WhitegiveCMA-writeup

原创

网安小小白 于 2022-03-20 17:26:33 发布

分类专栏： CTF 文章标签： 信息安全 RSA CTF

本文链接：https://blog.csdn.net/u013671216/article/details/123616887

WhitegiveCMA

题目链接，提取码：GAME

理论分析

下载后是一个文件

```
from Crypto.Util.number import *
from secret import flag
p = getPrime(512)
q = getPrime(512)
m = bytes_to_long(flag)
e10 = 2021*p-529*q+999
e16 = 0x2021*p-0x529*q-0xfff
N = p*q
c10 = pow(m,e10,N)
c16 = pow(m,e16,N)
print(N)
print(c10)
print(c16)
#105505147972090189190855626395970289980349478849272766172689292153677504113157945203168037148505500911004204286230594141359278451217226
#445948307057268334845424460216266874687933154464404913002425621323835722148383833968930451702515248228791781164491435715228808433728755
#447140550386082556612170091181492308333237954744898538256008308971163762071028459464057932871567284756647268019696085997341057785463172
```

给出了两个e，考虑共模攻击。

为了方便后续阅读，做如下变形：

$e_1 =$

$e_2 =$

首先通过变换获取只与p，q相关的密文形式，记为c_1, c_2

$c_1 =$ $-b$

$c_2 =$ a

令

$w =$

$e = b * d$,

$e =$

则

$c = m$ $+ a \ mod N$

$c = m$ $+ a \ mod N$

根据欧拉定理知道：m $\equiv 1 mod p$,m $\equiv 1 mod q$

所以$c \ mod p =$

令

$k = c$

$k = c$

如果知道$k = m$

则由

$k = x * n +$

所以

$c = k$

把两个方程看为两个多项式，第一个$k = (k +$ $) *$

一般化:

$(a * k + b) * (a * k +$ $)$

$(a * b +$

只要确定a ,b ,a ,b 即可求出解，

```
def mul(a1,b1,a2,b2):
  return (a1*b2+a2*b1+a1*a2*(Kp+Kq))%N,(b1*b2-a1*a2*Kp*Kq)%N
```

根据上述等式用多项式的二分乘法把pow(K,e0,N)化成关于K的一次多项式

```
a0 = 1
b0 = 0
a = 0
b = 1
while(e0):
  if(e0%2):
    a,b = mul(a,b,a0,b0)
  a0,b0 = mul(a0,b0,a0,b0)
  e0 = e0//2
```

化为一次多项式后，有$a*k+b=c_0 \pmod N$即可求出k

```
K = ((c0-b)*inverse(a,N))%N
```

根据上面方程可以求出p,q。最后求出结果

完整代码

```
import binascii
import gmpy2

# 数据
N = 10550514797209018919085562639597028998034947884927276617268929215367750411315794520316803714850550091100420428623059414135927845121722622046419957871131787052400939992515933008526889161170273269372518421947912623513129454878711970654190128997031350051367334782684375875174858098941827822652015693662692432705911
c10 = 44594830705726833484542446021626687468793315446440491300242562132383572214838383396893045170251524822879178116449143571522880843372875585376945929248227672182668860765415910480567064826316000549996363948338866696321566948086096629421501295545617054417677393365598002537621340410326525247666639842346491047367
c16 = 44714055038608255661217009118149230833323795474489853825600830897116376207102845946405793287156728475664726801969608599734105778546317268118984070287308254520723161252793270372766239693849125114583526383289679577470203270954263611671265610573237844472924495914342697052049517874123620614228215081764500905625
a1 = 2021
b1 = -529
d1 = 999
a2 = 0x2021
b2 = -0x529
d2 = -0xfff

# 获取cp和cq。
cp = gmpy2.powmod(c16,-b1,N)*gmpy2.powmod(c10,b2,N)%N
cq = gmpy2.powmod(c10,-a2,N)*gmpy2.powmod(c16,a1,N)%N
w = a1*b2-b1*a2
ep = b2*d1-b1*d2+w
eq = a1*d2-a2*d1+w

Kp = gmpy2.powmod(cp,eq,N)
Kq = gmpy2.powmod(cq,ep,N)

c0 = gmpy2.powmod(cp*cq,ep*eq,N)
e0 = w*N+ep+eq-w

def mul(a1,b1,a2,b2):
    return (a1*b2+a2*b1+a1*a2*(Kp+Kq))%N,(b1*b2-a1*a2*Kp*Kq)%N

# 把pow(K,e0,N)化成关于K的一次多项式。
a0 = 1
b0 = 0
a = 0
b = 1
while(e0):
    if(e0%2):
        a,b = mul(a,b,a0,b0)
    a0,b0 = mul(a0,b0,a0,b0)
    e0 = e0//2

# 得到关于K的一元一次方程，求解
K = ((c0-b)*gmpy2.invert(a,N))%N

# 再分解N
p = gmpy2.gcd(Kp-K,N)
```

```
p = gmpy2.gcd(Kp-K,N)
q = gmpy2.gcd(Kq-K,N)

# 解密
e10 = a1*p+b1*q+d1
e16 = a2*p+b2*q+d2
d = gmpy2.invert(e16,(p-1)*(q-1))
m = gmpy2.powmod(c16,d,N)
flag = binascii.unhexlify(hex(m)[2:])
print(flag)
```