长安“战疫”网络安全卫士守护赛crypto

原创

ephemeral-fever 于 2022-03-24 22:44:03 发布 6677 收藏
分类专栏： Crypto 文章标签： 安全

Crypto 专栏收录该内容
10 篇文章 0 订阅
订阅专栏

cry1

题目如下，就是一个脚本：

```python
from Crypto.Util.number import*
from secret import flag,key

assert len(key) <= 5
assert flag[:5] == b'cazy{'
def can_encrypt(flag,key):
    block_len = len(flag) // len(key) + 1
    new_key = key * block_len
    return bytes([i^j for i,j in zip(flag,new_key)])

c = can_encrypt(flag,key)
print(c)

# b'<pH\x86\x1a&"m\xce\x12\x00pm\x97U1uA\xcf\x0c:NP\xcf\x18~l'
```

> **zip()** 函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表。
> 如果各个迭代器的元素个数不一致，则返回列表长度与最短的对象相同，利用 * 号操作符，可以将元组解压为列表。

这道题就是一个简单的异或，从can_encrypt()函数中能看出来flag的每一位都和密钥进行了异或，只要再异或回去即可得到flag，即a^b=c能推得a=b^c

提示里给出了flag的前5位，并且密钥的长度小于等于5，再根据密文的前5位，将他们异或，即可得到密钥

根据密文的长度27，密钥的长度5还原出block_len为6

解题脚本如下：

```python
from Crypto.Util.number import*
#from secret import flag,key

#assert len(key) <= 5
#assert flag[:5] == b'cazy{'
def can_encrypt(flag,key):
    block_len = len(flag) // len(key) + 1
    new_key = key * block_len
    return bytes([i^j for i,j in zip(flag,new_key)])


c=b'<pH\x86\x1a&"m\xce\x12\x00pm\x97U1uA\xcf\x0c:NP\xcf\x18~l'
clue='cazy{'
key=[]
flag=''

for i in range(5):
    key.append(ord(clue[i])^c[i])
key=key*6

for i in range(27):
    flag+=chr(c[i]^key[i])
print(flag)
#cazy{y3_1s_a_h4nds0me_b0y!}
```

cry2

题目如下：

```python
import random
from Crypto.Util.number import long_to_bytes
from Crypto.Cipher import AES
from secret import flag

assert flag[:5] ==b'cazy{'

def pad(m):
    tmp = 16-(len(m)%16)
    return m + bytes([tmp for _ in range(tmp)])

def encrypt(m,key):
    aes = AES.new(key,AES.MODE_ECB)
    return aes.encrypt(m)

if __name__ == "__main__":
    flag = pad(flag)
    key = pad(long_to_bytes(random.randrange(1,1<<20)))
    c = encrypt(flag,key)
    print(c)
# b'\x9d\x18K\x84n\xb8b|\x18\xad4\xc6\xfc\xec\xfe\x14\x0b_T\xe3\x1b\x03Q\x96e\x9e\xb8MQ\xd5\xc3\x1c'
```

先理解题目意思，pad函数将m的长度补成16的整数倍，encrypt函数调用了aes加密。算一下1<<20为1048576不是很大，可以爆破，同样调用库里的aes解密函数即可，通过前5位来判断是否为目的flag。

解题脚本：

```
import random
from Crypto.Util.number import long_to_bytes
from Crypto.Cipher import AES
#from secret import flag

#assert flag[:5] ==b'cazy{'

def pad(m):
    tmp = 16-(len(m)%16)
    return m + bytes([tmp for _ in range(tmp)])

def encrypt(m,key):
    aes = AES.new(key,AES.MODE_ECB)
    return aes.encrypt(m)

c=b'\x9d\x18K\x84n\xb8b|\x18\xad4\xc6\xfc\xec\xfe\x14\x0b_T\xe3\x1b\x03Q\x96e\x9e\xb8MQ\xd5\xc3\x1c'
for i in range(1<<20):
    key = pad(long_to_bytes(i))
    aes=AES.new(key,AES.MODE_ECB)
    plain=aes.decrypt(c)
    if plain[:5]==b'cazy{':
        print(plain)

#cazy{n0_c4n,bb?n0p3!}
```

cry3

题目如下：

```
from Crypto.Util.number import*
from secret import flag

assert len(flag) <= 80
def sec_encry(m):
    cip = (m - (1<<500))**2 + 0x0338470
    return cip

if __name__ == "__main__":
    m = bytes_to_long(flag)
    c = sec_encry(m)
    print(c)

# 10715086071862673209484250490600018105614048117055336074437503883703510511248211671489145400471130049712947188505612184220711949974689275316345656079538583389095869818942817127245278601695124271626668045250476877726638182396614587807925457735428719972874944279172128411500209111406507112585996098530169
```

没啥好说的，按照他的操作逆过来就行

有一点需要注意，python内置的函数用来开方精度不够，需要用gmpy2里的iroot函数：

> gmpy2.iroot(x,n)：x开n次根，返回两个参数，第一个为开方结果，第二个布尔参数，表示是否能开尽

解题脚本：

```
from Crypto.Util.number import*
from gmpy2 import *
#from secret import flag

#assert len(flag) <= 80
def sec_encry(m):
    cip = (m - (1<<500))**2 + 0x0338470
    return cip

c=10715086071862673209484250490600018105614048117055336074437503883703510511248211671489145400471130049712947188
5056121842207119499746892753163456560795385833890958698189428171272452786016951242716266680452504768777266381823
96614587807925457735428719972874944279172128411500209111406507112585996098530169
c-=0x0338470
assert iroot(c,2)[1]
c=iroot(c,2)[0]
m=-1*c+(1<<500)
print(long_to_bytes(m))
#cazy{1234567890_no_m4th_n0_cRy}
```

cry4

题目如下：

```
from Crypto.Util.number import*
from secret import flag
assert flag[:5] == b'cazy{'
assert flag[-1:] == b'}'
flag = flag[5:-1]
assert(len(flag) == 24)

class my_LCG:
    def __init__(self, seed1 , seed2):
        self.state = [seed1,seed2]
        self.n = getPrime(64)
        while 1:
            self.a = bytes_to_long(flag[:8])
            self.b = bytes_to_long(flag[8:16])
            self.c = bytes_to_long(flag[16:])
            if self.a < self.n and self.b < self.n and self.c < self.n:
                break

    def next(self):
        new = (self.a * self.state[-1] + self.b * self.state[-2] + self.c) % self.n
        self.state.append( new )
        return new

def main():
    lcg = my_LCG(getRandomInteger(64),getRandomInteger(64))
    print("data = " + str([lcg.next() for _ in range(5)]))
    print("n = " + str(lcg.n))

if __name__ == "__main__":
    main()

# data = [2626199569775466793, 8922951687182166500, 454458498974504742, 7289424376539417914, 8673638837300855396]
# n = 10104483468358610819
```

根据题目意思，flag被拆成了三部分，在next函数里线性递推，已知n和递推出来的五个new值，根据题目意思我们可以得到如下的同余式组：

' ' ' ' { ' ' ' ' (

由于seed2和seed1未知，我们只考虑下面三个方程，并且下面三个方程也只有三个未知数，由此断定方程有解。

我们把方程组写成矩阵形式如下：

(new4 new5) = (new3 new4
new2 new3

解出a，b，c如下：

(bc) (new3 new4
new2 new3

由于python的numpy精度不够，所以我们用sagemath，下面给出一点基本使用方法：

> SageMath矩阵操作及解线性方程组_m0_46161993的博客-CSDN博客_sagemath 矩阵
> SageMath常用函数_panfengblog-CSDN博客_sagemath

解出a，b，c后写个脚本flag就出来了

解题脚本：

```python
from Crypto.Util.number import*

data = [2626199569775466793, 8922951687182166500,
       454458498974504742, 7289424376539417914, 8673638837300855396]
n = 10104483468358610819
a,b,c=5490290802446982981,8175498372211240502,6859390560180138873
flag=long_to_bytes(a)+long_to_bytes(b)+long_to_bytes(c)
print(b'cazy{'+flag+b'}')
#cazy{L1near_Equ4t1on6_1s_34sy}
```

cry5

题目如下：

```
pinvq:0x63367a2b947c21d5051144d2d40572e366e19e3539a3074a433a92161465543157854669134c03642a12d304d2d9036e6458fe4c85
0c772c19c4eb3f567902b3qinvp:0x79388eb6c541fffefc9cfb083f3662655651502d81ccc00ecde17a75f316bc97a8d888286f21b1235bde1f35ef
e13f8b3edb739c8f28e6e6043cb29569aa0e7bc:0x5a1e001edd22964dd501eac6071091027db7665e5355426e1fa0c6360accbc013c7a36da8
8797de1960a6e9f1cf9ad9b8fd837b76fea7e11eac30a898c7a8b6d8c8989db07c2d80b14487a167c0064442e1fb9fd657a519cac5651457d642
23baa30d8b7689d22f5f3795659ba50fb808b1863b344d8a8753b60bb4188b5e386e:0x10005d:0xae285803302de933cfc181bd4b9ab2ae09d
1991509cb165aa1650bef78a8b23548bb17175f10cddffcde1a1cf36417cc080a622a1f8c64deb6d16667851942375670c50c5a32796545784f0
bbcfdf2c0629a3d4f8e1a8a683f2aa63971f8e126c2ef75e08f56d16e1ec492cf9d26e730eae4d1a3fecbbb5db81e74d5195f49f1
```

一看特征就知道考察rsa，出于简便，我们把pinvq记为x，qinvp记为y。其中pinvq是q关于q的逆元，qinvp是q关于p的逆元，由此得出以下同余式：

$$\{$$

将其改写成如下形式：

$$\{$$

将上面两式做差得：$p(k + x) = q(k +$

由于p和q为素数，所以我们得到：

$$\{$$

将p和q代回上面 的$k * q + 1 =$ 并化简得到

$$k\ k + 1 = xy$$

所以$k =$

又由于$\phi(n) = (p-1)(q-1)$

将k2代入得$\phi(n) = (k + y$

为避免出现除法，我们写成如下形式：

$$k * \phi(n) = (k + y - 1)($$

由于$e * d =$

$$k = \quad e*d-1$$

注意到$d < \phi(n)$，所以k是小于e的，而e给的并不大，所以我们可以枚举k

通过$\phi(n) =$ 求出$\phi(n)$，这样上面的方程只有一个未知数k了，利用python的z3库即可解方程

z3基本使用如下：

> python z3库 - Hello World
> Z3Py教程(翻译)_40KO的博客-CSDN博客_python z3

解题脚本：

```
from Crypto.Util.number import *
from gmpy2 import *
from z3 import *

pinvq = 0x63367a2b947c21d5051144d2d40572e366e19e3539a3074a433a92161465543157854669134c03642a12d304d2d9036e6458fe4c850c772c19c4eb3f567902b3
qinvp = 0x79388eb6c541fffefc9cfb083f3662655651502d81ccc00ecde17a75f316bc97a8d888286f21b1235bde1f35efe13f8b3edb739c8f28e6e6043cb29569aa0e7b
c = 0x5a1e001edd22964dd501eac6071091027db7665e5355426e1fa0c6360accbc013c7a36da88797de1960a6e9f1cf9ad9b8fd837b76fea7e11eac30a898c7a8b6d8c8989db07c2d80b14487a167c0064442e1fb9fd657a519cac5651457d64223baa30d8b7689d22f5f3795659ba50fb808b1863b344d8a8753b60bb4188b5e386
e = 0x10005
d = 0xae285803302de933cfc181bd4b9ab2ae09d1991509cb165aa1650bef78a8b23548bb17175f10cddffcde1a1cf36417cc080a622a1f8c64deb6d16667851942375670c50c5a32796545784f0bbcfdf2c0629a3d4f8e1a8a683f2aa63971f8e126c2ef75e08f56d16e1ec492cf9d26e730eae4d1a3fecbbb5db81e74d5195f49f1


for k in range(1,e):
    phi=(e*d-1)//k
    if (e*d-1)%k!=0:
        continue
    if e*d%phi!=1:
        continue
    x=Int('x')
    s=Solver()
    s.add(x*phi==(x+qinvp-1)*(pinvq*qinvp-1+x*(pinvq-1)))
    if s.check()==sat:
        print(s.model())
        k1=int(str(s.model()[x]))
        k2=(pinvq*qinvp-1)//k1
        p=k1+qinvp
        q=k2+pinvq
        print(long_to_bytes(pow(c,d,p*q)))
#flag{c4617a206ba83d7f824dc44e5e67196a}
```