2021-07-15

原创

[De1CTF2019]babyrsa

题目

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

n =  [20129615352491765499340112943188317180548761597861300847305827141510465619670536844634558246439230371658836928103063432870245707180355907194284861510906071265352409579441048101084995923962148527097370705452070577098780246282820065573711015664291991372085157016901209114191068574208680397710042842835940428451949500607613634682684113208766694028789275748528254287705759528498986306494267817198340658241873024800336013946294891687591013414935237821291805123285905335762719823771647853378892868896078424572232934360940672962436849523915563328779942134504499568866135266628078485232098208237036724121481835035731201383423L, 31221650155627849964466413749414700613823841060149524451234901677160009099014018926581094879840097248543411980533066831976617023676225625067854003317018794041723612556008471579060428898117790587991055681380408263382761841625714415879087478072771968160384909919958010983669368360788505288855946124159513118847747998656422521414980295212646675850690937883764000571667574381419144372824211798018586804674824564606122592483286575800685232128273820087791811663878057827386379787882962763290066072231248814920468264741654086011072638211075445447843691049847262485759393290853117072868406861840793895816215956869523289231421L, 29944537515397953361520922774124192605524711306753835303703478890414163510777460559798334313021216389356251874917792007638299225821018849648520673813786772452822809546571129816310207232883239771324122884804993418958309460009406342872173189008449237959577469114158991202433476710581356243815713762802478454390273808377430685157110095496727966308001254107517967559384019734279861840997239176254236069001453544559786063915970071130087811123912044312219535513880663913831358790376650439083660611831156205113873793106880255882114422025746986403355066996567909581710647746463994280444700922867397754748628425967488232530303L, 25703437855600135215185778453583925446912731661604054184163883272265503323016295700357253105301146726667897497435532579974951478354570415554221401778536104737296154316056314039449116386494323668483749833147800557403368489542273169489080222009368903993658498263905567516798684211462607069796613434661148186901892016282065916190920443378756167250809872483501712225782004396969996983057423942607174314132598421269169722518224478248836881076484639837343079324636997145199835034833367743079935361276149990997875905313642775214486046381368619638551892292787783137622261433528915269333426768947358552919740901860982679180791L]
c =  [19131432661217908470262338421299691998526157790583544156741981238822158563988520225986915234570037383888112724408392918113942721994125505014727545946133307329781747600302829588248042922635714391033431930411180545085316438084317927348705241927570432757892985091396044950085462429575440060652967253845041398399648442340042970814415571904057667028157512971079384601724816308078631844480110201787343583073815186771790477712040051157180318804422120472007636722063989315320863580631330647116993819777750684150950416298085261478841177681677867236865666207391847046483954029213495373613490690687473081930148461830425717614569L, 15341898433226638235160072029875733826956799982958107910250055958334922460202554924743144122170018355117452459472017133614642242411479849369061482860570279863692425621526056862808425135267608544855833358314071200687340442512856575278712986641573012456729402660597339609443771145347181268285050728925993518704899005416187250003304581230701444705157412790787027926810710998646191467130550713600765898234392350153965811595060656753711278308005193370936296124790772689433773414703645703910742193898
```

```
47180008132146905521170953984639250070652367014525902426785836821690217648981478967947222734336303542854191511837816301203 1L, 18715065071648040017967211297231106538139985087685358555650567057715550586464814763683688299037897182845007578571401359061213777645114414642903077003568155508465819628553747173244235936586812445440095450755154357646737087071605811984163416590278352605433362327949048243722556262979909488202442530307505819371594747936223835233586945423522256938701002370646382097846105014981763307729234675737702252155130837154876831885888669150418885088089324534892506199724486783446267336789872782137895552509353583305880144947714110009893134162185382309992604435664777436197587312317224862723813510974493087450281755452428746194446L, 22822845612248582931384804474633192624749188476301487701124727031285490325921877972899655926151997098578790082717664334620323284985803409688712601896697075185571578365924249732573343629316398310725848241031234865225825316661523638743964827445617581336554064103644421749832270055018609278208712607118610088301206170568835145257987096017440881359994655983386357942751231491654989335801599450323638806135249219130233412094396571459623322134685734028637969205718124182008148170862342622803382211616227895168293638050847156521217390361832640261208687565237701962841422718498790032021909661503900611954693517168195391837971]
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 = 4572265178634012394696081500305932252881048184137824728064286855360769214950912696287258303714246139880668948914174149497483688234150523425532568321909216305284346163233844252901150237893114035611175693271282251681402316606890256945829993339197350407889895892180972334622989391366257729496352831842467680394228838643017243088030761974818686389005011393457382050557092810901784264759826663434444718234784936771456468634187100750588672839375114703355688921760464735562855750220836441226994490801130506412294144651699016892470968409220018386065317385627238284
ce2 = 1390846833233356715846913643993232599234969688912910393540076023931945440953972538974705921383523837304789919821112868937404972957814687530923196293655440328788299996784034621669520842458273977034261079550395918048421086843927009452479936045850799096750074359160775182238980989229190157551197830879877097703347301072427149474991803868325769967332356950863518504965486565464059770451458557744949735282131727956056279292800694203866167270268988437389945703117070604488999247750139568614939965885211276821987586882908159585863514561191905040244967655444219603287214405014887994238259270716355378069726760953320025828158
tmp = 8640787780786098351677795659825407576840704506978543090051717428134149634474625549990127189609250816215714874447255289824240374190521948407209498098911348548712226126821624909910650159354492899607078824633874
n = 1591158155579679861471162528850830970479183751623212241044095883072607882106905040401282089626007175138043699271063836429465817357110159693160579750971283962247936885025120641974809005975242730361176000462137822643122698366574683777905627153018186564811586294752721278782462951620483231302645639004776817476568704095063653048054901440127905434609803039510038700411157427881374963098672470626365516628958623045397595377379194540858948467937185411345775815749224122518090709023511632503482299374840901155467318049430600327283690508247347504627755408573762784655724036769621408127634507105557816929906070679419277682503 9
assert(pow(e1,ee1,n)==ce1)
assert(pow(e2+tmp,ee2,n)==ce2)

e = 46531
n = 1627852403427836484296438606247611351706791189169978999135598212108497395173832406330519063086551155488833021582772488796456597960780829416828299582586498260375938132304890781496127901237534649778104641720495410107645735098875118833235306273164115354710272111359378797858713570731375566115337648564716854368050316042009169326998400876444429128948680584043990662031316234405795659483619752150175537838794460924612066233579011090162374099045158662184621204795008420725159516914101564544921784718068335762638356563131725391394288639649439618983743242907825157322937891740084183219073751876329732390158686666459532785060 3
c = 1499213214099616033096730755850311725562692577742661197851833905067101304149072461689263491103091836086797489437153916085382718059610089218073577068872327076538769760442671567044527081962670936456647878127367611592165796776149461944809520716938636454116465912327323687464988823643339912740780184341267729351698639819016529110210931045830462626164834682519674353922019819936671185813527187766241035558576712405953921727469160682510335531034860761123305272580523676322034324987384964621985095494534679101585826171596795246102165030730745443451085186986296423622793296444228945950844134565242308840445353660881279935546 9
hint=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 =  1144011882274795846808840461512997046569205361687671329165891823575834610533363869961237832949325665677736
9542668944741031196945645857473118751297486829709263867751528358499441638287245016704641657347265884162769098722
8528798356894803559278308702635288537653192098514966089168123710854679638671424978221959513
c1 =  2627399757539302816909427843212523390359061968463407132375103823645576853795434987650744488257993421943326
8118112977004607501812203342198322788771961011202823060316652730302103638635078141444734715038378381686978400659
8225583375458609586450854602862569022571672049158809874763812834044257419199631217527367046624888837755311215081
1733865238060867832661983902890972311681726923266536573935225617419479518875771566666635842491088993270539518914
8635517993977015055099581247832773591700619457441251881929930378324388696245539978360122922771878708178539101042
4030509937403600351414176138124705168002288620664809270046124
c2 =  7395591129228876649030819616685821899204832684995757724924450812977470787822266387122334722132760470911599
1763626172252183454044682700145488172677276698728968381064515203928064974665769070632956037466600031884401709194
9015725082930817331071531892577164310506488262074617126649985904903801690216259926140905090714082335299075029823
9508355767238575709803167676810456559665476121149766947851911064706646506705397091626648713684511780456955453552
0204609096380161341245904384257388268286947739605142219101094739414514714316379031822057387381094297364250256213
0830089547318638182675665066784265605041629916631737270770959 6
assert(c1==pow(flag,e1,p*q1))
assert(c2==pow(flag,e2,p*q2))
```

解题

先解e1、e2:

```
ee1 = 42
ee2 = 3
ce1 =  4572265178634012394696081500305932252881048184137824728064286855360769214950912696287258303714246139880668948914174149497483688234150523425532568321909216305284346163233844252901150237893114035611175693271282251681402316606890256945829993339197350407889895892180972334622989391366257729496352831842467680394228838643017243088030761974818686389005011393457382050557092810901784264759826663434444718234784936771456468634187100750588672839375114703355688921760464735562855750220836441226994490801130506412294144651699016892470968409220018386065317385627238 4
ce2 =  1390846833233356715846913643993232599234969688912910393540076023931945440953972538974705921383523837304789919821112868937404972957814687530923196293655440328788299996784034621669520842458273977703426107955039591804842108684392700945247993604585079909675007435916077518223898098922919015755119783087987709770334730107242714947499180386832576996733235695086351850496548656546405977045145855774494973528213172795605627929280069420386616727026898843738994570311707060448899924775013956861493996588521127682198758688290815958586351456119190504024496765544421960328721440501488799423825927071635537806972676095332002582815 8
tmp =  8640787780786098351677795659825407576840704506978543090051717428134149634474625549990127189609250816215714874472552898242403741905219484072094980989113485487122261268216249099106501593544928996070788246338 7
n  =  1591158155579679861471162528850830970479183751623212241044095883072607882106905040401282089626007175138043699271063836429465817357110159693160579750971283962247936885025120641974809005975242730361176000462137822643122698366574683777905627153018186564811586294752721278782462951620483231302645639004776817476568704095063653048054901440127905434609803039510038700411157427881374963098672470626365516628958623045397595377379194540858948467937185411345775815749224122518090709023511632503482299374840901155467318049430600327283690508247347504627755408573762784655724036769621408127634507105557816929906070679419277682503 9
import gmpy2
e1 = gmpy2.iroot(ce1,ee1)[0]
for k in range(1000000):
    c = k*n + ce2
    if gmpy2.iroot(c,3)[1]:
        m = gmpy2.iroot(c,3)[0]
        e2 = m - tmp
        break

if ce2 == gmpy2.powmod((e2+tmp),3,n):
    print(e1,e2)
```

因为n没法直接分解，e1可以用小明文攻击，但e2不行，直接爆破，运行得到：

```
15218928658178
381791429275130
```

然后求hint和q1p，这次n能分解了，

```
e = 46531
n = 162785240342783648429643860624761135170679118916997899913559821210849739517383240633051906308655115548883302
158277248879645659796078082941682829958258649826037593813230489078149612790123753464977810464172049541010764573509
098875118833235306273164115354710272111359378797858713570731375566115337648564716854368050316042009169326998400876
764444291289486805840439906620313162344057956594836197521501755378387944609246120662335790110901623740990451586621
846212047950084207251595169141015645449217847180683357626383565631317253913942886396494396189837432429078251573229
378917400841832190737518763297323901586866664595327850603
c = 149921321409961603309673075585031172556269257774266119785183390506710130414907246168926349110309183608679748943
715391608538271805961008921807357706887232707653876976044267156704452708196267093645664787812736761159216579677614
946194480952071693863645411646591232732368746498882364333991274078018434126772935169863981901652911021093104583046
262616483468251967435392201981993667118581352718776624103555857671240595392172746916068251033553103486076112330527
258052367632203432498738496462198509549453467910158582617159679524610216503073074544345108518698629642362279329644
422894595084413456524230884044535366088127993554690
```

```
import gmpy2
q1p=127587319253436643569312142058559706815497211661083866592534217079310497260365307426095661281103710042392775453866174657404985539066741684196020137840472950102380232067786400322600902938984916355631714439668326671310160916766472897536055371474076089779472372913037040153356437528808922911484049460342088834871
q1q=127587319253436643569312142058559706815497211661083866592534217079310497260365307426095661281103710042392775453866174657404985539066741684196020137840472950102380232067786400322600902938984916355631714439668326671310160916766472897536055371474076089779472372913037040153356437528808922911484049460342088835693

phi = (q1q-1) * (q1p-1)
d = gmpy2.invert(e,phi)
hint = gmpy2.powmod(c,d,n)
print(q1p,hint)
```

运行得到：

```
127587319253436643569312142058559706815497211661083866592534217079310497260365307426095661281103710042392775453866174657404985539066741684196020137840472950102380232067786400322600902938984916355631714439668326671310160916764728975360553714740760897794723729130370401533564375288089229114840494603420888348711359023682381752345826602955392877144623020570939395059395533025741411559308080529422522799281538296521307189886688033780706298388420417938879521
```

然后继续，该求flag了，但是我好像还不知道p，先用中国剩余定理求p，代码直接百度就有：

```
import gmpy2
import math
def  merge(a1,n1,a2,n2):
    d = math.gcd(n1,n2)
    c = a2-a1
    if c%d!=0:
        return 0
    c = (c%n2+n2)%n2
    c = c//d
    n1 = n1//d
    n2 = n2//d
    c *= gmpy2.invert(n1,n2)
    c %= n2
    c *= n1*d
    c += a1
    global n3
    global a3
```

```
    global a3
    n3 = n1*n2*d
    a3 = (c%n3+n3)%n3
    return 1
def exCRT(a,n):
    a1=a[0]
    n1=n[0]
    le= len(a)
    for i in range(1,le):
        a2 = a[i]
        n2=n[i]
        if not merge(a1,n1,a2,n2):
            return -1
        a1 = a3
        n1 = n3
    global mod
    mod=n1
    return (a1%n1+n1)%n1
def exCRT_getequation(a,n):
    a1=a[0]
    n1=n[0]
    le= len(a)
    for i in range(1,le):
        a2 = a[i]
        n2=n[i]
        if not merge(a1,n1,a2,n2):
            return -1
        a1 = a3
        n1 = n3
    return (a1,n1)
#a为余数列表
#n为模数列表
n =  [201296153524917654993401129431883171805487615978613008473058271415104656196705368446345582464392303716588369281030634328702457071803559071942848615109060712653524095794410481010849959239621485270973707054520705770987802462828200655737110156642919913720851570169012091141910685742086803977100428428359404284519495006076136346826841132087666940287892757485282542877057595284989863064942678171983406582418730248003360139462948916875910134149352378212918051232859053357627198237716478533788928688960784245722329343609406729624368495239155633287799421345044995688661352666280784852320982082370367241214818350357312013834231, 312216501556278499644664137494147006138238410601495244512349016771600090990140189265810948798400972485434119805330668319766170236762256250678540033170187940417236125560084715790604288981177905879910556813804082633827618416257144158790874780727719681603849099199580109836693683607885052888559461241595131188477479986564225214149802952126466758506909378837640005716675743814191443728242117980185868046748245646061225924832865758006852321282738200877918116638780578273863797878829627632900660722312488149204682647416540860110726382110754454478436910498472624857593932908531170728684068618407938958162159568695232892314219, 299445375153979533615209227741241926055247113067538353037034788904141635107774605597983343130212163893562518749177920076382992258210188496485206738137867724528228095465711298163102072328832397713241228848049934189583094600094063428721731890084492379595774691141589912024334767105813562438157137628024784543902738083774306851571100954967279663080012541075179675593840197342798618409972391762542360690014535445597860639159700711300878111239120443122195355138806639138313587903766504390836606118311562051138737931068802558821144220257469864033550669965679095817106477464639942804447009228673977547486284259674882325303032, 257034378556001352151857784535839254469127316616040541841638832722655033230162957003572531053011467266678974974355325799749514783545704155542214017785361047372961543160563140394491163864943236684837498331478005574033684895422731694890802220093689039936584982639055675167986842114626070697966134346611481869018920162820659161909204433787561672508098724835017122257820043969699969830574239426071743141325984212691697225182244782488368810764846398373430793246369971451998350348333677430799353612761499909978759053136427752144860463813686196385518922927877831376222614335289152693334267689473585529197409018600982679180791]
c =  [191314326612179084702623384212996919985261577905835441567419812388221585639885202259869152345700373838881127244083929181139427219941255050147275459461333073297817476003028295882480429226357143910334319304111805450853164380843179273487052419275704327578929850913960449500854624295754400606529672538450413983996484423400429708144155719040576670281575129710793846017248163080786318444801102017873435830738151867717904777120400511571803188044221204720076367220639893153208635806313306471169938197777506841509504162980852614788411776816778672368656662073918471860982679180791]
```

```
046483954029213495373613490690687473081930148461830425717614569, 153418984332266382351600720298757338269567999829581079102500559583349224602025549247431441221700183551174524594720171336146422424114798493690614828605702798636924256215260568628084251352676085448558333583140712006873404425128565752787129866415730124567294026605973396094437711453471812682850507289259935187048990054161872500033045812307014447051574127907870279268107109986461914671305507136007658982343923501539658115950606567537112783080051933709362961247907726894337734147036457039107421938984718000813214690552117093398463925007065236701452590242678583682169021764898147896794722273433630354285419151183781631230 12, 187150650716480400179672112972311065381399850876853585556505670577155505864648147636836882990378971828450075785714013590612137776451144146429030770035681555084658196285537471732442359365868124454400954507551543576467370870716058119841634165902783526054333623279490482437225562629799094882024425303075058193715947479362238352335869454235222569387010023706463820978461050149817633077292346757377022521551308371548768318858886691504188850880893245348925061997244867834462673367898727821378955525093535833058801449477141100098931341621853823099926044356647774361975873123172248627238135109744930874502817554524287461944462282284561224858293138480447463319262474918847630148770112472703128549032592187797289965592615199709857879008271766433462032328498580340968871260189669707518557157836592424973257334362931639831072584824103123486522582531666152363874396482744561758133655406410364442174983227005501860927820871260711861008830120617056883514525798709601744088135999465598338635794275123149165498933580159945032363880613524921913023341209439657145962332213468573402863796920571812418200814817086234262280338221161622789516829363805084715652121739036183264026120868756523770196284142271849879003202190966150390061195469351716819539183797]
p_4=exCRT(c,n)
p=gmpy2.iroot(p_4,4)[0]
print(p)
```

运行得到:

```
109935857933867829728985398563235455481120300859311421762540858762721955038310117609456763338082237907005937380873151279351831600225270995344096532750271070807051984097524900957809427861441436796934012393707770012556604479065826879107677002380580866325868240270494148512743861326447181476633546419262340100453
```

最后，求flag

但是gcd(e1,phi1) = gcd(e2,phi2) = 14□
所以这次还要用到中国剩余定理

```
import gmpy2
#记得导入中国剩余定理的代码
p=109935857933867829728985398563235455481120300859311421762540858762721955038310117609456763338082237907005937380873151279351831600225270995344096532750271070807051984097524900957809427861441436796934012393707770012556604479065826879107677002380580866325868240270494148512743861326447181476633546419262340100453
phi1=(p-1)*(q1-1)
phi2=(p-1)*(q2-1)
xx1=gmpy2.gcd(e1,phi1)
xx2=gmpy2.gcd(e2,phi2)
d1=gmpy2.invert(e1//xx1,phi1)
d2=gmpy2.invert(e2//xx2,phi2)
nn=[]
aa=[]
nn.append(q1)
nn.append(q2)
a1=gmpy2.powmod(c1,d1,p*q1)%q1
a2=gmpy2.powmod(c2,d2,p*q2)%q2
aa.append(a1)
aa.append(a2)
last=exCRT_getequation(aa,nn)#最终方程组 aa=n^14%q1*q2
new_e=7
new_phi=(q1-1)*(q2-1)
new_d=gmpy2.invert(new_e,new_phi)
m_2=gmpy2.powmod(last[0],new_d,last[1])#特解m_2
flag=gmpy2.iroot(m_2,2)[0]
import binascii
print(binascii.unhexlify(hex(flag)[2:]))
```

运行得到：b’de1ctf{9b10a98b-71bb-4bdf-a6ff-f319943de21f}’

答案

flag{9b10a98b-71bb-4bdf-a6ff-f319943de21f}

[ACTF新生赛2020]crypto-aes

题目

output

```
91144196586662942563895769614300232343026691029427747065707381728622849079757
b'\x8c-\xcd\xde\xa7\xe9\x7f.b\x8aKs\xf1\xba\xc75\xc4d\x13\x07\xac\xa4&\xd6\x91\xfe\xf3\x14\x10|\xf8p'
```

```
from Cryptodome.Cipher import AES
import os
import gmpy2
from flag import FLAG
from Cryptodome.Util.number import *

def main():
    key=os.urandom(2)*16
    iv=os.urandom(16)
    print(bytes_to_long(key)^bytes_to_long(iv))
    aes=AES.new(key,AES.MODE_CBC,iv)
    enc_flag = aes.encrypt(FLAG)
    print(enc_flag)
if __name__=="__main__":
    main()
```

解题

已知key与iv异或的结果，而且key是两字节的字符串重复16次，所以

```
key_iv = 91144196586662942563895769614300232343026691029427747065707381728622849079757
enc_flag = b'\x8c-\xcd\xde\xa7\xe9\x7f.b\x8aKs\xf1\xba\xc75\xc4d\x13\x07\xac\xa4&\xd6\x91\xfe\xf3\x14\x10|\xf8p'
print(hex(key_iv))
key=hex(key_iv)[2:6]*16
iv=key_iv^eval('0x'+key)
```

求出key与iv之后进行AES的解密

```
import Crypto.Util.number
import Crypto.Cipher.AES

iv=Crypto.Util.number.long_to_bytes(iv)
key=Crypto.Util.number.long_to_bytes(eval('0x'+key))
decrypt=Crypto.Cipher.AES.new(key,Crypto.Cipher.AES.MODE_CBC,iv)
print(decrypt.decrypt(enc_flag))
```

答案

flag{W0W_y0u_can_so1v3_AES_now!}