Secure Programming 2020

HW2 write-up

RSA

腳本solve.py附在zip /RSA/solve.py

這一題給了n跟c, n由三個數p,q1,q2相乘而得,這三個數有一定的關係, q1 = next_prime(2*p), q2 = next_prime(3*q1),也就是說我們假設一個p就可以推出對應的q1跟q2, 也就可以直接驗證p*q1*q2是否等於n,搜索的速度應該會比正常的RSA快非常多,為了加快搜尋的速度,可以使用binary search.

```
...
p = getPrime(512)
q1 = next_prime(2 * p)
q2 = next_prime(3 * q1)

n = p * q1 * q2
...
```

根據題目,p是一個512bit的數字,所以p的值應該會落在100...000(512bit)跟111...111(512bit)之間,在這個區間作binary search,p等於中間值,如果不是質數,就找下一個大於p的質數,然後算出對應的q1,q2,再比對p*q1*q2是否等於n,如果等於n,就找到n的質因數分解,就可以算d,然後算出原文了.

```
...
L = 1000...000(512bit)
R = 1111...111(512bit)
...
p = (L + R) // 2
#if p is not prime, find next prime
if sp.isprime(p) is False:
  p = next_prime(p)
q1 = next_prime(2*p)
q2 = next_prime(3*q1)

temp = p * q1 * q2
if n == temp:
  #find factor p of n
  ...
elif n > temp:
  #p is too small, need to be larger
  L = p
else:
  #p is too large, need to be smaller
  R = p
```

```
FLAG{Ew9xeANumjDr6bXemHsh}
```

LSB

腳本solve2.py附在zip /LSB/solve2.py

上課提到的LSB Oracle Attack,跟通常的mod2不太一樣,這次變成了mod3,我是用解法二,由於server會回給你mod3的餘數,你可以由餘數推出最後一個位數,在mod2的情境下,餘數就是最後一個bit,mod3的話,就是在base3下的最後一個位數,

```
...
  ------------  ----
m -   y1     -  -x0-
  ------------  ----
m = x0 + 3y1
r = ⌊x0 + 3y1⌋n mod3
  = ⌊x0⌋n mod3
x0(最後一位數在base3下) = r mod3
...
```

倒數第二個位數可以從最後一個位數跟server回的餘數推得,以此類推就能得到原來的明文.

```
...
  ------------  ---- ----
m -   y1     -  -x1- -x0-
  ------------  ---- ----
3-1m = 3-1x0 + x1 + 3y2
r = ⌊3-1x0 + x1⌋n mod3
x1(倒數第二位數在base3下) = r  - ⌊3-1x0⌋n (mod3)
...
```

我是直接用講者的模板,將mod2跟2-1的部份都改成3,還有搜索的總次數會是log(3,n)次,就可以得到flag.

```
FLAG{nF9Px2LtlNh5fJiq3QtG}
```