427607899191810616953021324742137492746159921284982146320175356395325890407704697018412456350862990849606200323084717352630282539156670636025924425865741196506478163922312894384285889848355244489**  **c2=67054203666901691181215262587447180910225473339143260100831118313521471029889304176235434129632237116993910316978096018724911531011857469325115308802162172965564951703583450817489247675458024801774590728726471567407812572210421642171456850352167810755440990035255967091145950569246426544351461548548423025004**  **hint3=25590923416756813543880554963887576960707333607377889401033718419301278802157204881039116350321872162118977797069089653428121479486603744700519830597186045931412652681572060953439655868476311798368015878628002547540835719870081007505735499581449077950263721606955524302365518362434928190394924399683131242077**  **hint4=104100726926923869566862741238876132366916970864374562947844669556403268955625670105641264367038885706425427864941392601593437305258297198111819227915453081797889565662276003122901139755153002219126366611021736066016741562232998047253335141676203376521742965365133597943669838076210444485458296240951668402513**  **e=65537**  **q1=gcd(n1,pow(hint2-212121,202020,n1)\*pow(2020,202020,n1)-pow(2021,202020,n1)\*hint1)**  **p1=n1//q1**  **d1=gmpy2.invert(e,(q1-1)\*(p1-1))**  **p=pow(c1,d1,n1)**  **q2=gcd(n2,pow(hint3,212121,n2)\*pow(2021,202020\*212121,n2)-pow(hint4,202020,n2)\*pow(2020,202020\*212121,n2))**  **p2=n2//q2**  **d2=gmpy2.invert(e,(q2-1)\*(p2-1))**  **d2 = gmpy2.invert(e,(q2-1)\*(p2-1))**  **q=pow(c2,d2,n2)**  **d = gmpy2.invert(e,(p-1)\*(q-1))**  **m=pow(c,d,p\*q)**  **print(long\_to\_bytes(m))** |

**总结：**

**CTF中常见的类型**

## 1.已知若已知私钥d，则可以直接解密： m=pow(c,d,n)

## 2.已知质数p和q，则通过依次计算欧拉函数值phi、私钥d可解密

|  |
| --- |
| **def rsa\_decrypt(e, c, p, q):**  **phi = (p - 1) \* (q - 1)**  **n = p \* q**  **try:**  **d = gmpy2.invert(e, phi) #求e模phi的逆**  **return pow(c, d, n)**  **except Exception as e:**  **print "e and phi are not coprime!"**  **raise e** |

## 3.N可分解

用yafu分解n： **factor(n)**

再如1和2解题

## 4.模不互素（gcd(N1,N2)!=1）

多个模数n共用质数，则可以很容易利用欧几里得算法求得他们的质因数之一gcd(N1,N2) ，然后这个最大公约数可用于分解模数分别得到对应的p和q，即可进行解密。

### 5.共模攻击

题目链接：https://www.ichunqiu.com/battalion?q=2451

题目来源平台：i春秋

题目名称：veryhardRSA

题目描述：利用共模攻击RSA，求得flag

题目类型：CRYPTO

题目附件：

FLAG：PCTF{M4st3r\_oF\_Number\_Th3ory}

|  |  |  |  |
| --- | --- | --- | --- |
| 工具类型 | 工具名称 | 版本 | 备注 |
| 基础环境 | Python | 3.7 |  |
|  |  |  |
| 脚本 | gongmo.py | \ | 根据改写gongmo攻击的脚本 |
|  |  |  |
|  |  |  |

从题目看很明显是一个N，两个不同的e，所以是共模攻击，直接上脚本

|  |
| --- |
| **# coding=utf-8**  **import gmpy2**  **def ByteToHex(bins):**  **return ''.join(["%02X" % x for x in bins]).strip()**  **def n2s(num):**  **t = hex(num)[2:]**  **if len(t) % 2 == 1:**  **t = '0' + t**  **return ''.join([chr(int(b, 16)) for b in [t[i:i + 2] for i in range(0, len(t), 2)]])**  **n = **  **e1 = 17**  **e2 = 65537**  **s = gmpy2.gcdext(e1, e2)**  **s1 = s[1]**  **s2 = -s[2]**  **file1 = open("flag.enc1", 'rb').read()**  **c1 = int(ByteToHex(file1), 16)**  **file2 = open("flag.enc2", 'rb').read()**  **c2 = int(ByteToHex(file2), 16)**  **c2 = gmpy2.invert(c2, n)**  **m = (pow(c1, s1, n) \* pow(c2, s2, n)) % n**  **print(n2s(m))** |

## 6.小明文攻击

当e很小的时候，并且明文也小时，导致明文的三次方仍然小于n，通过直接对密文进行开方，即可得到明文。

所以直接上py脚本（以e=3为例子）

|  |
| --- |
| **i=0**  **while 1:**  **if(gmpy.root(c+i\*N, 3)[1]==1):**  **print gmpy.root(c+i\*N, 3)**  **break**  **i=i+1** |

## 7.低解密指数攻击

直接跑python：https://github.com/pablocelayes/rsa-wiener-attack

## 8.利用公约数

如果两次加密的n1和n2具有相同的素因子，可以利用欧几里德算法直接分解n1和n2.通过欧几里德算法计算出两个n的最大公约数p

上脚本

|  |
| --- |
| **def gcd(a, b):**  **if a < b:**  **a, b = b, a**  **while b != 0:**  **temp = a % b**  **a = b**  **b = temp**  **def gcd\_digui(a, b):**  **if b != 0:**  **return a**  **return gcd(b,a%b)**  **p = gcd(n1,n2)** |

## 9. Coppersmith Attack

可分为：

### 已知p的高位攻击

知道p的高位为p的位数的约1/2时即可



### 已知明文高位攻击



### 已知d的低位攻击

如果知道d的低位，低位约为n的位数的1/4就可以恢复d



### 明文高位相同



## 10.加密指数e=2（rabin）

直接上脚本

|  |
| --- |
| #!/usr/bin/python  # coding=utf-8  # 适合e=2  import gmpy  import string  from Crypto.PublicKey import RSA  # 读取公钥参数  with open('./tmp/pubkey.pem', 'r') as f:  key = RSA.importKey(f)  N = key.n  e = key.e    p = 275127860351348928173285174381581152299  q = 319576316814478949870590164193048041239  with open('./tmp/flag.enc', 'r') as f:  cipher = f.read().encode('hex')  cipher = string.atoi(cipher, base=16)  # print cipher  # 计算yp和yq  yp = gmpy.invert(p,q)  yq = gmpy.invert(q,p)  # 计算mp和mq  mp = pow(cipher, (p + 1) / 4, p)  mq = pow(cipher, (q + 1) / 4, q)  # 计算a,b,c,d  a = (yp \* p \* mq + yq \* q \* mp) % N  b = N - int(a)  c = (yp \* p \* mq - yq \* q \* mp) % N  d = N - int(c)  for i in (a,b,c,d):  s = '%x' % i  if len(s) % 2 != 0:  s = '0' + s  print s.decode('hex') |

### 11. 已知dp或dq

dp = d mod p-1，dq = d mod q-1



附加：题目所给的公钥是pem文件，要先用openssl rsa -pubin -text -modulus -in pubkey.pem获取公钥