Crypto_[De1CTF2019]babyrsa

原创

M3ng@L 于 2021-11-16 11:25:30 发布 40 收藏 2

文章标签： 密码学 python

本文链接：https://blog.csdn.net/qq_51999772/article/details/121352197

[De1CTF2019]babyrsa

题目描述：

```
import binascii
from data import e1,e2,p,q1p,q1q,hint,flag

n = [20129615352491765499340112943188317180548761597861300847305827141510465619670536844634558246439230371658836928103063432870245707180355907194284861510906071265352409579441048101084995923962148527097370705452070577098780246282820065573711015664291991372085157016901209114191068574208680397710042842835940428451949500607613634682684113208766694028789275748528254287705759528498986306494267817198340658241873024800336013946294891687591013414935237821291805123285905335762719823771647853378892868896078424572232934360940672962436849523915563328779942134504499568866135266628078485232098208237036724121481835035731201383423L, 31221650155627849964466413749414700613823841060149524451234901677160009099014018926581094879840097248543411980533066831976617023676225625067854003317018794041723612556008471579060428898117790587991055681380408263382761841625714415879087478072771968160384909919958010983669368360788505288855946124159513118847747998656422521414980295212646675850690937883764000571667574381419144372824211798018586804674824564606122592483286575800685232128273820087791811663878057827386379787882962763290066072231248814920468264741654086011072638211075445447843691049847262485759393290853117072868406861840793895816215956869523289231421L, 29944537515397953361520922774124192605524711306753835303703478890414163510777460559798334313021216389356251874917792007638299225821018849648520673813786772452822809546571129816310207232883239771324122884804993418958309460009406342872173189008449237959577469114158991202433476710581356243815713762802478454390273808377430685157110095496727966308001254107517967559384019734279861840997239176254236069001453544559786063915970071130087811123912044312219535513880663913831358790376650439083660611831156205113873793106880255882114422025746986403355066996567909581710647746463994280444700922867397754748628425967488232530303L, 25703437855600135215185778453583925446912731661604054184163883272265503323016295700357253105301146726667897497435532579974951478354570415554221401778536104737296154316056314039449116386494323668483749833147800557403368489542273169489080222009368903993658498263905567516798684211462607069796613434661148186901892016282065916190920443378756167250809872483501712225782004396969996983057423942607174314132598421269169722518224478248836881076484639837343079324636997145199835034833367743079935361276149990997875905313642775214486046381368619638551892292787783137622261433528915269333426768947358552919740901860982679180791L]
c = [19131432661217908470262338421299691998526157790583544156741981238822158563988520225986915234570037383888112724408392918113942721994125505014727545946133307329781747600302829588248042922635714391033431930411180545085316438084317927348705241927570432757892985091396044950085462429575440060652967253845041398399648442340042970814415571904057667028157512971079384601724816308078631844480110201787343583073815186771790477712040051157180318804422120472006367220639893153208635806313306471169938197777506841509504162980852614788411776816778672368656662073918470464839540292134953736134906906874730819301484618304257176145691, 15341898433226638235160072029875733826956799982958107910250055958334922460202554924743144122170018355117452459472017133614642242411479849369061482860570279863692425621526056862808425135267608544855833358314071200687340442512856575278712986641573012456729402660597339609443771145347181268285050728925993518704899005416187250003304581230701444705157412790787027926810710998646191467130550713600765898234392350153965811595060656753711278308005193370936296124790772689433773414703645703910742193898471800081321469055211709339846392500706523670145259024267858368216902176489814789679472227343363035428541915118378163012031L, 18715065071648040017967211297231106538139985087685358555650567057715550586464814763683688299037897182845007578571401359061213777645114414642903077003568155508465819628553747173244235936586812445440095450755154357646737087071605811984163416590278352605433362327949048243722556262979909488202442530307505819371594747936223835233586945423522256938701002370646382097846105014981763307729234675737702252155130837154876831885888669150418885088089324534892506199724486783446267336789872782137895552509353583305880144947714110009893134162185382309992604435661
```

```
47774361975873123172248627238135109744930874502817554524287461944 46L, 22822845612248582931384804474633192624749188476301487701124727031285490325921877972899655926151997098578790082717664334620323284985803409688712601896697075185571578365924249732573343629316398310725848241031234865225825316661523638743964827445617581336554064103644421749832270055018609278208712607118610088301206170568835145257987096017440881359994655983386357942751231491654989335801599450323638806135249219130233412094396571459623322134685734028637969205718124182008148170862342622803382211616227895168293638050847156521217390361832640261208687565237701962841422718498790032021909661503900611954693517168195391837975L]
f=lambda m,e,n,c:pow(m,e,n)==c
assert(sum(map(f,[p]*4,[4]*4,n,c))==4)

ee1 = 42
ee2 = 3
ce1 = 45722651786340123946960815003059322528810481841378247280642868553607692149509126962872583037142461398806689489141741494974836882341505234255325683219092163052843461632338442529011502378931140356111756932712822516814023166068902569458299933391973504078898958921809723346229899391366257729496352831842467680394228838643017243088030761974818686389005011393457382050557092810901784264759826663434444718234784936771456468634187100750588672839375114703355688921760464735562855750220836441226994490801130506412294144651699016892470968409220018386065317385627238 4ce2 = 13908468332333567158469136439932325992349696889129103935400760239319454409539725389747059213835238373047899198211128689374049729578146875309231962936554403287882999967840346216695208424582739777034261079550395918048421086843927009452479936045850799096750074359160775182238980989229190157551197830879877097703347301072427149474991803868325769967332356950863518504965486565464059770451458557744949735282131727956056279292800694203866167270268988437389945703117070604488999247750139568614939965885211276821987586882908159585863514561191905040244967655444219603287214405014887994238259270716355378069726760953320025828158tmp = 86407877807860983516777956598254075768407045069785430900517174281341496344746255499901271896092508162157148744725528982424037419052194840720949809891134854871222612682162490991065015935449289960707882463387n = 15911581555796798614711625288508309704791837516232122410440958830726078821069050404012820896260071751380436992710638364294658173571101596931605797509712839622479368850251206419748090059752427303611760004621378226431226983665746837779056271530181865648115862947527212787824629516204832313026456390047768174765687040950636530480549014401279054346098030395100387004111574278813749630986724706263655166289586230453975953773791945408589484679371854113457758157492241225180907090235116325034822993748409011554673180494306003272836905082473475046277554085737627846557240367696214081276345071055578169299060706794192776825039assert(pow(e1,ee1,n)==ce1)
assert(pow(e2+tmp,ee2,n)==ce2)

e = 46531
n = 16278524034278364842964386062476113517067911891699789991355982121084973951738324063305190630865511554888330215827724887964565979607808294168282995825864982603759381323048907814961279012375346497781046417204954101076457350988751188332353062731641153547102721113593787978587135707313755661153376485647168543680503160420091693269984008764444291289486805840439906620313162344057956594836197521501755378387944609246120662335790110901623740990451586621846212047950084207251595169141015645449217847180683357626383565631317253913942886396494396189837432429078251573229378917400841832190737518763297323901586866664595327850603c = 14992132140996160330967307558503117255626925777426611978518339050671013041490724616892634911030918360867974894371539160853827180596100892180735770688723270765387697604426715670445270819626709364566478781273676115921657967761494619448095207169386364541164659123273236874649888236433399127407801843412677293516986398190165291102109310458304626261648346825196743539220198199366711858135271877662410355585767124059539217274691606825103355310348607611233052725805236763220343249873849646219850954945346791015858261715967952461021650307307454434510851869862964236227932964442289459508441345652423088404453536608812799355469hint=int(binascii.hexlify(hint),16)
assert(q1p*q1q==n)
assert(q1p<q1q)
assert(c==pow(hint,e,n))

flag=int(binascii.hexlify(flag),16)
q1=q1p
q2 = 11440118822747958468088404615129970465692053616876713291658918235758346105333638699612378329493256656777369542668944741031196945645857473118751297486829709263867751528358499441638287245016704641657347265884162769098722852879835689480355927830870263528853765319209851496608916812371085467963867142497822195951 3c1 = 26273997575393028169094278432125233903590619684634071323751038236455768537954349876507444882579934219433268118112977004607501812203342198322788771961011202823060316652730302103638635078141444734715038378381686978400659822558337545860958645085460286256902257167204915880987476381283404425741919963121752736704662488883775531121508111
```

```
2255833754586095864508546028625690225716720491588098747638128340442574191996312175273670466248888377553112150811
7338652380608678326619839028909723116817269232665365739352256174194795188757715666666358424910889932705395189148
6355179939770150550995812478327735917006194574412518819299303783243886962455399783601229227718787081785391010424
03050993740360035141417613812470516800228862066480927004612 4
c2 =  7395591129228876649030819616685821899204832684995757724924450812977470787822266387122334722132760470911599 1
7636261722521834540446827001454881726772766987289683810645152039280649746657690706329560374666000318844017091949
0157250829308173310715318925771643105064882620746171266499859049038016902162599261409050907140823352990750298239
5083557672385757098031676768104565596654761211497669478519110647066465067053970916266487136845117804569554535520
2046090963801613412459043842573882682869477396051422191010947394145147143163790318220573873810942973642502562130
830089547318638182675665066784265605041629916631737270770959 6
assert(c1==pow(flag,e1,p*q1))
assert(c2==pow(flag,e2,p*q2))
```

分析程序：

第一段：给出了多组n和c；其中的明文是最后一段的p

定义了一个函数f：

相当于四组$c \equiv p \pmod n$

应用中国剩余定理，直接解出明文p

$p = c * M$

再对p 开四次方即可

第二段：给出了两组e和c，还有共同的n；其中e很小；而加密的明文为最后一段的e1和e2，也就是说明文很大可能不会很长，而在这次加密中的指数e也很小，很有可能可以直接爆破出来明文

也就是$e1 = ce1 + k*$ $e2 =$

爆破k ,k ； 使

$ce2+ k$ 满足能被开 次方 同理 使用gmpy2.iroot()函数

得到明文$e1$，$e2$

第三段：最后一段q1参与构成n；直接在线分解n factordb.com；

得到q1；试着把hint解出来了，但是hint没有用

> hint: orz…you.found.me.but.sorry.no.hint…keep.on.and.enjoy.it!

第四段：终于明文是flag了；其中c1，c2，e1，e2和构成n的p和q1，q2均已知

进行正常的RSA求解；`But!`

这里的$e1$，$e2$和对应的$fai(n)$不互质；求得他们之间的最大公因数$com1=$ $com2= 14$

先让$e1$,$e2$ 除以$com1$,$com2$

再求解$d1$,$d2$

这时求得的是m

推导一下：

> $c \equiv m \ (mod\ n)$
> $d*$
> 而e不是素数；可以分解成a,b,其中a与$fai(n)$互质
> 而此时实际上是$d*$
> 所以$c \equiv m \equiv m\ (mod\ fai(n))$
> 也就是说使用e/com求得的d进而求得的是$m \pmod n$；写作a

我们现在有两个同余式：

$$m \equiv a \ (mod\ n1)$$

$$m \equiv a\ (mod\ n2)$$

其中指数为14，相比3太大了，要是暴力破解或者直接开方是不可能的；

再使用中国剩余定理分解模数：

$$m \equiv a \ (mod\ p)$$

$$m \equiv a\ (mod\ q1)$$

$$m \equiv a\ (mod\ p)$$

$$m \equiv a\ (mod\ q2)$$

这里转换思维（偷个懒）这里使用中国剩余定理求出来的m不是我们需要的明文

取其中两个等式：

$$m \equiv a\ (mod\ q1)$$

$$m \equiv a\ (mod\ q2)$$

为什么不取另外两个等式呢？因为另外两个等式的模数的欧拉函数$p-1$与7不互素（之后要用到）

把这两个等式分别看作另外一个RSA加密公式；进行解密；这个时候m的指数14依然与$q1,q2$不是互素的；但是可以把14看作$2*7$这样7就被当做e进行正常的RSA求解，同理于之前化简m

得到

$$m \equiv b\ (mod\ q1)$$

$$m \equiv b\ (mod\ q2)$$

试了试直接开方b ,b 但是得到m不是我们要的

但实际上m是同时满足这两个等式，那我们就把这两个等式合并

继续使用中国剩余定理进行等式合并

得到的m 进行开方，得到flag

代码实现：

```
from Crypto.Util.number import *
import gmpy2

n = [201296153524917654993401129431883171805487615978613008473058271415104656196705368446345582464392303716588369
28103063432870245707180355907194284861510906071265352409579441048101084995923962148527097370705452070577098780 24
```

```
628282006557371101566429199137208515701690120911419106857420868039771004284283594042845194950060761363468268411320876669402878927574852825428770575952849898630649426781719834065824187302480033601394629489168759101341493523782129180512328590533576271982377164785337889286889607842457223293436094067296243684952391556332877994213450449956886613526662807848523209820823703672412141818350357312013834231222165015562784996446641374941470061382384106014952445123490167716000909901401892658109487984009724854341198053306683197661702367622562506785400331701879404172361255600847157906042889811779058799105568138040826338276184162571441587908747807277196816038490991995801098366936836078850288855946124159513118847747998656422521414980295212646675850690937883764000571667574381419144372824211798018586804674824564606122592483286575800685232128273820087791811663878057827386379787882962763290066072231248814920468264741654086011072638211075445447843691049847262485759393290853117072868406861840793895816215956869523289231421 , 2994453751539795336152092277412419260552471130675383530370347889041416351077746055979833431302121638935625187491779200763829922582101884964852067381378677245282280954657112981631020723288323977132412288480499341895830946000940634287217318900844923795957746911415899120243347671058135624381571376280247845439027380837743068515711009549672796630800125410751796755938401973427986184099723917625423606900145354455978606391597007113008781112391204431221953551388066391383135879037665043908366061183115620511387379310688025588211442202574698640335506699656790958171064774646399428044470092286739775474862842596748823253030 3, 257034378556001352151857784535839254469127316161604054184163883272265503323016295700357253105301146726667897497435532579974951478354570415554221401778536104737296154316056314039449116386494323668483749833147800557403368489542273169489080222009368903993658498263905567516798684211462607069796613434661148186901892016282065916190920443378756167250809872483501712225782004396969996983057423942607174314132598421269169722518224478248836881076484639837343079324636997145199835034833367743079935361276149990997875905313642775214486046381368619638551892292787783137622261433528915269333426768947358552919740901860982679180791]
c =  [191314326612179084702623384212996919985261577905835441567419812388221585639885202259869152345700373838881127244083929181139427219941255050147275459461333073297817476003028295882480429226357143910334319304111805450853164380843179273487052419275704327578929850913960449500854624295754400606529672538450413983996484423400429708144155719040576670281575129710793846017248163080786318444801102017873435830738151867717904777120400511571803188044221204720076367220639893153208635806313306471169938197777506841509504162980852614788411776816778672368656662073918470464839540292134953736134906906874730819301484618304257176145691534189843322663823516007202987573382695679998295810791025005595833492246020255492474314412217001835511745245947201713361464224241147984936906148286057027986369242562152605686280842513526760854485583335831407120068734044251285657527871298664157301245672940266059733960944377114534718126828505072892599351870489900541618725000330458123070144470515741279078702792681071099864619146713055071360076589823439235015396581159506065675371127830800519337093629612479077268943377341470364570391074219389847180008132146905521170933984639250070652367014525902426785836821690217648981478967947222734336303542854191511837816301203 1 , 187150650716480400179672112972311065381399850876853585556505670577155505864648147636836882990378971828450075785714013590612137776451144146429030770035681555084658196285537471732442359365868124454400954507551543576467370870716058119841634165902783526054333623279490482437225562629799094882024425303075058193715947479362238352335869454235222569387010023706463820978461050149817633077292346757377022521551308371548768318858886691504188850880893245348925061997244867834462673367898727821378955525093535833058801449477141100098931341621853823099926044356647774361975873123172248627238135109744930874502817554524287461944461, 228228456122485829313848044746331926247491884763014877011247270312854903259218779728996559261519970985787900827176643346203232849858034096887126018966970751855715783659242497325733436293163983107258482410312348652258253166615236387439648274456175813365540641036444217498322700550186092782087126071186100883012061705688351452579870960174408813599946559833863579427512314916549893358015994503236388061352492191302334120943965714596233221346857340286379692057181241820081481708623426228033822116162278951682936380508471565212173903618326402612086875652377019628414227184987900320219096615039006119546935171681953918379 7]
e = 4
# 中国剩余定理
sum_n = 1
rev_n = []
ni = []
m = 0
for i in n:
    sum_n = sum_n * i
for i in n:
    temp = sum_n//i
    ni.append(temp)
    rev_n.append(gmpy2.invert(temp,i))
for i in range(len(n)):
    m += c[i]*rev_n[i]*ni[i] % sum_n
p = gmpy2.iroot(m%sum_n,e)[0]
print(p)
```

```
print(p)

ee1 = 42
ee2 = 3
ce1 = 45722651786340123946960815003059322528810481841378247280642868553607692149509126962872583037142461398806689489141741494974836882341505234255325683219092163052843461632338442529011502378931140356111756932712822516814023166068902569458299933391973504078898958921809723346229893913662577294963528318424676803942288386430172430880307619748186863890050113934573820505570928109017842647598266634344471823478493677145646863418710075058867283937511470335568892176046473556285575022083644122699449080113050641229414465169901689247096840922001838606531738562723846ce2 = 13908468332333567158469136439932325992349696889129103935400760239319454409539725389747059213835238373047899198211128689374049729578146875309231962936554403287882999967840346216695208424582739777034261079550395918048421086843927009452479936045850799096750074359160775182238980989229190157551197830879877097703347301072427149474991803868325769967332356950863518504965486565464059770451458557744949735282131727956056279292800694203866167270268988437389945703117070604488999247750139568614939965885211276821987586882908159585863514561191905040244967655444219603287214405014887994238259270716355378069726760953320025828158tmp = 86407877807860983516777956598254075768407045069785430900517174281341496344746255499901271896092508162157148744725528982424037419052194840720949809891134854871222612682162490991065015935449289960707882463387n = 15911581555796798614711625288508309704791837516232122410440958830726078821069050404012820896260071751380436992710638364294658173571101596931605797509712839622479368850251206419748090059752427303611760004621378226431226983665746837779056271530181865648115862947527212787824629516204832313026456390047768174765687040950636530480549014401279054346098030395100387004111574278813749630986724706263655166289586230453975953773791945408589484679371854113457758157492241225180907090235116325034822993748409011554673180494306003272836905082473475046277554085737627846557240367696214081276345071055578169299060706794192776825039# 暴力破解k的值def force(ee,ce,n):    k = 0    while True:        temp = ce + k*n        if gmpy2.iroot(temp,ee)[1]:            return gmpy2.iroot(temp,ee)[0]        k = k + 1e1 = force(ee1,ce1,n)# print(e1)e2 = force(ee2,ce2,n) - tmp# print(e2)

p_ = 12758731925343664356931214205855970681549721166108386659253421707931049726036530742609566128110371004239277545386617465740498553906674168419602013784047295010238023206778640032260090293898491635563171443966832667131016091676647289753605537147407608977947237291303704015335643752880892291148404946034208883487 1# 求hint的过程# q_ = 12758731925343664356931214205855970681549721166108386659253421707931049726036530742609566128110371004239277545386617465740498553906674168419602013784047295010238023206778640032260090293898491635563171443966832667131016091676647289753605537147407608977947237291303704015335643752880892291148404946034208883569# c = 14992132140996160330967307558503117255626925777426611978518339050671013041490724616892634911030918360867974894371539160853827180596100892180735770688723270765387697604426715670445270819626709364566478781273676115921657967761494619448095207169386364541164659123273236874649888236433399127407801843412677293516986398190165291102109310458304626261648346825196743539220198199366711858135271877662410355585767124059539217274691606825103355310348607611233052725805236763220343249873849646219850954945346791015858261715967952461021650307307454434510851869862964236227932964442289459508441345652423088404453536608812799355469# fai_n = (p_-1)*(q_-1)# e = 46531# d = gmpy2.invert(e,fai_n)# n = p_*q_# m = pow(c,d,n)# print(long_to_bytes(m))q1 = p_

q2 = 11440118822747958468088404615129970465692053616876713291658918235758346105333638699612378329493256656777369542668944741031196945645857473118751297486829709263867751528358499441638287245016704641657347265884162769098722852879835689480355927830870263528853765319209851496608916812371085467963867142497822195951 3
```

```
c1 = 26273997575393028169094278432125233903590619684634071323751038236455768537954349876507444882579934219433268
11811297700460750181220334219832278877196101120282306031665273030210363863507814144473471503837838168697840065982255833754586095864508546028625690225716720491588098747638128340442574191996312175273670466248888377553112150811733865238060867832661983902890972311681726923266536573935225617419479518875771566666635842491088993270539518914863551799397701505509958124783277359170061945744125188192993037832438869624553997836012292277187870817853910104240305099374036003514141761381247051680022886206648092700461 24
c2 = 73955911292288766490308196166858218992048326849957577249244508129774707878222663871223347221327604709115991763626172252183454044682700145488172677276698728968381064515203928064974665769070632956037466600031884401709194901572508293081733107153189257716431050648826207461712664998590490380169021625992614090509071408233529907502982395083557672385757098031676768104565596654761211497669478519110647066465067053970916266487136845117804569554535520204609096380161341245904384257388268286947739605142219101094739414514714316379031822057387381094297364250256213083008954731863818267566506678426560504162991663173727077095 96
fai_n1 = (p-1)*(q1-1)
fai_n2 = (p-1)*(q2-1)
com1 = gmpy2.gcd(e1,fai_n1)
com2 = gmpy2.gcd(e2,fai_n2)


d1 = gmpy2.invert(e1//com1,(p-1)*(q1-1))
d2 = gmpy2.invert(e2//com2,(p-1)*(q2-1))

a1 = pow(c1,d1,p*q1) #同余于m**14
a2 = pow(c2,d2,p*q2)
d1_ = gmpy2.invert(7,(q1-1)) #q1的欧拉函数等于(q1-1);同理q2
d2_ = gmpy2.invert(7,(q2-1))
b1 = pow(a1,d1_,q1) #同余于m**2
b2 = pow(a2,d2_,q2)
#进行中国剩余定理合并等式
M = q1*q2
M1 = q2
M2 = q1
rev_M1 = gmpy2.invert(M1,q1)
rev_M2 = gmpy2.invert(M2,q2)
m = (b1*M1*rev_M1 + b2*M2*rev_M2) % M
print(long_to_bytes(gmpy2.iroot(m,2)[0]))
```

思路来源：当中国剩余定理邂逅RSA - 安全客，安全资讯平台 (anquanke.com)