2022年HGAME中CRYPTO的Multi Prime RSA

原创

沐一 · 林 于 2022-04-04 16:19:52 发布 123 收藏 1

分类专栏： CTF 密码学 文章标签： ctf

本文链接：https://blog.csdn.net/xiao__1bai/article/details/122779920

CTF 同时被 2 个专栏收录

167 篇文章 6 订阅

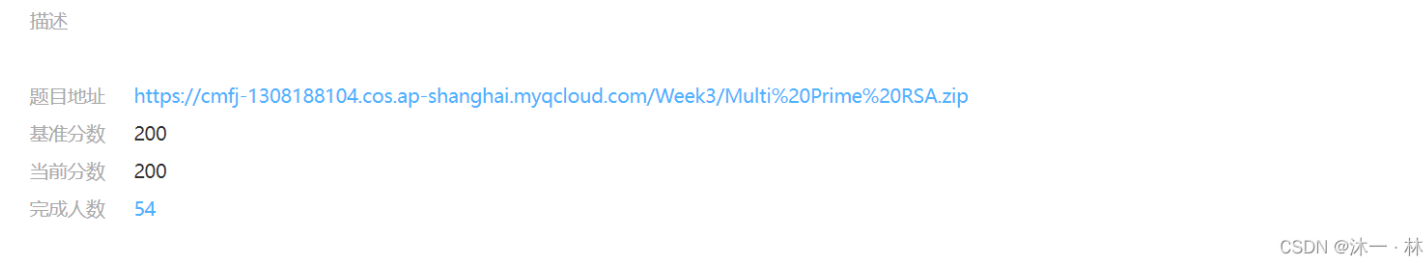
订阅专栏

密码学

51 篇文章 1 订阅

订阅专栏

2022年HGAME中CRYPTO的Multi Prime RSA



.

.

照例下载附件，看了一眼，发现是变形的RSA加密，变形处为 n = p ** 2 * q ** 3 * r ** 5 * s ** 7：

```python
from Crypto.Util.number import getPrime
from libnum import s2n
from secret import flag

p = getPrime(256)
q = getPrime(256)
r = getPrime(256)
s = getPrime(256)
n = p ** 2 * q ** 3 * r ** 5 * s ** 7
e = 65537
c = pow(s2n(flag), e, n)
print(f"p = {p}")
print(f"q = {q}")
print(f"r = {r}")
print(f"s = {s}")
print(f"n = {n}")
print(f"e = {e}")
print(f"c = {c}")
```

```
p = 61789932148719477384027458333380568978056286136137829092952317307711908353477
q = 91207969353355763685633284378833506319794714507027332929290701748727534193861
r = 10547129960737538862234727247920794450967050283565125094520339753001086180936 7
s = 83153238748903772448138307505579799277162652151244477391465130504267171881437
n = 10393443721650871000010639205981518123241510646848418452509747585252651485677061037849584248731817213524402092848124937539725565194820263272826446190914668865238048412482772103531733834079445984538481138158669085953356194585494869587644901038084753295980858421849630650684994898864679110872950871637625992846220551854569057745072457816672931992053176920298294959614873479448138744154237719806607789862111458417124126311563691291464701191351363781582034595765962461691914194885608327340460761076730919958600218632398826086384581499302559441848638012783865510319801464602315157477544116786517526988810014649739814242407814130849419472618752897255389597205724963293484998705800579975408444883091110592407450810483247628665729483712228392787180344357398276771900255008024536268723562086127184172496494745711971670769164035823941863578126405662509303612762299695531281283127362454401295560201081888359661314259564317964177204364740933817707964316295230543782584975460132224949745492621404155851589859409664154594781507228321196913086975101890264473591899940558850907354117383322962540112085476769140048647323278638842177334562873697710870945147084686856418203752208354850534825708526193630911733242033345034618239836108868499309442505539288555060126845042115255429985752756267841297363451427723991092736195224459195 e = 65537
c = 84467739549646641152039419086978726120996024673441540621797598641886576068002454211923187325913186120887852203000992305799152676134642313024212188449325773206770085789737985954535660915183422380426217493519171827121180922173060160282712224923808603058097137610472498780104950068913412260983432158660922376114053807946083021382467436160104636763722709401838190129148865964272054958385681274787751960080432557042177057599928938917502164634737187923402364765750717851904723674607142032715518821383929338228878785377754022619264476102882225616570678739589113476590822903604446847351916614161060479148507170280885494467241812420328932812479334819804860133847608648231824826450878978196791020539374083534508678434514535136749119771793375741496781159491369258831416166933314773304817104438654689234647518119748270216446854243018788507416317784328594899994332804915902187382125426747106752360915100788513192189646216121635645411692979635581575664262136997426036537807033629054297159988632523282198108034185895060915781376941645533793509669663562342641816631673131174435618543058086342714723330814586496030805366321181723292731710369013923285787724941830672247377301048663929453294620044701627159066468762709113137517559435822623284148112827473010030736329596829357275518641576798298066541516764673029908084962144713
```

.

.

因为有一点`变形`，所以常规的`CTF-RSA-tool`工具需要二次修改脚本才行，但是它和2`021年9月绿城杯中CRYPTO的RSA-1`的明文变形不同，没法直接在`RSAutils.p`y处修改明文输出。

然后又从`第七届“湖湘杯”Crypto的signin`中得到一点启发，逆模求d时 p 、q、r、s `每一个`都看成因子即可。

.

.

.

所以最终脚本只是常规脚本的多因子加密而已：

```
import libnum
from Crypto.Util.number import long_to_bytes

p = 617899321487194773840274583333805689780562861361378290929523173077119083534770
q = 912079693533557636856332843788335063197947145070273329292907017487275341938610
r = 105471299607375388622347272479207944509670502835651250945203397530010861809367
s = 831532387489037724481383075055797992771626521512444773914651305042671718814370
n = 103934437216508710000106392059815181232415106468484184525097475852526514856770610378495842487318172135244020928481249375397255651948202632728264461909146688652380484124827721035317338340794459845384811381586690859533561945854948695876449010380847532959808584218496306506849948988646791108729508716376259928462205518545690577450724578166729319920531769202982949596148734794481387441542377198066077898621114584171241263115636912914647011913513637815820345957659624616919141948856082373404607610767309199586002186323988260863845814993025594418486380127838655103198014646023151574775441167865175269888100146497398142424078141308494194726187528972553895972057249632934849987058005799754084448830911105924074508104832476286657294837122283927871803443573982767719002550080245362687235620861271841724964947457119716707691640358239418635781264056625093036127622996955312812831273624544012955602010818883596613142595643179641772043647409338177079643162952305437825849754601322249497454926214041558515898594096641545947815072283211969130869751018902644735918999405588509073541173833229625401120854767691400486473232786388421773345628736977108709451470846868564182037522083548505348257085261936309117332420333450346182398361088684993094425055392885550601268450421152554299857527562678412973634514277239910927361952244591 9
e = 65537
c = 844677395496466411520394190869787261209960246734415406217975986418865760680024542119231873259131861208878522030009923057991526761346423130242121884493257732067700857897379859545356609151834223804262174935191718271211809221730601602827122249238086030580971376104724987801049500689134122609834321586609223761140538079460830213824674361601046367637227094018381901291488659642720549583856812747877519600804325570421770575999289389175021646347371879234023647657507178519047236746071420327155188213839293382288787853777540226192644761028822256165706787395891134765908229036044468473519166141610604791485071702808854944672418124203289328124793348198048601338476086482318248264508789781967910205393740835345086784345145351367491197717933757414967811594913692588314161669333147733048171044386546892346475181197482702164468542430187885074163177843285948999943328049159021873821254267471067523609151007885131921896462161216356454116929796355815756642621369974260365378070336290542971599886325232821981080341858950609157813769416455337935096696635623426418166316737131174435618543058086342714723330814586496030805366321181723292731710369013923285787724941830672247377301048663929453294620044701627159066468762709113137517559435822623284148112827473010030736329596829357275518641576798298066541516764673029908084962144713

n = p ** 2 * q ** 3 * r ** 5 * s ** 7
d = libnum.invmod(e, (((p-1)*p)*((q-1)*q*q)*((r-1)*r*r*r*r)*((s-1)*s*s*s*s*s*s)))   #invmod(a, n) - 求a对于n的模逆,这里逆向加密过程中计算ψ(n)=(p-1)(q-1)，对ψ(n)保密,也就是对应根据ed=1modψ(n),求出d
m = pow(c, d, n)        # pow(x, y[, z])--函数是计算 x 的 y 次方，如果 z 在存在，则再对结果进行取模，其结果等效于 pow(x,y) %z，对应前面解密算法中M=D(C)=C^d(mod n)
#print(m) #明文的十进制格式
string = long_to_bytes(m)      # m明文，用长字节划范围
print(string.decode())
```

```
└─$ python 12.py
hgame{EulEr:fUNcTIon;iS.So*IMpORTaNt*In&RsA}
```

最终flag:

> hgame{EulEr:fUNcTIon;iS.So*IMpORTaNt*In&RsA}

.
.
解毕！
敬礼！