长安“战疫”网络安全卫士守护赛_crypto_复现

原创

M3ng@L 于 2022-01-13 21:02:39 发布 3355 收藏

分类专栏： CTF比赛复现 文章标签： 密码学 python

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/qq_51999772/article/details/122482632

版权

CTF比赛复现 专栏收录该内容

31 篇文章 0 订阅

订阅专栏

长安“战疫”网络安全卫士守护赛_Crypto

math

涉及的知识点：`RSA加密未知模数`， `已知p对q的逆元以及q对p的逆元求RSA的模数N`

题目描述

题目没有描述，只有已知量`c,e,d,pinvq,qinvp`

公式推导

已知`pinvq,qinvp`求`n`，这样我们才能求得$m=c^d \pmod n$

所谓`pinvq`也就是$p*pinvq=1(mod\ q)$

`qinvp`就是$q*qinvp=1(mod\ p)$

为了之后方便书写，我们设$x=pinvq$，$y=qinvp$

那么同余式转换为

$p*$

等式两边同时加上$k*q$或者$k*$

得到

$p*q+1$

$q*p+1$

那么等式右边都是一样的，连在一起得到

$p*(x+1)$

那么由于p,q都是素数，也就是说彼此是互素的

一定有p是$y+k$的因数，是$x-1$的因数

看一看$y+k$和$x-1$的范围

由于$x*$也就是在横的世界里一定小于

所以$0<x<$

由于$q*$

同理y是小于p的，但是要让等式成立的话，那么要使$0<k<$

综上$0<x+k<$

同理$0<y+k<$

这个范围有什么意义呢

联系之前的推出的结果p整除$y+k$的，是整除$x-1$的

所以p,q只有一种可能$p=$

到了这里我们已经用其他的量来表示p,q了，现在需要知道的是k ,k 的值

在$\phi(n)$的表达式中看看（为什么要用$\phi(n)$呢，因为由于e,d已知，我们实际上可以爆破出$\phi(n)$的大小）

$\phi(n)=(p-1)*($

$=((x-1)$

$=(x-1)$

中心思想是要求k或者k的值，也就是说先求出来一个，另外一个自然也就知道了，所以我们需要用k来表示k

由已知得

$p*$

将p,q代换成x,y,k ,k

得到$(y+k)*$

移项得到$x*$

那么通过这个等式我们就可以用k来表示k了

也就是$k=$

接着刚才的$\phi(n)$展开式

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

移项

$0=k*(x-1)$

等式两边同时乘以k

$0=(x-$

未知量是k，那么这就是一个关于k 的一元二次方程，系数均已知（$\phi(n)$可以爆破出来）

关于$\phi(n)$的爆破可以这样进行

```
for k in range(3,e):
    if temp % k == 0:
        phi_n = temp // k
```

使用一元二次方程的求根公式可以得到k

那么$k=$

所以求得$p=$

那么$n=$

到这里推导就结束了，然后就是正常的RSA解密了

代码实现

```python
from Crypto.Util.number import *
import gmpy2
from tqdm import tqdm

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

x = pinvq
y = qinvp
temp = e * d - 1
for k in tqdm(range(3,e)):
    if temp % k == 0:
        phi_n = temp // k
        # print(k)
        a = x - 1
        b = (x - 1) * (y - 1) + x * y - 1 - phi_n
        c = (x * y - 1) * (y - 1)
        Delta = b ** 2 - 4 * a * c
        if gmpy2.is_square(Delta):
            result1 = (-b + gmpy2.isqrt(Delta)) // (2 * a)
            result2 = (-b - gmpy2.isqrt(Delta)) // (2 * a)
            temp1,temp2 = result1,result2
            if (x * y - 1) % temp1 == 0:
                k1,k2 = temp1,(x * y - 1) // temp1
            elif (x * y - 1) % temp2 == 0:
                k1,k2 = temp2,(x * y - 1) // temp2
            q = x + k2
            p = y + k1
            n = p * q
            m = pow(cipher,d,n)
            flag = bytes.decode(long_to_bytes(m))
            print(flag)
            break
        else:
            continue
```

参考文章

write-up/README.md at master · pcw109550/write-up (github.com)

LinearEquations

知识点：`LCG变种`

题目描述

```
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
```

程序分析

把flag分成了三段，也就是对应着三个密文，`self.a，self.b，self.c`

加密过程是线性同余生成器的变种，线性同余生成器_百度百科 (baidu.com)

总的来说就是不断继续递归

看一下`next()`函数的加密过程有两个系数，`self.state[-1],self.state[-2]`

```
new = (self.a * self.state[-1] + self.b * self.state[-2] + self.c) % self.n
```

这两个系数最初是`seed1，seed2`充当的，是未知量，也无法推出，也没有必要退出

因为ta加密了5轮，每轮结束之后，得到的结果`new`加入`self.state`，这里的new就是data

而我们已知`data`，也就是每组加密之后的结果，所以在加密两轮之后，

```
new = (self.a * self.state[-1] + self.b * self.state[-2] + self.c) % self.n
```

这个等式里的所有量除了self.a，self.b，self.c需要求，其他的量都已知

总共加密了5轮，那么除去前面两轮加密过程中我们有一个或者两个系数未知，剩下3轮所有系数和等式左边的结果都已知，也就是三个未知量，三个方程，可以解。那么剩下的就是解方程组

注意这个方程组是在`模self.n`的世界里面进行的，所以在进行求方程组之前我们需要对数据进行模初始化（也可以直接在涉及除法的时候写在`inverse_mod()`且在结果之后`模self.n`），self.n是已知的

$data3 = data2 *$

$data4 = data3 *$

$data5 = data4 *$

联立方程求`a,b,c`

代码实现

```
sage: data = [2626199569775466793, 8922951687182166500, 454458498974504742, 7289424376539417914, 8673638837300855396]
....: n = 10104483468358610819
....: d1 = mod(data[0],n)
....: d2 = mod(data[1],n)
....: d3 = mod(data[2],n)
....: d4 = mod(data[3],n)
....: d5 = mod(data[4],n)
#这里注释的公式运行出来是错误的，似乎因为在模运算中不能有两个及以上的"/"，保险一点建议大家用inverse_mod()来表达除法
....: # b = ((d4 - d3)*(d4 - d2) - (d5 - d3)*(d3 - d2))/((d3 - d2)*(d4 - d2)) / (((d2 - d1)*(d5 - d3) - (d3 - d1)*(d3 - d2))/((d3 -d2)*(d5 - d3)))
....: # print(b)
sage: b = ((d4 - d3)^2 - (d5 - d4)*(d3 - d2)) / ((d2 - d1)*(d4 - d3) - (d3 - d2)^2)
....: print(b)
....: a = ((d4 - d3)*(d3 - d2) - b*(d2 - d1)*(d3 - d2)) / (d3 - d2)^2
....: print(a)
....: c = d3 - d2*a - d1*b
....: print(c)
8175498372211240502
5490290802446982981
6859390560180138873
```

得到的a,b,c进行long_to_bytes()即可

其他

其他的密码题难度不大

可以参考这篇文章(11条消息) 2022 长安“战疫”网络安全卫士守护赛 WriteUp_是Mumuzi的博客-CSDN博客