长城杯2021 Crypto writeup

原创

M@ku1i 于 2021-10-02 15:10:22 发布 126 收藏 1

文章标签： 密码学

本文链接：https://blog.csdn.net/weixin_56678592/article/details/120586024

长城杯2021 Crypto writeup

#1 baby_rsa:

题目：

```python
#!/usr/bin/env python3

from Crypto.Util.number import *
from secret import flag, v1, v2, m1, m2


def enc_1(val):
    p, q = pow(v1, (m1+1))-pow((v1+1), m1), pow(v2, (m2+1))-pow((v2+1), m2)
    assert isPrime(p) and isPrime(q) and (p*q).bit_length() == 2048 and q < p < q << 3
    return pow(val, 0x10001, p*q)


def enc_2(val):
    assert val.bit_length() < 512
    while True:
        fac = [getPrime(512) for i in range(3)]
        if isPrime(((fac[0]+fac[1]+fac[2]) << 1) - 1):
            n = fac[0]*fac[1]*fac[2]*(((fac[0]+fac[1]+fac[2]) << 1) - 1)
            break
    c = pow(val, 0x10001, n)
    return (c, n, ((fac[0]+fac[1]+fac[2]) << 1) - 1)


if __name__ == "__main__":
    assert flag[:5] == b'flag{'
    plain1 = bytes_to_long(flag[:21])
    plain2 = bytes_to_long(flag[21:])
    print(enc_1(plain1))
    print(enc_2(plain2))

'''
158087739211657463782246495540327740951985317824559041695522233035139409682928968141592884174992207398758337545739436070478552567399761615985999039329811699795098715919999648568069295978059041340999018268583677783863423767685080315548022490750723667100388893062688067441790866486847380230734589829060669723404143989284111479705939352440779254487327724736197830793283515222691708798070641113188710742910735813430393895611753910397669363762678751845816433359160494617847533411152275151635457094547462725148270006018537353565514956852299956374835067354489006568853653534343086394120350031195166933033770815769755409483110
(406259810172502629452305484507389517255665202521634101245656221267547396936812716491271041090381648527877672964036974624754596705408458221503976399230132231029126747484024275015880188664908783946784820615615212533655500290755655079882327290320552989927927125745697048460755146248246541276917439441120757038140436225995304961007133786967618799825426799176315704510721078933487928173216525934717949742271834767329806238354839910670803451849784821913424306274903985169127144519841529603488995895327519192725830987641181610560785367813417501425531970829250707301780925613144005181510199551049897909114603578483660162630830, 430017260469550789813440169817904459801990720660193233820682441428889315396028123180230952564749396972578026461503485467796475451522881586075552393028876891376457486284212476852254633461180812387180497013207262954353767332156814157742552584194186614660104039285912429614341787308465374712361426835173991094664297763773601183551734310161075439772413580640931027418196261634671398333524540944722293495984793583672034524526068337964831118920763437459583949321321994427180487206335563104670192224346937854239966563066122627146090761196348147834381118437736495191011693260727935960275940579883651330370411335661468978682690, 397962725923318964006267849517132395268572731687321330466675723996226603305878815793193140945570115548518730683890166290859630861361164253525359025983787390)
'''
```

分析题目：

flag分成两段加密，前半段交给enc_1函数加密，后半段交给enc_2函数加密。

enc_1:

```
def enc_1(val):
    p, q = pow(v1, (m1+1))-pow((v1+1), m1), pow(v2, (m2+1))-pow((v2+1), m2)
    assert isPrime(p) and isPrime(q) and (p*q).bit_length() == 2048 and q < p < q << 3
    return pow(val, 0x10001, p*q)
```

有以下式子和条件：

$$\begin{cases} p = v_1^{m_1+1} - (v_1+1)^{m_1} \\ q = v_2^{m_2+1} - (v_2+1)^{m_2} \end{cases} \quad \begin{cases} p \quad is \quad prime \\ q \quad is \quad prime \\ bit\_length(p*q) = 2048 \\ q < p < 8*q \end{cases}$$

p和q的素数生成过程是相同的。使用两组秘密数v1，v2和m1，m2，使得$v1^{(m1+1)-(v1+1)}m1$和$v2^{(m2+1)-(v2+1)}m2$是素数。

p，q的比特长度均小于2048（应该更小）。我们可以通过这个条件来枚举v和m，求出该范围内的所有素数，然后再从中选取任意两个素数，求n，长度若为2048bits，则尝试解密。

exp（enc_1）：

```
from Crypto.Util.number import long_to_bytes
from gmpy2 import invert, is_prime
from tqdm import tqdm

primes = []

for v1v2 in tqdm(range(500)):
    for m1m2 in range(500):
        prime = v1v2**(m1m2+1) - (v1v2+1)**m1m2
        if prime.bit_length() > 2048: break
        if is_prime(prime):
            primes.append(prime)

val1 = 158087739211657463782246495540327740951985317824559041695522233035139409682928968141592884174992207398758
337545739436070478552567399761615985999039329811699795098715919999648568069295978059041340999018268583677783863 4
237676850803155480224907507236671003888930626880674417908664868473802307345898290606697234041439892841114797059 3
935244077925448732772473619783079328351522269170879807064111318871074291073581343039389561175391039766936376267 8
751845816433359160494617847533411152275151635457094547462725148270006018537353565514956852299956374835067354489 0
065688536535343430863941203500311951669330337708157697554094831 1
```

```
for i in range(len(primes)):
    for j in range(i, len(primes)):
        pq = primes[i]*primes[j]
        if len(bin(pq)[2:]) == 2048:
            try:
                d = invert(0x10001, (primes[i]-1)*(primes[j]-1))
                dec = long_to_bytes(pow(val1, d, pq))
                if b'flag' in dec:
                    print(dec)
            except ValueError:
                pass
  #flag1 = flag{8102c552-3d78-4a
```

enc_2:

```
def enc_2(val):
    assert val.bit_length() < 512
    while True:
        fac = [getPrime(512) for i in range(3)]
        if isPrime(((fac[0]+fac[1]+fac[2]) << 1) - 1):
            n = fac[0]*fac[1]*fac[2]*(((fac[0]+fac[1]+fac[2]) << 1) - 1)
            break
    c = pow(val, 0x10001, n)
    return (c, n, ((fac[0]+fac[1]+fac[2]) << 1) - 1)
```

n是由三个未知素数构成的，我们将他们表示为f0,f1,f2,f3。

f0=fac[0],f1=fac[1],f2=fac[2],f3=((fac[0]+fac[1]+fac[2]) << 1) - 1

由于flag位数小于512位，而f3的位数不小于513位，尝试分解f3，得到四个因子，求欧拉函数phi，解密即可。

exp（enc_2）：

```
from Crypto.Util.number import *
from tqdm import *
import gmpy2
c = 40625981017250262945230548450738951725566520252163410124565622126754739693681271649127104109038164852787767296403697462475459670540845822150397639923013223102912674748402427501588018866490878394678482061561521253365550029075565507988232729032055298992792712574569704846075514624824654127691743944112075703814043622599530496100713378696761879982542679917631570451072107893348792817321652593471794974227183476732980623835483991067080345184978482191342430627490398516912714451984152960348899589532751919272583098764118161056078536781341750142553197082925070730178092561314400518151019955104989790911460357848366016263083
n = 43001726046955078981344016981790445980199072066019323382068244142888931539602812318023095256474939697257802646150348546779647545152288158607555239302887689137645748628421247685225463346118081238718049701320726295435376733215681415774255258419418661466010403928591242961434178730846537471236142683517399109466429776377360118355173431016107543977241358064093102741819626163467139833352454094472229349598479358367203452452606833796483111892076343745958394932132199442718048720633556310467019222434693785423996656306612262714609076119634814783438111843773649519101169326072793596027594057988365133037041133566146897868269
f3 = 39796272592331896400626784951713239526857273168732133046667572399622660330587881579319314094557011554851873068389016629085963086136116425352535902598378739
p = [191,193,627383,172075473847731712775868228546503193989105983587397515755503132707011112362878983329943354966961932516067971935533818787775831148578519749271049 1]
phi=1
for i in tqdm(p):
    phi*=i-1
d=gmpy2.invert(0x10001,phi)
print(long_to_bytes(gmpy2.powmod(c,d,f3)))
#flag2 = 42-b659-0c96ef827f05
```

拼接得flag：flag{8102c552-3d78-4a42-b659-0c96ef827f05}