

week1-V

Crypto

InfantRSA

题目介绍:

真*签到题

$p = 681782737450022065655472455411$;

$q = 675274897132088253519831953441$;

$e = 13$;

$c = \text{pow}(m, e, p * q) = 275698465082361070145173688411496311542172902608559859019841$

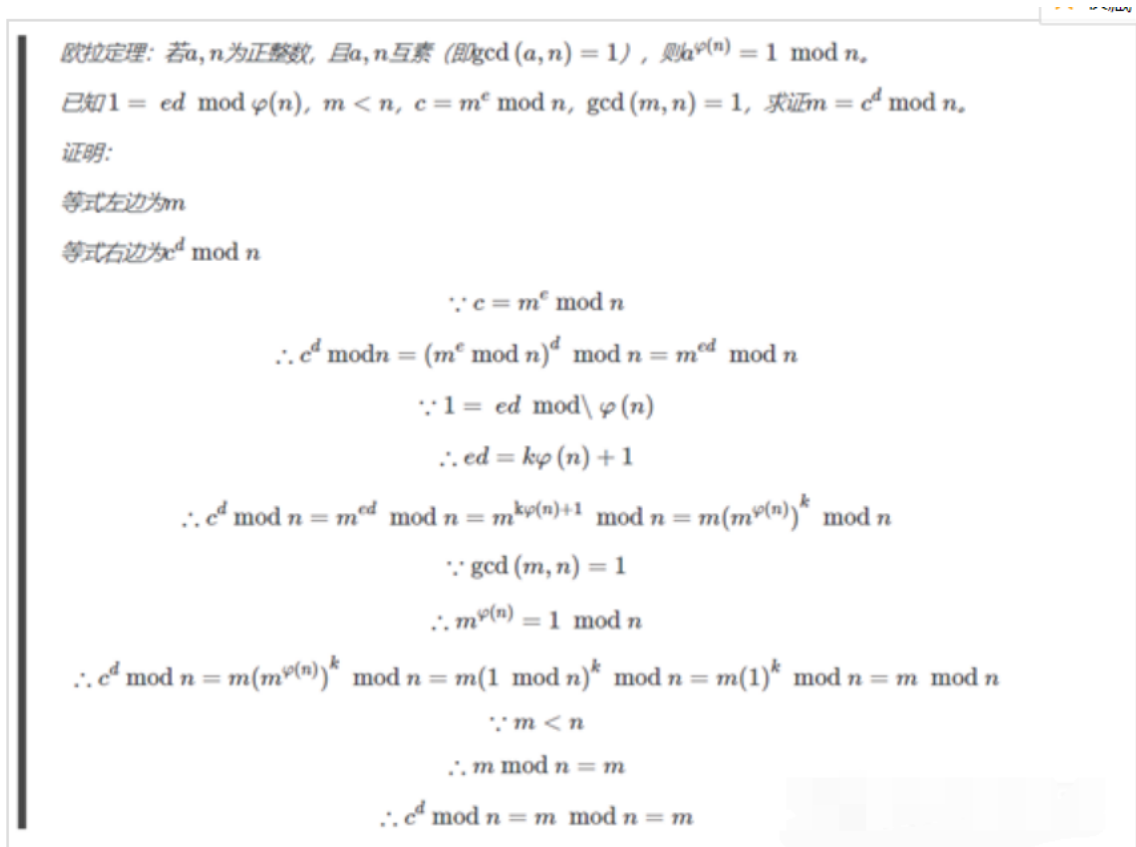
解题思路:

知道所有的条件了，我们只需要算出私钥d就可以解密了，私钥d是公钥在模 $\varphi(n)$ ，即是这里的 $\varphi(p * q) = (p - 1) * (q - 1)$

直接用gmpy2模块下的invert可以求出私钥来， $d = \text{gmpy2.invert}(e, (p - 1) * (q - 1))$

然后m（即flag）为： $m = \text{pow}(c, d, p * q)$

这里解密方程的证明：



欧拉定理: 若 a, n 为正整数, 且 a, n 互素 (即 $\gcd(a, n) = 1$), 则 $a^{\varphi(n)} = 1 \pmod n$.

已知 $1 = ed \pmod{\varphi(n)}$, $m < n$, $c = m^e \pmod n$, $\gcd(m, n) = 1$, 求证 $m = c^d \pmod n$.

证明:

等式左边为 m

等式右边为 $c^d \pmod n$

$$\begin{aligned} \because c &= m^e \pmod n \\ \therefore c^d \pmod n &= (m^e \pmod n)^d \pmod n = m^{ed} \pmod n \\ \because 1 &= ed \pmod{\varphi(n)} \\ \therefore ed &= k\varphi(n) + 1 \\ \therefore c^d \pmod n &= m^{ed} \pmod n = m^{k\varphi(n)+1} \pmod n = m(m^{\varphi(n)})^k \pmod n \\ \because \gcd(m, n) &= 1 \\ \therefore m^{\varphi(n)} &= 1 \pmod n \\ \therefore c^d \pmod n &= m(m^{\varphi(n)})^k \pmod n = m(1 \pmod n)^k \pmod n = m(1)^k \pmod n = m \pmod n \\ \because m &< n \\ \therefore m \pmod n &= m \\ \therefore c^d \pmod n &= m \pmod n = m \end{aligned}$$

exp

```
import gmpy2
p = 681782737450022065655472455411;
q = 675274897132088253519831953441;
e = 13;
c = 275698465082361070145173688411496311542172902608559859019841
d = gmpy2.invert(e, (p-1)*(q-1))
m = pow(c, int(d), p*q)
print hex(m)[2:-1].decode('hex')
```

flag:hgame{t3Xt6O0k_R5A!!!}

Affine

题目介绍:

Some basic modular arithmetic...

解题思路:

加密代码:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import gmpy2
from secret import A, B, flag
assert flag.startswith('hgame{') and flag.endswith('}')

TABLE = 'zxcvbnmasdfghjklqwertyuiop1234567890QWERTYUIOPASDFGHJKLZXCVBNM'
MOD = len(TABLE)

cipher = ''
for b in flag:
    i = TABLE.find(b)
    if i == -1:
        cipher += b
    else:
        ii = (A*i + B) % MOD
        cipher += TABLE[ii]

print(cipher)
# A8I5z{xr1A_J7ha_vG_TpH410}
```

是拓展了字符表的仿射加密

我们可以知道放射加密的公式是: $c \equiv Ax + B \pmod{m}$ 其中, x 为明文, c 为密文, m 为模数, 这里 $m = \text{len}(\text{TABLE})$

所以对应的解密公式即为: $x \equiv (c - B) * A^{-1} \pmod{m}$, 这里, A^{-1} 是 A 在模 m 下的逆元, (如rsa中 d 是 e 在模 $\phi(n)$ 下的逆元一般),

题目中, 我不知道 A, B , 所以我选择爆破 A, B 的方法, 然后在所有答案中找包含字符 'hgame' 的结果。
(因为知道 flag 中前五个字符为 hgame, 所以其实也可以解方程, 但是因为 TABLE 比较小, 为了抢个一血, 就直接爆了, 甚至脚本也是在原基础上直接改的)

exp

```

TABLE = 'zxcvbnmasdfghjklqwertyuiop1234567890QWERTYUIOPASDFGHJKLZXCVBNM'
MOD = 62
for A in range(MOD):

    for B in range(MOD):
        flag=""
        cipher = 'A8I5z{xr1A_J7ha_vG_TpH410}'
        for b in cipher:
            i = TABLE.find(b)
            if i == -1:
                flag += b
            else:
                try:
                    ii = (i-B)*invert(A,MOD)%MOD
                    flag += TABLE[ii]
                except:
                    break
        if 'hgame' in flag:
            print flag
            exit(0)
        # A8I5z{xr1A_J7ha_vG_TpH410}

```

flag: hgame{M4th_u5Ed_iN_cRYpt0}

not_One_time

题目介绍:

In cryptography, the one-time pad (OTP) is an encryption technique that cannot be cracked, but...

Just XOR ;P

nc 47.98.192.231 25001

hint: reduced key space

解题思路:

一次性密码本本身无解，程序漏洞在于key的空间被缩减了，可以多次连接获取密文文本，此时一直不变的flag反而可以看作是用来加密的key了。然后爆破flag，使得flag的每一位能解密 **密文文本** 的**每一项**的对应位，解密成功的判断条件是恢复出的明文在给定的key的空间中。

举个栗子:

flag的第一个字符肯定是h吧,

比如我们连接20次，获得20组密文，

h和每一组的密文的第一个字符异或，肯定也是一个可见字符，并且20组异或下来，获得的可见字符肯定也是20个，不会少一个。

所以我们只需要遍历所有的密钥空间内的元素，找到满足条件（即，和20组密文的对应位异或下来能生成20个可见字符）的字符，组合起来即为flag

exp

```

from pwn import *
import string, binascii, base64

def getex():
    example=[]

```



```

[root@iZbp15axph2ymoey4m1kisZ ~]# nc 47.98.192.231 25002
> ~!@#$$%^&*()_+`1234567890-=
~2)^`_%!@$*+&(#13 9 8457- 0=6
> abcdefghijklmnopqrsatxyz
apkgnlfbceimhjqdoq y xrst z a
> qwertyuiopasdfghjklzxcvbnm
qhaufsywetodiprgj v cklxn bmz
> qwertyuiopasdfghjklzxcvbnmQWERTY
qhaufsywetodiprgjYQvRWcklxEbmzT
> 1
1
> 2
2
> 3
3
> 4
4
> 5
5
> 6
6
Rua!!!
hL+jm5{gae$IUtmp3}iu!0m_PRAnTTe!

```

那么我们只需要输入32位字符，观察他置换后结果，得到映射关系，然后我们根据这个映射关系，将最后给出的置换过后的flag置换回来即可。(注意，为了让我们的映射关系为一一映射，即映射关系为双射关系，我们的输入应该是不重复的字符)

exp

```

cin='qhaufsywetodiprgjYQvRWcklxEbmzT'
cout='qwertyuiopasdfghjklzxcvbnmQWERTY'
flagout='hL+jm5{gae$IUtmp3}iu!0m_PRAnTTe!'
index=[]
for i in cout :
    index.append(cin.find(i))
flag=''
for i in range(32):
    flag+=flagout[index[i]]
print flag

```

flag:hgame{jU\$t+5ImpL3_PeRmuTATi0n!!}