CTF-crypto(RSA)

原创

[hangshao0.0](#) 于 2020-03-30 22:32:13 发布 545 收藏 5

分类专栏： ctf-crypto

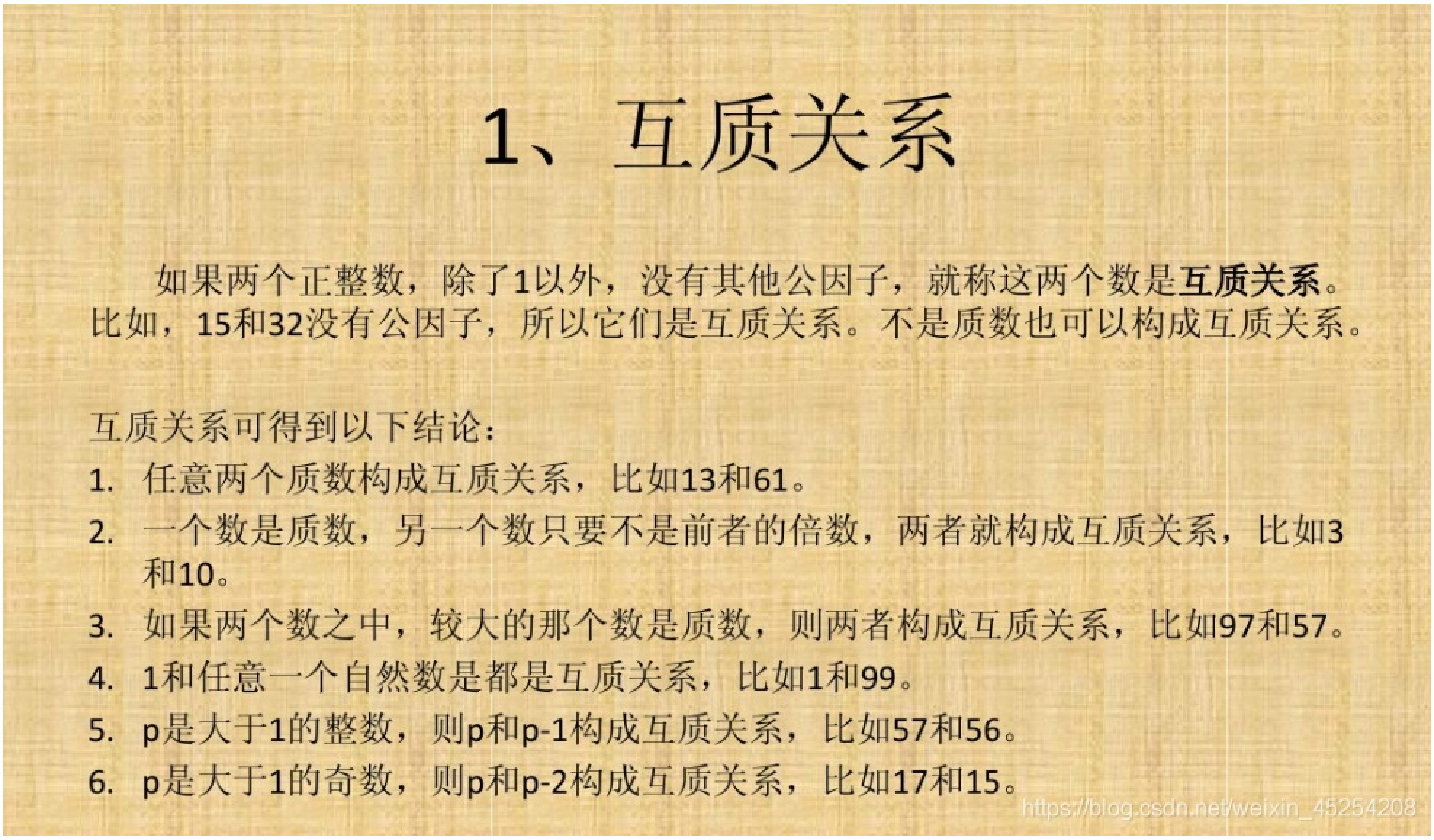
本文链接：https://blog.csdn.net/weixin_45254208/article/details/105209025

首先要向教我们信安数学基础的老师道个歉，我记得当初碰到难题的时候抱怨过："真搞不懂学这个东西有什么用？"

------现在看来，当时还是太年轻了哈哈哈。

RSA基础

有基础后还是好啊，基本定理随便扫一眼就行



欧拉函数：

表示少于或等于n的正整数中与n互质的数的数目，又称为φ函数、欧拉商数等。 欧拉函数，以φ(n)表示。在1到8之中，与8形成互质关系的是1、3、5、7，所以 φ(8) = 4。

φ函数的值：

通式：φ(x)=x(1-1/p1)(1-1/p2)(1-1/p3)(1-1/p4).....(1-1/pn),其中p1, p2......pn为x的所有质因数，x是不为0的整数。φ(1)=1（唯一和1互质的数(小于等于1)就是1本身）。(注意：每种质因数只一个。比如12=2*2*3那么φ（12）=12*（1-1/2）*(1-1/3)=4。

基本性质：

1. φ(1)=1, 小于等于1的正整数中唯一和1互质的数就是1本身。
2. 如果n是质数，则 φ(n)=n-1 。因为质数与小于它的每一个数，都构成互质关系。比如5与1、2、3、4都构成互质关系。
3. 如果正整数是质数的次幂, 那么φ(n)=φ(p^k)=p^k – p^(k-1) = (p-1)p^(k-1)。
4. 欧拉函数是积性函数, 即对于两个互质的正整数m和n, φ(mn)=φ(m) φ(n) 。

3、同余

同余：

当两个整数除以同一个正整数，若得相同余数，则二整数同余。

同余符号：

两个整数a，b，若它们除以正整数m所得的余数相等，则称a，b对于模m同余记作 a ≡ b (mod m)，读作a同余于b模m，或读作a与b关于模m同余。比如26 = 14 (mod 12)。

4、欧拉定理

欧拉定理：

如果两个正整数a和n互质，则n的欧拉函数 φ(n) 可以让下面的等式成立：

$$a^{\phi(n)} \equiv 1 \pmod n$$

也就是说，a的φ(n)次方被n除的余数为1。或者说，a的φ(n)次方减去1，可以被n整除。比如，

3和7互质，而7的欧拉函数φ(7)等于6，所以3的6次方（729）减去1，可以被7整（728/7=104）。

$$3^{\phi(7)}=3^6=729 \equiv 728+1 \equiv 104*7+1 \equiv 1 \pmod 7$$

欧拉定理可以大大简化某些运算。比如，7和10互质，根据欧拉定理，已知 φ(10) 等于4，所以马上得到7的4倍数次方的个位数肯定是1。

欧拉定理是RSA算法的核心。

5、模反元素

模反元素

如果两个正整数a和n互质，那么一定可以找到整数b，使得 ab-1 被n整除，或者说ab被n除的余数是1。这时，b就叫做a的模反元素。

$$ab \equiv 1 \pmod n$$

比如，3和11互质，那么3的模反元素就是4，因为 (3 × 4)-1 可以被11整除。显然，模反元素不止一个，4加减11的整数倍都是3的模反元素 {...,-18,-7,4,15,26,...}，即如果b是a的模反元素，则 b+kn 都是a的模反元素。

欧拉定理可以用来证明模反元素必然存在。a的 φ(n) -1 次方，就是a的模反元素。

来举个例子

第一步，随机选择两个不相等的质数p和q。

爱丽丝选择了61和53。（实际应用中，这两个质数越大，就越难破解。）

第二步，计算p和q的乘积n。

爱丽丝就把61和53相乘。

n = 61×53 = 3233

n的长度就是密钥长度。3233写成二进制是110010100001，一共有12位，所以这个密钥就是12位。实际应用中，RSA密钥一般是1024位，重要场合则为2048位。

第三步，计算n的欧拉函数φ(n)。

根据公式：φ(n) = (p-1)(q-1)

爱丽丝算出φ(3233)等于60×52，即3120。

第四步，随机选择一个整数e，条件是1< e < φ(n)，且e与φ(n) 互质。

爱丽丝就在1到3120之间，随机选择了17。（实际应用中，常常选择65537。）

第五步，计算e对于φ(n)的模反元素d。

所谓模反元素就是指有一个整数d，可以使得ed被φ(n)除的余数为1。

$$ed \equiv 1 \pmod{\phi(n)}$$

这个式子等价于

$$ed - 1 = k\phi(n)$$

于是，找到模反元素d，实质上就是对下面这个二元一次方程求解。

$$ex + \phi(n)y = 1$$

已知 e=17, φ(n)=3120，

$$17x + 3120y = 1$$

这个方程可以用"扩展欧几里得算法"求解，此处省略具体过程。总之，爱丽丝算出一组整数解为 (x,y)=(2753,-15)，即 d=2753。

至此所有计算完成。

第六步，将n和e封装成公钥，n和d封装成私钥。

在爱丽丝的例子中，n=3233，e=17，d=2753，所以公钥就是 (3233,17)，私钥就是（3233, 2753）。

实际应用中，公钥和私钥的数据都采用ASN.1格式表达。

回顾上面的密钥生成步骤，一共出现六个数字：

p
q
n
φ(n)
e
d

这六个数字之中，公钥用到了两个（n和e），其余四个数字都是不公开的。其中最关键的是d，因为n和d组成了私钥，一旦d泄漏，就等于私钥泄漏。

那么，有无可能在已知n和e的情况下，推导出d？

（1）ed≡1 (mod φ(n))。只有知道e和φ(n)，才能算出d。

（2）φ(n)=(p-1)(q-1)。只有知道p和q，才能算出φ(n)。

（3）n=pq。只有将n因数分解，才能算出p和q。

结论：如果n可以被因数分解，d就可以算出，也就意味着私钥被破解。

用python脚本解CTF中有关RSA的题目

上题

```
n=73069886771625642807435783661014062604264768481735145873508846925735521695159
c=28767758880940662779934612526152562406674613203406706867456395986985664083182
e=65537
```

思路

```
n=pq
phi =(p-1)(q-1)
ed=1 mod phi
公钥：(n,e)
私钥：(n,d)
```

用到的两个模块

```
import libnum
'''
libnum.n2s(n)     数字转字符串(10进制和16进制)
libnum.s2n(s)     字符串转数字(10进制和16进制)
libnum.b2s(s)     二进制数字转字符串
libnum.s2b(s)     字符串转二进制数字数字
libnum.generate_prime(n) 随机生成n位二进制内的质数
libnum.factorize(n)    因数分解，数字太大的话就很慢，下面附上在线分解大素数的地址
'''
```

在线分解大素数：
http://www.factordb.com/index.php?query=

```
import gmpy2
'''
gmpy2.invert(e,phi)    求模反元素d
pow(c,d,n)      求c^d mod n，也就是RSA解密结果
gmpy2.iroot(x,n)     x开n次根
gmpy2.gcdext(a,b)    扩展欧几里得算法
gmpy2.gcd(a,b)     欧几里得算法，最大公约数
'''
```

解题代码

```
import libnum
import gmpy2
# 题目信息
c=28767758880940662779934612526152562406674613203406706867456395986985664083182
n=73069886771625642807435783661014062604264768481735145873508846925735521695159
e=65537
# 在线分解出p和q
p = 386123125371923651191219869811293586459
q = 189239861511125143212536989589123569301

phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(libnum.n2s(m))
```