ASISCTF

原创

逃课的小学生 于 2021-10-31 23:40:16 发布 62 收藏 1
分类专栏： crypto ctf 文章标签： ctf crypto

warm up

题目代码如下图所示，我们会发现整个加密过程如下所示，先在flag后面加上p长度的随机字符，然后选择pow(s,i,p)的结果选择字符重新连接字符串实现加密

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

from Crypto.Util.number import *
import string
from secret import is_valid, flag

def random_str(l):
 rstr = ''
 for _ in range(l):
  rstr += string.printable[:94][getRandomRange(0, 93)]
 return rstr

def encrypt(msg, nbit):
 l, p = len(msg), getPrime(nbit)
 rstr = random_str(p - l)
 msg += rstr
 while True:
  s = getRandomNBitInteger(1024)
  if is_valid(s, p):
   break
 enc = msg[0]
 for i in range(p-1):
  enc += msg[pow(s, i, p)]
 return enc

nbit = 15
enc = encrypt(flag, nbit)
print(f'enc = {enc}')
```

我们发现pow(s,i,p)=2,那么pow(s,2*i,p)=4，所以如果相差一倍的位置上的字符是I和{，我们可以确定i，2*i，从而对s求解，当算出s后即将源字符串还原。由于s和p较小，可以使用爆破s的方式，发现还原字符串开头为ASIS{且flag为规定字符即可找到s和答案

```
import gmpy2
import string
quanbu=string.ascii_letters + string.digits+"_-]+.!?|"
f=open("output.txt","r")
enc=f.read()
f.close()
enc=enc[7:]
p=len(enc)+1
print(p)

for i in range(len(enc)):
 if enc[i]=="I" and enc[(2*i)%(p-1)]=="{":
  print(i)


i=10778//2
#i=3758//2
ini=gmpy2.invert(i,p-1)
s2=pow(2,ini,p)


s=[]
for zhong in range(p):
 zhongjian=pow(zhong,2,p)
 if zhongjian==s2:
  s.append(zhong)
print(s)


flag=["A"]*p
for j in range(p-1):
 ms=pow(s[0],j,p)
 flag[ms]=enc[j]
if flag[1]=="S" and flag[2]=="I" and flag[3]=="S" and flag[4]=="{":
 print(flag[:30])


flag=["A"]*p
for j in range(p-1):
 ms=pow(s[1],j,p)
 flag[ms]=enc[j]
if flag[1]=="S" and flag[2]=="I" and flag[3]=="S" and flag[4]=="{":
 print(flag[:30])
```

Spiritual

题目给了nc的接口，其中给出椭圆曲线的p和椭圆曲线的阶，计算当模p^k的时候椭圆曲线的阶是多少。我们发现在sage中椭圆曲线函数有一个extension_degree函数可以快速计算这样的问题，但我们并不知道椭圆曲线的参数，无法还原椭圆曲线，所以我们借助set_order函数强行赋值给椭圆曲线阶，从而完成后面的计算

```
p = 126479891943345541230373341807457340927

k = 126479891943345541240647426557172181202

n=7

E = EllipticCurve(GF(p),[10000,10000])
E.set_order(k,num_checks=0)
print(E.order(extension_degree=n))
```

Madras

题目如下，给出一组方程结果，求解方程后即可使用RSA通用求解方式计算

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

from Crypto.Util.number import *
from flag import FLAG

def gentuple(nbit):
 a, b, c = [getPrime(nbit // 3) for _ in '012']
 return a, b, c

def encrypt(msg, params):
 a, b, c = params
 e, n = 65537, a * b * c
 m = bytes_to_long(msg)
 assert m < n
 enc = pow(m, e, n)
 return enc

nbit = 513
a, b, c = gentuple(nbit)
enc = encrypt(FLAG, (a, b, c))

print(f'a*b + c = {a*b + c}')
print(f'b*c + a = {b*c + a}')
print(f'c*a + b = {c*a + b}')
print(f'enc % a = {enc % a}')
print(f'enc % b = {enc % b}')
print(f'enc % c = {enc % c}')
print(f'enc     = {enc}')
```

这里我们使用sage做求解，再解RSA

```
n1=455335299459612190471911809531430557474489899674861766264573043429167196471180026265692731161274171590 2
n2=441418714838434827803117286571594239778600312504735343641895267998067761701648492704519545039272311040 2
n3=262133149779799868008784142501188122628334200802251163811601367617539338709578751229100854127135577280 2
a, b, c = var('a b c')
print(solve([a*b+c-n1,b*c+a-n2,c*a+b-n3],a,b,c))
```

```
import gmpy2
a = 164437650133676186953391452799914031694646700547921l
b = 27690452830568715591082376398326529111140080815766l1
c = 15941188016655805106155412225275917078349320582135 41
n1 = 455335299459612190471911809531430557474489899674861766264573043429167196471180026265692731161274171590
n2 = 441418714838434827803117286571594239778600312504735343641895267998067761701648492704519545039272311040
n3 = 262133149779799868008784142501188122628334200802251163811601367617539338709578751229100854127135577280
n4 = 123569109825390386847092952004245363125004276902996 8
n5 = 223572750583541515747285668796036521662605834354657 2
n6 = 119797693364816372260960177240289584409386658977772 1
enc     = 623854889789791246270851438210638730598437811313219298035369574691288239999128526893754894983550 0

assert a*b+c==n1
assert b*c+a==n2
assert c*a+b==n3
assert enc%a==n4
assert enc%b==n5
assert enc%c==n6

print(gmpy2.is_prime(a))
print(gmpy2.is_prime(b))
print(gmpy2.is_prime(c))
n=a*b*c
phi=(a-1)*(b-1)*(c-1)
e=65537
d=gmpy2.invert(e,phi)
m=pow(enc,d,n)
print(hex(m)[2:].decode("hex"))
```