RSA入门总结

原创

薛定谔的呱 于 2019-02-23 20:42:07 发布 184 收藏
分类专栏： ctf 文章标签： ctf crypto

本文链接：https://blog.csdn.net/qq_36360189/article/details/87896865

---

title: RSA入门总结
date: 2017-11-13 03:14:43
tags: CRYPTO

RSA

…先简单说一下rsa相关的东西。

RSA 算法的可靠性由其极大整数因数分解的难度决定。

也就是说，极大整数做因数分解愈困难，RSA 算法愈可靠。

在还没找到一种快速因数分解的算法之前，rsa加密还是很可靠的。目前短的 RSA 秘钥可能被强力方式破解。而只要密钥长度足够长，用RSA加密的信息实际上是不能被解破的。

最近稍微看了一下相关的东西。然后整理了一下。

a^b是a的b次方…忘了应该怎么打出来了orz…

```
RSA算法涉及到的参数：

p : 大质数p
q : 大质数q
n : n=p*q

欧拉函数：
ϕ(n) = ϕ(p) * ϕ(q) = (p - 1)(q - 1) = n - (p + q - 1)

e : encryption key (public key) (又称加密指数)

d : decryption key (private key) e对于ϕ(n)的模反元素即e模ϕ(n)的逆元

m：message（m < n），明文。

c : ciphertext，c≡m^e(mod n),即密文
```

加密过程：

1.随机选择两个不相等的质数p和q。

2.计算p和q的乘积n。

n的长度就是密钥长度。

如n为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。）

3.计算n的欧拉函数φ(n)

φ(n) = (p-1)(q-1)

4.随机选择一个整数e，条件是**1< e < φ(n)**，且e与φ(n) **互质**，将e作为公钥。

5.计算e关于φ(n)的模反元素d，作为私钥。

计算过程：

模反元素即模逆元（参考离散数学或者信安数学基础或者百科学习）

简单来说，就是一个整数d可以使ed乘积被φ(n)除的余数为1：
ed ≡ 1 (mod φ(n))（也就是ed%φ(n)＝1）
即求解

ed＋φ(n)x＝1（x未知，但是求得e和φ(n)就可以通过扩展欧几里得算法得到d的值，具体可以参考最底下ctf中rsa题目解析第二个链接。）

公开e，保留d。

**公钥为(n,e)，私钥为(n,d) **

针对明文M，进行加密：C=(M^e)%n，得到的C即为密文

针对密文C，进行解密，M=(C^d)%n，得到的M即为明文

而在ctf比赛里，一般会给出n值和e值，然后可以在**http://factordb.com/**或者大质数分解工具里因式分解，得到p，q。然后通过p，q算出φ(n）。通过φ(n）和e，算出d。

jarvis oj一道简单的rsa：veryeasyrsa

```
已知RSA公钥生成参数：

p = 348758394758943758923795872389234625477 7
q = 876786784356893476598347658437657838 9
e = 65537

求d
```

py脚本如下。
modinv函数就是求模逆的函数。

```
import math



def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not    exist')
    else:
        return x % m

p = 348758394758943758923795872389234625477 7
q = 876786784356893476598347658437657838 9
e = 65537
d = modinv(e, (p - 1) * (q - 1))
print d
```

明文就是m＝pow(c,d,n)

（ps.当e＝1时，c≡ m mod n，因为n远大于c，m＝c.也就是明文密文是一样的.实际上没加密.）

有的时候，会出现多个n的公约数分解.

比如给了2个n（n1，n2），q的值相同（而p不同所以有多个n），e的值和密文c。py脚本如下：

```
def gcd(a, b):
    if a < b:
        a, b = b, a

    while b != 0:
        temp = a % b
        a = b
        b = temp

    return a

n1=。。。
n2=。。。

p = gcd(n1,n2)

print p,"\n\n",n1 // p
```

就可以算出**q**的值啦。

也就是最后输出的**n1除以p**。

老方法，知道p，q就知道φ(n）。通过φ(n）和e，算出d。

其他：

一般题目不会直接给参数，可能给比如pem文件（后缀.enc/.pem），可以通过linux下openssl命令提取。linux下命令：

openssl rsa -pubin -text -modulus -in warmup -in 喵.pem

和

openssl rsautl -encrypt -in FLAG -inkey public.pem -pubin -out 喵.enc

在p，q的取值差异过大，或者p，q的取值过于相近的时候，Format算法与Pollard rho算法都可以很快将n分解成功。

（利用开源项目**yafu**可以自动化实现，语句直接：**factor(n)**

也可以通过linux下的openssl命令提取。

RSA入门级别的大概就这样…
以后大概会补充后续？…

学习了之后的感受是…
我，我以后一定好好听数学课！（ry

```
具体算法解析：
   http://www.ruanyifeng.com/blog/2013/07/rsa_algorithm_part_two.html

关于rsa公钥指数：
   http://blog.csdn.net/hherima/article/details/52461759

关于ctf中rsa的题目解析：

   http://bobao.360.cn/learning/detail/3058.html
(算法思路)

   http://blog.csdn.net/qq_18661257/article/details/54563017
```

```
   http://blog.csdn.net/veritas501/article/details/55257957
（常见类型）

yafu使用：

http://www.mamicode.com/info-detail-1999309.html
```