2022年HGAME中CRYPTO的RSA Attack2

原创

沐一 · 林 于 2022-04-04 16:21:26 发布 95 收藏

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/122773241

2022年HGAME中CRYPTO的RSA Attack2

RSA Attack 2[已完成]

描述

RSA 加密算法可安全啦，一般人可破解不了呢

可惜，再安全的算法也架不住使用的人“蠢”啊

小伙子我看你骨骼惊奇，下面这些人错误的使用了 RSA

快用你的密码学知识给他一个大嘴巴子，让他们长点教训

题目地址 https://cmfj-1308188104.cos.ap-shanghai.myqcloud.com/Week2/RSA%20Attack%202.zip

基准分数 250

当前分数 250

完成人数 185

CSDN @沐一 · 林

.

.

照例下载附件，一个 task.py，一个output.txt：

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

flag_parts = list(map(s2n, re.findall(rf".{{,{ceil(len(flag) / 3)}}}", flag)))

print("# task1")
m = flag_parts[0]
e = 65537
p = getPrime(1024)
q = getPrime(1024)
r = getPrime(1024)
n1 = p * q
c1 = pow(m, e, n1)
n2 = r * q
c2 = pow(m, e, n2)
print("e =", e)
print("n1 =", n1)
print("c1 =", c1)
print("n2 =", n2)
print("c2 =", c2)

print("# task2")
m = flag_parts[1]
e = 7
p = getPrime(1024)
q = getPrime(1024)
n = p * q
c = pow(m, e, n)
print("e =", e)
print("n =", n)
print("c =", c)

print("# task3")
m = flag_parts[2]
p = getPrime(1024)
q = getPrime(1024)
n = p * q
e1 = getPrime(32)
e2 = getPrime(32)
c1 = pow(m, e1, n)
c2 = pow(m, e2, n)
print("n =", n)
print("e1 =", e1)
print("c1 =", c1)
print("e2 =", e2)
print("c2 =", c2)
```

```
# task1
e = 65537
n1 = 14611545605107950827581005165327694782823188603151768169731431418361306231114985037775917461433925308054396
97080969080407398583537646462986060971029218136860061862659049849185040450344341424145548730444834489233787742246571570915423865350514160590418498531187376349576134572215528945788968601974666329372010687422732369928827779429220895717244652342059639111489155953781102947315012364162410810367651675444949280512664255275127830963484677763604211413599051624590751737732019009140072927730763672489059215525643799656616099545674301822501385193759388608612913135158295881100359644580606149295251385193223856362719455 3
```

```
c1 = 96507580355493298866427181643918380232881201369420374132076310537603691258499503164767234846811131042368085
81019906700670653062375961216648843536799876895323054378013469230701455241062713377706669476771157527249933073871
22132705797012726237073550669419110046308257408484535063515678066777681017211510981429273346928022971149411064556
22500128739914130613608172247107503242307969290838026716021414372051674800073498706868510467525441168700569031
21168249660368515682238288843351121446372680903971585329371411226540759527300523315739807011363782120029567192951
92733955673315234274064519957670199895100508623561838510479
n2 = 20937478725109983803079185450449616567464596961348727453817249035110047585580142823551289577145958127121586
79287850938608517845217111245589042947445779721920282703088426227306133475249349679793534663150980668558917961836
74539927497533182738341130162371206868805141104151136734311704889587302039634894554189675441286192343949158203929
08422974075932751838012185542968842691824203206517795693893863945100661940988455695923511777306566419373394091
90734943168664648551632557549490268233751843804271129643751322144839703481309927920395553502593912013968060449548
69807659108924382849454507333751569338631508083697968308923631
c2 = 11536506945313747180442473461658912307154460869003392732178457643224057969838224601059836860883718459986003
10697037577844372574860708562093878771408132131581714441411558995223749244848343891037886535923957516932611666803
04632758176098276260489623045933244795464534718810999766444108896572483460389868364617797801834116862607567767117
20577053319504691373550107525296560936467435283812493396486678178020292433365898032597027338876045182743492831
81417567383419834533751406559639647770983986838726584043032298394590646464682447043778327160749908979186939859055
73147130946742082617612998947057725134409481394290114259480900
# task2
e = 7
n = 14157878492255346300993349653813018105991884577529909522555551468374307942096214964604172734381913051273745228293930832314483466922529240958994897697475939867025561348042725919663546949015024693952641936481841552751484604123097148071800416608762258562797116583678332832016172177459664959920497625303735311638219796273612009215442235781707187413482420121641155937777009039544091031100929215788210489333468932128050716822355758137241139783415928859577673775874922027401859708286297675016621953562768625850259136159108396798606699172552717344138652113401265441997606284450541316614841848766796269463607530095126343495370
c = 10262871020519116406312674685238364023536657841034751572844570983750295909492149101500869806418603732181350082576447594766587572350246675445508931577670158295558641219582729345581697448231116318080456112516700717984731655900726388185866905989088504004805024490513718243036445638662260558477697146032055765285263446084259814560197549018044099935158351931885157616527235283229066145390964094929007056946332051364474528453970904251050605631514869007890625
# task3
n = 18819509188106230363444813350468162056164434642729404632983082518225388069544777374544142317612858448345344137372222988033366528086236635213756227816610865045924357232188768913642158448603346330462535696121739622702200540344105464126695432011739181531217582949804939555720700457350512898322376591813135311921904580338340203569582681889243452495363849558955947124975293736509426400460083981078846138740050634906824438689712748324336878791622676974341814691041262280604277357889892211717124319329666052810029131172229930723477981468761369516771720250571713027972064974999802168017946274736383148001865929719248159075729
e1 = 2519901323
c1 = 32307797262255448725314411690093070720737545787618883879834032063645484514967365139054603819079281073100300863465893511058090285996503035396075814076278197979443373986014005105609924624550484513265939935950898001503429990218747347480666929623626505400360020737487665093476498181393043639140838799189298735777063235996280316186417930740183045212434604875513648232996850525188526857066878002095052774268691400510569962428821326162566951888707826343103629731537666982862589468968663966708724518031142808467095727797805584822233937594759991036077045106183322537105038575610256136325926829315522281501714238462038753448700
e2 = 3676335737
c2 = 94081859562227916143983671964170784679029465088879982233500738585416673645928312943476906299512237107363678537180085763384137913976109189042613798111308751993485466377669594448943038566301171391702257434238015571831779420498862611636286514412513662472278230945545225775880817241588440390984065155448536430923785388525187694147709800869038960054439899866963596249598973602102071539641537589072033569750483704518862610314220447494275141081946637943709156961029457568779306094552510898666085127747507999446647485911409264379741892764572643017592824747688487981703434665256011659796519120406105140191628281488668846786
```

.

.

首先对 flag_parts = list(map(s2n, re.findall(rf".{{,{ceil(len(flag) / 3)}}}", flag))) 不是很熟悉，尝试分解一下：

```python
import re
from math import ceil
from Crypto.Util.number import getPrime
from libnum import s2n
flag="hgame{welc0me_to_4sm_w0rld}"  #自己给个flag来测试
print(len(flag))
print(len(flag)/3)
print(ceil(len(flag) / 3))    #ceil是下取整
print(rf".{{,{ceil(len(flag) / 3)}}}") #这里{{}}有点像python模板的语法，结果是{,9},好像是取开头到第9位的，像字符串运算符a[,9]
print(re.findall(rf".{{,{ceil(len(flag) / 3)}}}", flag))  #在字符串中找到正则表达式所匹配的所有子串，并返回一个列表
print(s2n('hgame{wel'),s2n('c0me_to_4'),s2n('sm_w0rld}'))
print(map(s2n, re.findall(rf".{{,{ceil(len(flag) / 3)}}}", flag)))   #map生成的是一个迭代器的对象
print(list(map(s2n, re.findall(rf".{{,{ceil(len(flag) / 3)}}}", flag))))  #list用来遍历map的迭代器
flag_parts = list(map(s2n, re.findall(rf".{{,{ceil(len(flag) / 3)}}}", flag)))
print(flag_parts)
```

```
27
9.0
9
.{,9}
['hgame{wel', 'c0me_to_4', 'sm_w0rld}', '']
192591073920707142590018297172200441773668362129256717399481934973
<map object at 0x7fd347341f10>
[192591073920707142590018297172200441773668362129256717399481934973, 0]
[192591073920707142590018297172200441773668362129256717399481934973, 林]
```


.

.

所以总的流程是把flag分成3部分分别加密，由于看起来是没怎么变形的RSA，所以试试 CTF-RSA-tool 工具先：

```
#task1
e = 65537
n = 14611545605107950827581005165327694782823188603151768169731431418361306231114985037775917461433925308054396970809690804073985835376464629860609710292181368600618626590498491850404503443414241455487304448344892337877422465715709154238653505141605904184985311873763495761345722155289457889686019746663293720106874227323699288277942922089571724465234205963911148915595378110294731501236416241081036765167544494928051266425527512783096348467776360421141359905162459075173773201900914007292773076367248905921552564379965661609954567430182250138519375938860861291313515829588110035964458060614929525138519322385636271945 53
n = 20937478725109983803079185450449616567464596961348727453817249035110047585580142823551289577145958127121586792878509386085178452171112455890429474457797219202827030884262273061334752493496797935346631509806685589179618367453992749753318273834113016237120686880514110415113673431170488958730203963489455418967544128619234394915820392908422974075932751838012185542968842691824203206517795693893863945100661940988455695923511777306566419373394091907349431686664685516325575494902682337518438042711296437513221448397034813099279203955535025939120139680604495486980765910892438284945450733375156933863150808369796830892363
c = 96507580355493298866427181643918380232881201369420374132076310537603691258499503164767234846811131042368085810199067006706530623759612166488435367998768953230543780134692307014552410627133777066694767711575272499330738712213270579701272623707355066941911004630825740848453506351567806677768101721151098142927334692802297114941106455622500128739914130613608172247107503242307969290838026716021414372051674800073498706868510467525441168700569031211682496603685156822382888433511214463726809039715853293714112265407595273005233157398070113637821200295671929519273395567331523427406451995767019989510050862356183851047 9
```

```
#task2
e = 7
n = 14157878492255346300993349653813018105991884577529909522555551468374307942096214964604172734381913051273745228293930832314483466922529240958994897697475939867025561348042725919663546949015024693952641936481841552751484604123097148071800416608762258562797116583678332832015617217745966495992049762530373531163821979627361200921544223578170718741348242012164115593777700903954409103110092921578821048933346893212805071682235575813724113978341592885957767377587492202740185970828629767501662195356276862585025913615910839679860669917255271734413865211340126544199760628445054131661484184876679626946360753009512634349537
c = 10262871020519116406312674685238364023536657841034751572844570983750295909492149101500869806418603732181350082576447594766587572350246675445508931577670158295558641219582729345581697448231116318080456112516700717984731655900726388185866905989088504004805024490513718243036445638662260558477697146032055765285263446084259814560197549018044099935158351931885157616527235283229066145390964094929007056946332051364474528453970904251050605631514869007890625

#task3
n = 18819509188106230363444813350468162056164434642729404632983082518225388069544777374544142317612858448345344137372222988033366528086236635213756227816610865045924357232188768913642158448603346330462535696121739622702200540344105464126695432011739181531217582949804939555720700457350512898322376591813135311921904580338340203569582681889243452495363849558955947124975293736509426400460083981078846138740050634906824438689712748324336878791622676974341814691041262280604277357889892211717124319329666052810029131172229930723477981468761369516771720250571713027972064974999802168017946274736383148001865929719248159075729
e = 2519901323
c = 32307797262255448725314411690093070720737545787618883879834032063645484514967365139054603819079281073100300863465893511058090285996503035396075814076278197979443373986014005105609924624550484513265939935950898001503429990218747347480666929623626505400360020737487665093476498181393043639140838799189298735777063235996280316186417930740183045212434604875513648232996850525188526857066878002095052774268691400510569962428821326162566951888707826343103629731537666982862589468968663966708724518031142808467095727797805584822233937594759991036077045106183322537105038575610256136325926829315522281501714238462038753448 70
e = 3676335737
c = 940818595622279161439836719641707846790294650888799822335007385854166736459283129434769062995122371073636785371800857633841379139761091890426137981113087519934854663776695944489430385663011713917022574342380155718317794204988626116362865144125136624722782309455452257758808172415884403909840651554485364309237853885251876941477098008690389600544398998669635962495989736021020715396415375890720335697504837045188626103142204474942751410819466379437091569610294575687793060945525108986660851277475079994466474859114092643797418927645726430175928247476884879817034346652560116597965191204061051401916282814886688467861
```

.

.

除了第一个模不互素只需要一个 C 外，其它可以直接用工具解出：

```
└─$ python2 solve.py --verbose  -i /home/wdnmd/桌面/2.txt
INFO: Here are your plain text:
hgame{RsA@hAS!a&VArIETY?of.

┌──(wdnmd㉿kali)-[~/桌面/CTF-RSA-tool]
└─$ python2 solve.py --verbose  -i /home/wdnmd/桌面/3.txt
INFO: start Hastad attack. If there was no result after a long time. Press ctrl+c to stop, and try other ways.
INFO: Here are your plain text:
AttacK^mEThodS^whAT:other!A

┌──(wdnmd㉿kali)-[~/桌面/CTF-RSA-tool]
└─$ python2 solve.py --verbose  -i /home/wdnmd/桌面/4.txt
INFO: Here are your plain text:
ttACK|METHOdS~do@you_KNOW}
```


.

.

最终flag：

```
hgame{RsA@hAS!a&VArIETY?of.AttacK^mEThodS^whAT:other!AttACK|METHOdS~do@you_KNOW}
```

.

.

解毕！

敬礼！