

# Crypto-RSA加密

原创

Qwzf 于 2019-08-06 23:16:10 发布 3237 收藏 14

文章标签: [CTF](#)

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\\_43625917/article/details/98314507](https://blog.csdn.net/qq_43625917/article/details/98314507)

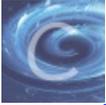
版权



[Crypto](#) 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏



[RSA](#)

1 篇文章 0 订阅

订阅专栏

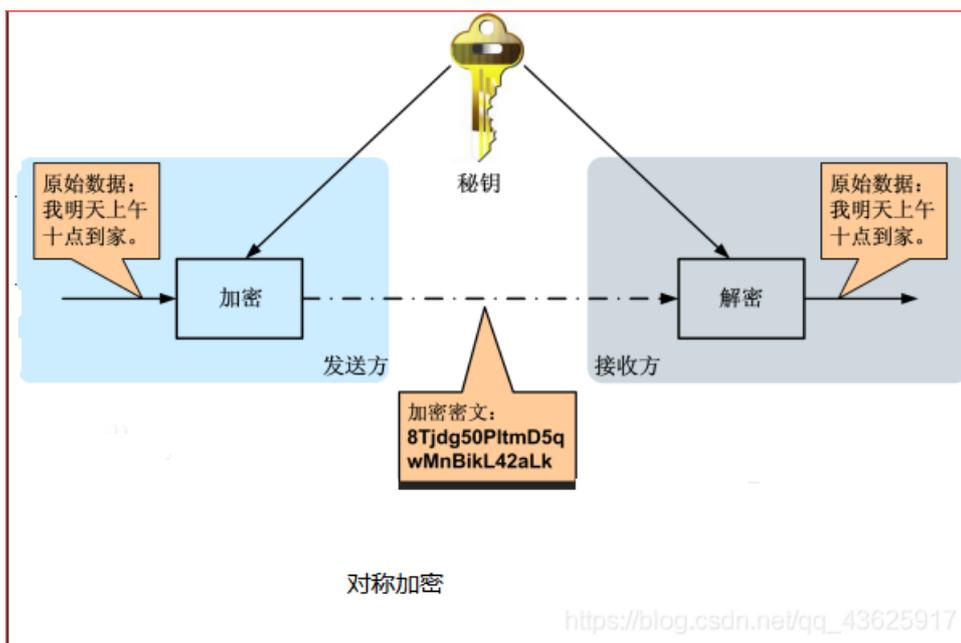
## 前言

最近学习了RSA加密原理, 并且做了些有关RSA的Crypto题。收获很大, 总结了一下

## 一、对称加密和非对称加密

### 对称加密算法

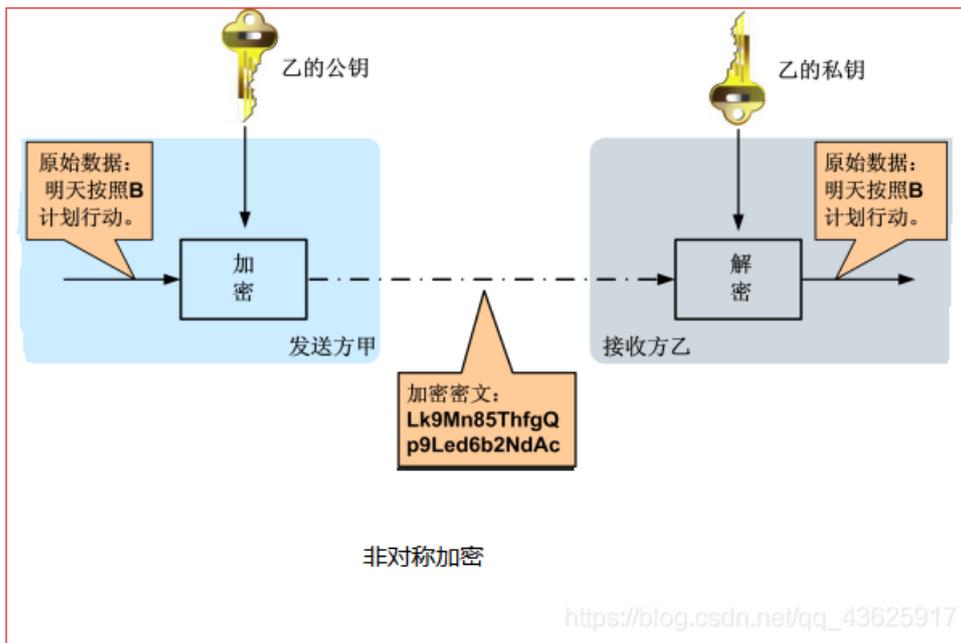
- (1) 甲方选择某一种加密规则, 对信息进行加密;
- (2) 乙方使用同一种规则, 对信息进行解密。



最大弱点: 甲方必须把加密规则告诉乙方, 否则无法解密。保存和传递密钥, 就成了最头疼的问题。

### 非对称加密算法

- (1) 乙方生成两把密钥（公钥和私钥）。公钥是公开的，任何人都可以获得，私钥则是保密的。
- (2) 甲方获取乙方的公钥，然后用它对信息加密。
- (3) 乙方得到加密后的信息，用私钥解密。



公钥加密的信息只有私钥解得开，那么只要私钥不泄漏，通信就是安全的

## 二、RSA基本介绍

### 介绍

- RSA加密算法是一种非对称加密算法。在公开密钥加密和电子商业中RSA被广泛使用。
- 对极大整数做因数分解的难度决定了RSA算法的可靠性。换言之，对一极大整数做因数分解愈困难，RSA算法愈可靠。现在只有短的RSA钥匙才可能被强力方式破解。到目前为止，世界上还没有任何可靠的攻击RSA算法的方式。只要其钥匙的长度足够长，用RSA加密的信息实际上是不能被破解的。

### RSA签名验签

使用私钥将明文进行签名生成密文串与明文一起传输。对方收到数据后使用公钥和密文串进行验签。如果验签通过就说明就说明第一数据没有被修改过；第二这些数据一定是持有私钥的人发送的，因为私钥只有自己持有，起到防抵赖的作用。

## 三、RSA数学知识

### 1、互质关系

如果两个正整数，除了1之外没有其他公因子，我们称这两个数是互质关系。不是质数也可以构成互质关系。

关于互质关系，有以下结论：

- 任意两个质数构成互质关系，比如13和61。
- 一个数是质数，另一个数只要不是前者的倍数，两者就构成互质关系，比如3和10。
- 如果两个数中，较大的那个数是质数，则两者构成互质关系，比如97和57。
- 1和任意一个自然数都是互质关系。
- $p$ 是大于1的整数，则 $p$ 和 $p-1$ 构成互质关系，比如57和56。
- $p$ 是大于1的奇数，则 $p$ 和 $p-2$ 构成互质关系，比如17和15。

## 2、欧拉函数

任意给定正整数 $n$ ,计算在小于等于 $n$ 的正整数之中,有多少个与 $n$ 构成互质关系。计算这个值的方法叫做欧拉函数。以 $\varphi(n)$ 表示。在1到8之中,与8形成互质关系的是1、3、5、7,所以  $\varphi(8) = 4$ 。

如果 $n=1$ ，则  $\varphi(1) = 1$ 。因为1与任何数（包括自身）都构成互质关系。

如果 $n$ 是质数，则  $\varphi(n)=n-1$ 。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。

如果 $n$ 是质数的某一个次方，即  $n = p^k$  ( $p$ 为质数， $k$ 为大于等于1的整数)，则

$$\phi(p^k) = p^k - p^{k-1}$$

比如  $\varphi(8) = \varphi(2^3) = 2^3 - 2^2 = 8 - 4 = 4$ 。

这是因为只有当一个数不包含质数 $p$ ，才可能与 $n$ 互质。而包含质数 $p$ 的数一共有 $p^{k-1}$ 个，即 $1 \times p$ 、 $2 \times p$ 、 $3 \times p$ 、...、 $p^{k-1} \times p$ ，把它们去除，剩下的就是与 $n$ 互质的数。

上面的式子还可以写成下面的形式：

$$\phi(p^k) = p^k - p^{k-1} = p^k \left(1 - \frac{1}{p}\right)$$

可以看出，上面的第二种情况是  $k=1$  时的特例。

如果 $n$ 可以分解成两个互质的整数之积，

$$n = p_1 \times p_2$$

则

$$\varphi(n) = \varphi(p_1 p_2) = \varphi(p_1) \varphi(p_2)$$

即积的欧拉函数等于各个因子的欧拉函数之积。比如， $\varphi(56) = \varphi(8 \times 7) = \varphi(8) \times \varphi(7) = 4 \times 6 = 24$ 。

因为任意一个大于1的正整数，都可以写成一系列质数的积。

$$n = p_1^{k_1} p_2^{k_2} \dots p_r^{k_r}$$

根据第4条的结论，得到

$$\phi(n) = \phi(p_1^{k_1}) \phi(p_2^{k_2}) \dots \phi(p_r^{k_r})$$

再根据第3条的结论，得到

$$\phi(n) = p_1^{k_1} p_2^{k_2} \dots p_r^{k_r} \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_r}\right)$$

也就等于

$$\phi(n) = n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_r}\right)$$

这就是欧拉函数的通用计算公式。比如，1323的欧拉函数，计算过程如下：

$$\varphi(1323) = \varphi(3^2 \times 7^2) = 1323 \left(1 - \frac{1}{3}\right) \left(1 - \frac{1}{7}\right) = 756$$

### 3、欧拉定理

如果两个正整数a和n互质，则n的欧拉函数  $\phi(n)$  可以让下面的等式成立：

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

也就是说，a的 $\phi(n)$ 次方被n除的余数为1。或者说，a的 $\phi(n)$ 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 $\phi(7)$ 等于6，所以3的6次方（729）减去1，可以被7整除（728/7=104）。

如果正整数a与质数p互质，应为 $\phi(p)=p-1$ ，所以欧拉函数可写成：

$$a^{p-1} \equiv 1 \pmod{p}$$

这是著名的费马小定理。它是欧拉定理的特例。欧拉定理是RSA算法的核心。

#### 4、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。

$$ab \equiv 1 \pmod{n}$$

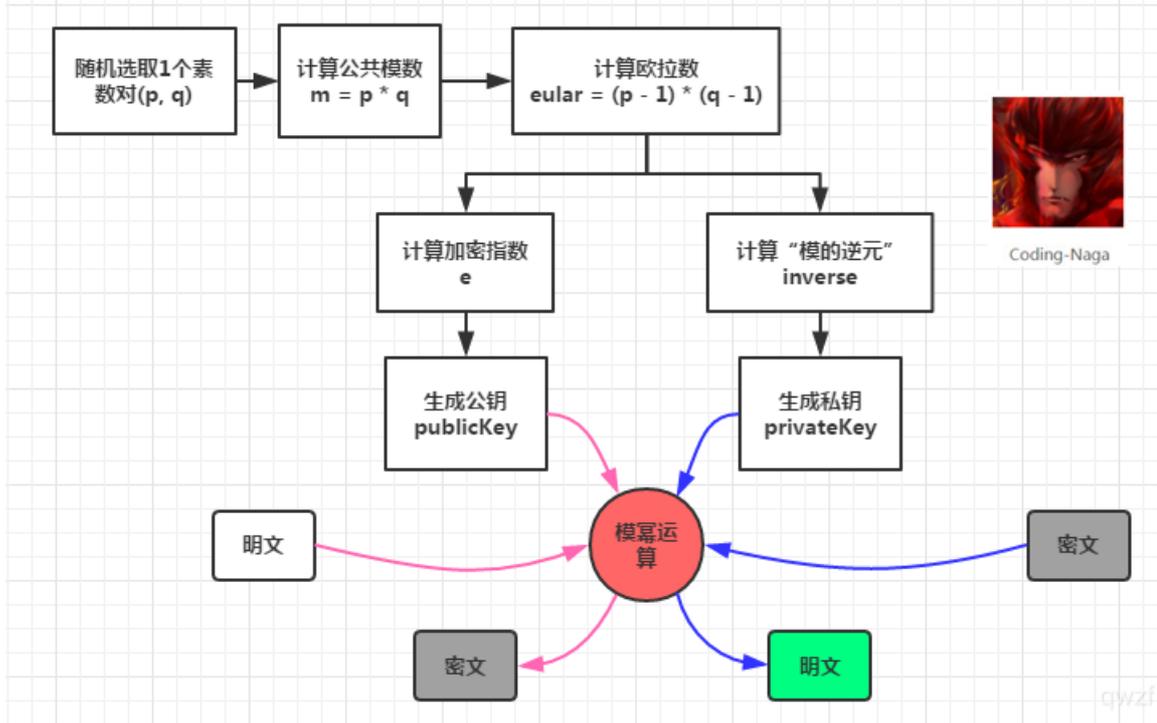
比如：3和11互质，那么3的模反元素是4，应为 $3*4-1$  可以被11整除。4加减11的整数倍数都是3的模反元素。欧拉定理可以用来证明模反元素必然存在：

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

a的 $\phi(n)-1$ 次方，就是a的模反元素

#### 四、RSA算法

## RSA加密算法流程及加密过程图



### 1、公钥和私钥的生成

RSA算法之一种非对称加密算法。具体的加密工程如下：

使用A和他的小伙伴B来举例子。

假设A想通过一个不可靠的媒体接受B的一条私人消息，他可以用下面的方式产生一个公钥和私钥。

1. 随意选择两个大的质数 $p$ 和 $q$ ， $p$ 不等于 $q$ ，计算 $N = pq$ 。
2. 根据欧拉函数，求得 $r = \phi(N) = \phi(p)\phi(q) = (p-1)(q-1)$ 。
3. 选择一个小于 $r$ 的整数 $e$ ，是 $e$ 与 $r$ 互质。并求得 $e$ 关于 $r$ 的模反元素，命名为 $d$ 。（求 $d$ 令 $ed \equiv 1 \pmod{r}$ ）。（模反元素存在，当且仅当 $e$ 与 $r$ 互质）
4. 将 $p$ 和 $q$ 的记录销毁。

其中 $(N, e)$ 是公钥， $(N, d)$ 是私钥。

例子：

1. A随机选两个不相等的质数61和53，并计算两数的积 $N = 61 * 53 = 3233$ ， $N$ 的长度就是密钥长度。3233的二进制是110010100001，一共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，总有的场合是2048位。
2. 计算 $N$ 的欧拉函数。 $\phi(N) = (p-1)(q-1) = 60 * 52 = 3120$ 。
3. A在1到3120上随机选择了一个随机数 $e = 17$ 。
4. 计算 $e$ 对 $\phi(N)$ 的模反元素 $d$ ， $ed \equiv 1 \pmod{\phi(N)}$ 等价于 $ed - 1 = k\phi(N)$ 。  
找到模反元素 $d$ ，实质上就是对这个二元一次方程求解： $17x + 3120y = 1$ 。  
用扩展欧几里得算法求解。可以算出一组解 $(x, y) = (2753, -15)$ ，即 $d = 2753$ 。

其中 $N = 3233, e = 17, d = 2753$ 。所以公钥就是 $(N, e) = (3233, 17)$ ，私钥 $(N, d) = (3233, 2753)$ 。实际应用中公钥和私钥都是采用ASN.1格式表达的。

### 2、可靠性

密钥生成步骤，一共出现六个数字： $p$ 、 $q$ 、 $N$ 、 $\phi(N)$ 、 $e$ 、 $d$

一旦 $d$ 泄露，就等于私钥泄露。

$ed \equiv 1 \pmod{\phi(N)}$ 。只有知道 $e$ 和 $\phi(N)$ ，才能算出 $d$   
 $\phi(N) = (p-1)(q-1)$ 。只有知道 $p$ 和 $q$ ，才能算出 $\phi(N)$   
 $N = pq$ ，只有将 $n$ 分解才能算出 $p$ 和 $q$

只有将 $n$ 质因数分解，才能算出 $d$ 。也就意味着私钥破译。但大整数的质因数分解是非常困难的。

### 3、加密和解密

#### 加密

加密要用到公钥 $(N, e)$ 。

假设 $B$ 要向 $A$ 发送加密信息 $m$ ， $B$ 就要用 $A$ 的公钥 $(N, e)$ 对 $m$ 进行加密。

但 $m$ 必须是整数(字符串可以取ascii值或unicode值)，且 $m$ 必须小于 $n$ 。

加密就是计算下式的 $c$ 。

$$m^e \equiv c \pmod{n}$$

假设 $m=65$ ,  $A$ 的公钥 $(3233, 17)$ , 所以等式如下:

$$65^{17} \equiv 2790 \pmod{3233}$$

所以 $c$ 等于2790， $B$ 就把2790发给 $A$ 。

#### 解密

$A$ 收到 $B$ 发来的 $c$ (也就是2790)后，就用自己的私钥 $(3233, 2753)$ 进行解密。

$$c^d \equiv m \pmod{n}$$

也就是 $c$ 的 $d$ 次方除以 $n$ 的余数就是 $m$ 。

$$2790^{2753} \equiv 65 \pmod{3233}$$

因此得到原文65。

#### 证明

证明为什么用私钥就能解密。就是要证明这个式子:

$$c^d \equiv m \pmod{n}$$

因为加密规则是:

$$m^e \equiv c \pmod{n}$$

于是， $c$ 可以写成:

$$c = m^e - kn$$

将 $c$ 代入我们要证明的那个解密规则:

$$(m^e - kn)^d \equiv m \pmod{n}$$

等同于求证:

$$m^{ed} \equiv m \pmod{n}$$

因为:  $ed \equiv 1 \pmod{\phi(n)}$

所以:  $ed = h\phi(n) + 1$

将 $ed$ 代入:

$$m^{(h\phi(n)+1)} \equiv m \pmod{n}$$

接下来，分成两种情况证明上面这个式子。

### 1. 当m与n互质

根据欧拉定理，此时

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

$$\text{得到: } (m^{\phi(n)})^h \times m \equiv m \pmod{n}$$

由此原始得到证明。

### 2. 当m与n不是互质时

此时，由于n等于质数p和q的乘积，所以m必然等于kp或kq。

以  $m = kp$  为例，考虑到这时k与q必然互质，则根据欧拉定理，下面的式子成立：

$$(kp)^{q-1} \equiv 1 \pmod{q}$$

进一步得到：

$$[(kp)^{q-1}]^{h(p-1)} \times kp \equiv kp \pmod{q}$$

$$\text{即: } (kp)^{ed} \equiv kp \pmod{q}$$

$$\text{改写成: } (kp)^{ed} = tq + kp$$

上式t必然能被p整除，即  $t = t'p$

$$(kp)^{ed} = t'pq + kp$$

因为  $m = kp, n = pq$ ，所以：

$$m^{ed} \equiv m \pmod{n}$$

证明完毕。

参考博客：

[RSA算法原理\(一\)](#)

[RSA算法原理\(二\)](#)

## 五、RSA实战

### Crypto1: easy\_RSA

#### easy RSA

查看Writeup 题目建议

难度系数: ★ 1.0

题目来源: poxlove3

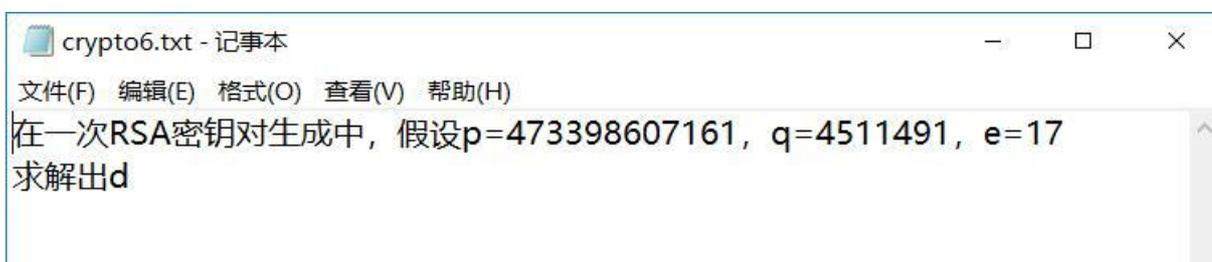
**题目描述:** 解答出来了上一个题目的你现在可是春风得意，你们走向了下一个题目所处的地方 你一看这个题目傻眼了，这明明是一个数学题啊!!! 可是你的数学并不好。扭头看向小鱼，小鱼哈哈一笑，让你在学校里面不好好听讲现在傻眼了吧~来来来! 三下五除二，小鱼便把这个题目轻轻松松的搞定了

**题目场景:** 暂无

题目附件: 附件0

[https://blog.csdn.net/qq\\_43625917](https://blog.csdn.net/qq_43625917)

题目提示是RSA，下载题目查看



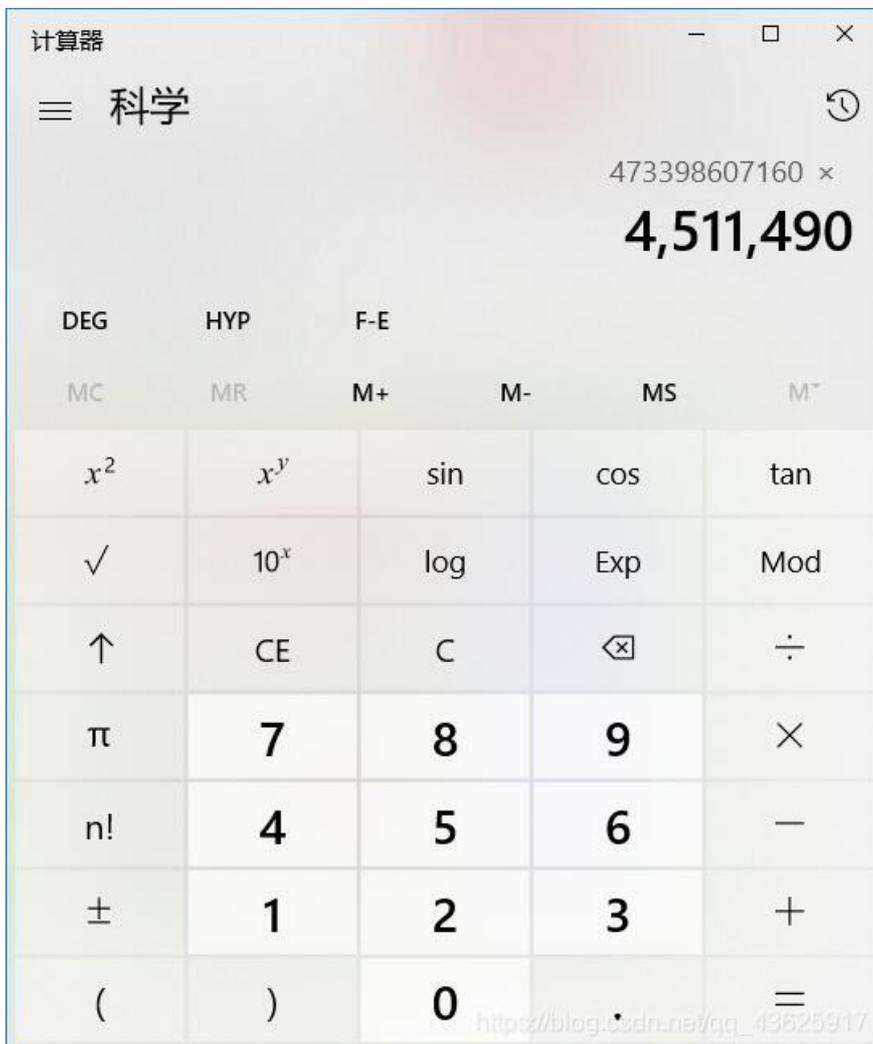
由题目可知，只要计算出私钥d即可

$$\varphi(N) = \varphi(p) \varphi(q) = (p-1)(q-1)$$

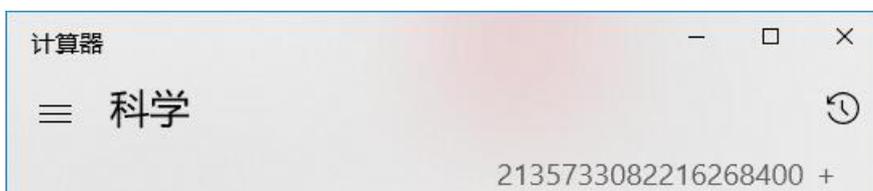
$$ed \equiv 1 \pmod{\varphi(N)}$$

所以  $ed = \varphi(N) + 1$ ，即  $17d = (p-1)(q-1) + 1 = 473398607160 * 4511490 + 1$

先计算欧拉函数  $\varphi(N)$



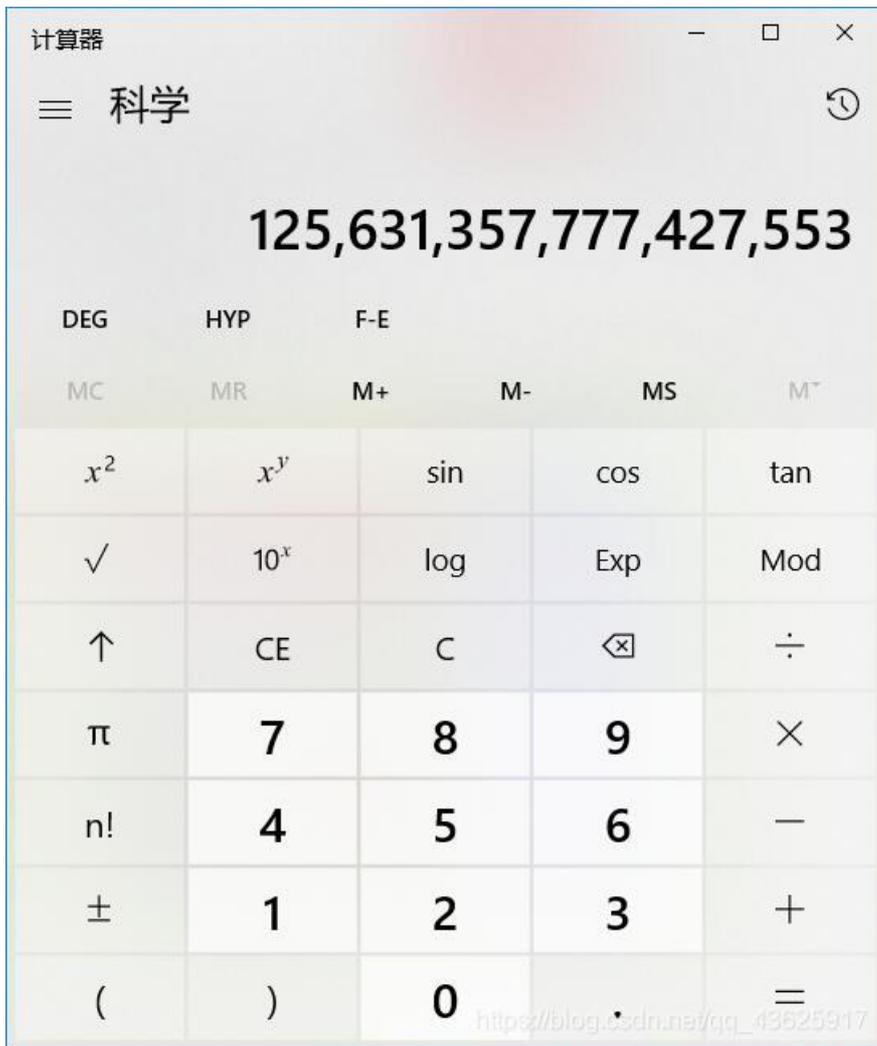
再加1



DEG	HYP	F-E			
MC	MR	M+	M-	MS	M*
$x^2$	$x^y$	sin	cos	tan	
$\sqrt{\quad}$	$10^x$	log	Exp	Mod	
$\uparrow$	CE	C	$\leftarrow \times$	$\div$	
$\pi$	<b>7</b>	<b>8</b>	<b>9</b>	$\times$	
n!	<b>4</b>	<b>5</b>	<b>6</b>	-	
$\pm$	<b>1</b>	<b>2</b>	<b>3</b>	+	
(	)	<b>0</b>	.	=	

[https://blog.csdn.net/qq\\_43625917](https://blog.csdn.net/qq_43625917)

然后除以17即是私钥d，也即是flag内容，加上flag{}提交



百度发现一个脚本，不过要安装gmpy2库

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
import gmpy2

p = 473398607161
q = 4511491
e = 17
s = (p-1)*(q-1)
d = gmpy2.invert(e,s)
print('flag is :',d)
```

```
1.py - D:\网安\做题\XCTF攻防世界\Crypto\easy_RSA\1.py (3.6.0b2)
File Edit Format Run Options Window Help
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

import gmpy2

p = 473398607161
q = 4511491
e = 17

s = (p-1)*(q-1)
d = gmpy2.invert(e, s)
print('flag is :', d)

Python 3.6.0b2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.0b2 (default, Oct 10 2016, 21:15:32) [MSC v.1900 64 bit (AMD64)] on w
in32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\网安\做题\XCTF攻防世界\Crypto\easy_RSA\1.py =====
=====
flag is : 12563135777427553
>>> |
```

## Crypto2: Normal\_RSA

### Normal RSA

查看Writeup 题目建议

难度系数: ★★ 2.0

题目来源: PCTF

题目描述: 你和小鱼走啊走走啊走，走到下一个题目一看你又一愣，怎么还是一个数学题啊 小鱼一笑，hhhh数学在密码学里面很重要的！现在知道吃亏了吧！你哼一声不服气，我知道数学 很重要了！但是工具也很重要，你看我拿工具把他解出来！你打开电脑折腾了一会还真的把答案 做了出来，小鱼有些吃惊，向你投过来一个赞叹的目光

题目场景: 暂无

题目附件: 附件1

[https://blog.csdn.net/qq\\_43625917](https://blog.csdn.net/qq_43625917)

下载题目，得到flag.enc和pubkey.pem。根据题目提示肯定是需要用到工具。RSA工具及使用

这道题目主要需要用到RSAtools和OpenSSL

### 1、用OpenSSL得到N

命令：

```
openssl rsa -pubin -text -modulus -in public.pem
```

N=C2636AE5C3D8E43FFB97AB09028F1AAC6C0BF6CD3D70EBCA281BFFE97FBE30DD

Modulus 是N的值，Exponent是e的值。

```
MINGW64:/d/网安/做题/XCTF攻防世界/Crypto/Normal_RSA
ASUS@LAPTOP-5D8T0UO2 MINGW64 /d/网安/做题/XCTF攻防世界/Crypto/Normal_RSA
$ openssl rsa -pubin -text -modulus -in pubkey.pem
RSA Public-Key: (256 bit)
Modulus:
 00:c2:63:6a:e5:c3:d8:e4:3f:fb:97:ab:09:02:8f:
 1a:ac:6c:0b:f6:cd:3d:70:eb:ca:28:1b:ff:e9:7f:
 be:30:dd
Exponent: 65537 (0x10001)
Modulus=C2636AE5C3D8E43FFB97AB09028F1AAC6C0BF6CD3D70EBCA281BFFE97FBE30DD
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAMJjauXD20Q/+5erCQKPGqxsC/bNPXDr
yigb/+1/vjDdAgMBAAE=
-----END PUBLIC KEY-----
ASUS@LAPTOP-5D8T0UO2 MINGW64 /d/网安/做题/XCTF攻防世界/Crypto/Normal_RSA
$
```

[https://blog.csdn.net/qq\\_43625917](https://blog.csdn.net/qq_43625917)

16进制转10进制

十进制

87924348264132406875276140514499937145050893665602592992418171647042491658461

十六进制

C2636AE5C3D8E43FFB97AB09028F1AAC6C0BF6CD3D70EBCA281BFFE97FBE30DD

[https://blog.csdn.net/qq\\_43625917](https://blog.csdn.net/qq_43625917)

## 2、分解N的值，得到p、q

在线大数分解

Search	Sequences	Report results	Factor tables	Status	Downloads	Login
<input type="text" value="87924348264132406875276140514499937145050893665602592992418171647042491658461"/>						<input type="button" value="Factorize!"/> (?)
<b>Result:</b>						
status (?)	digits	number				
FF	77 (show)	8792434826...61<77> = 275127860351348928173285174381581152299<39> · 319576316814478949870590164193048041239<39>				
<b>More information</b>						
<b>ECM</b>						

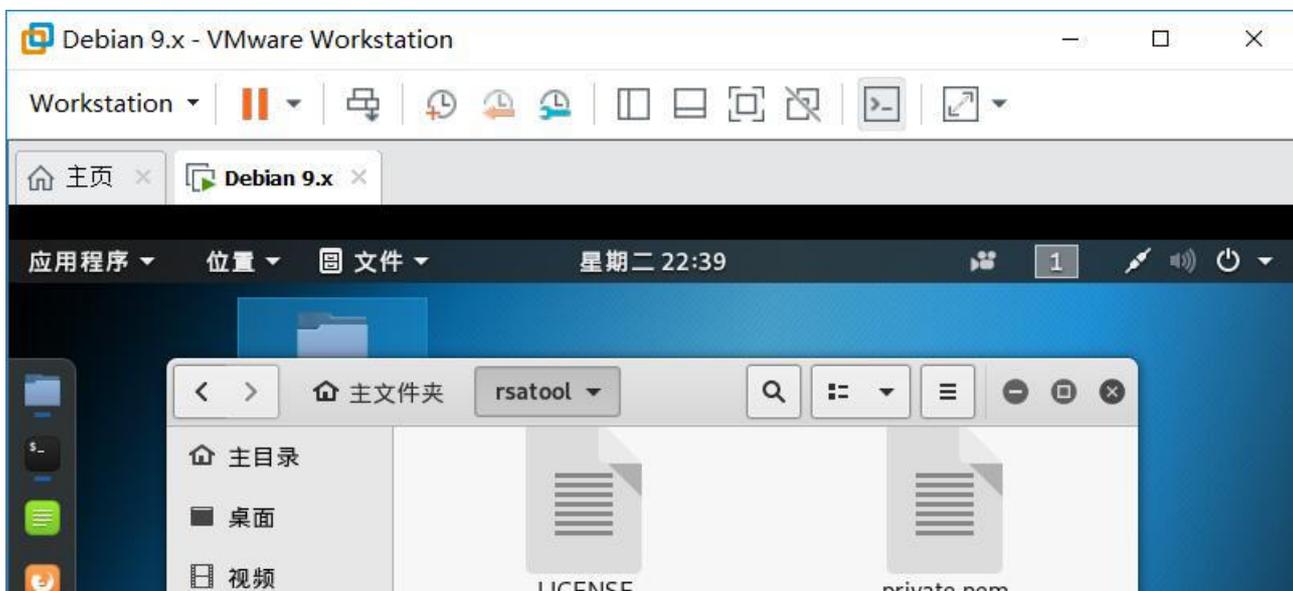
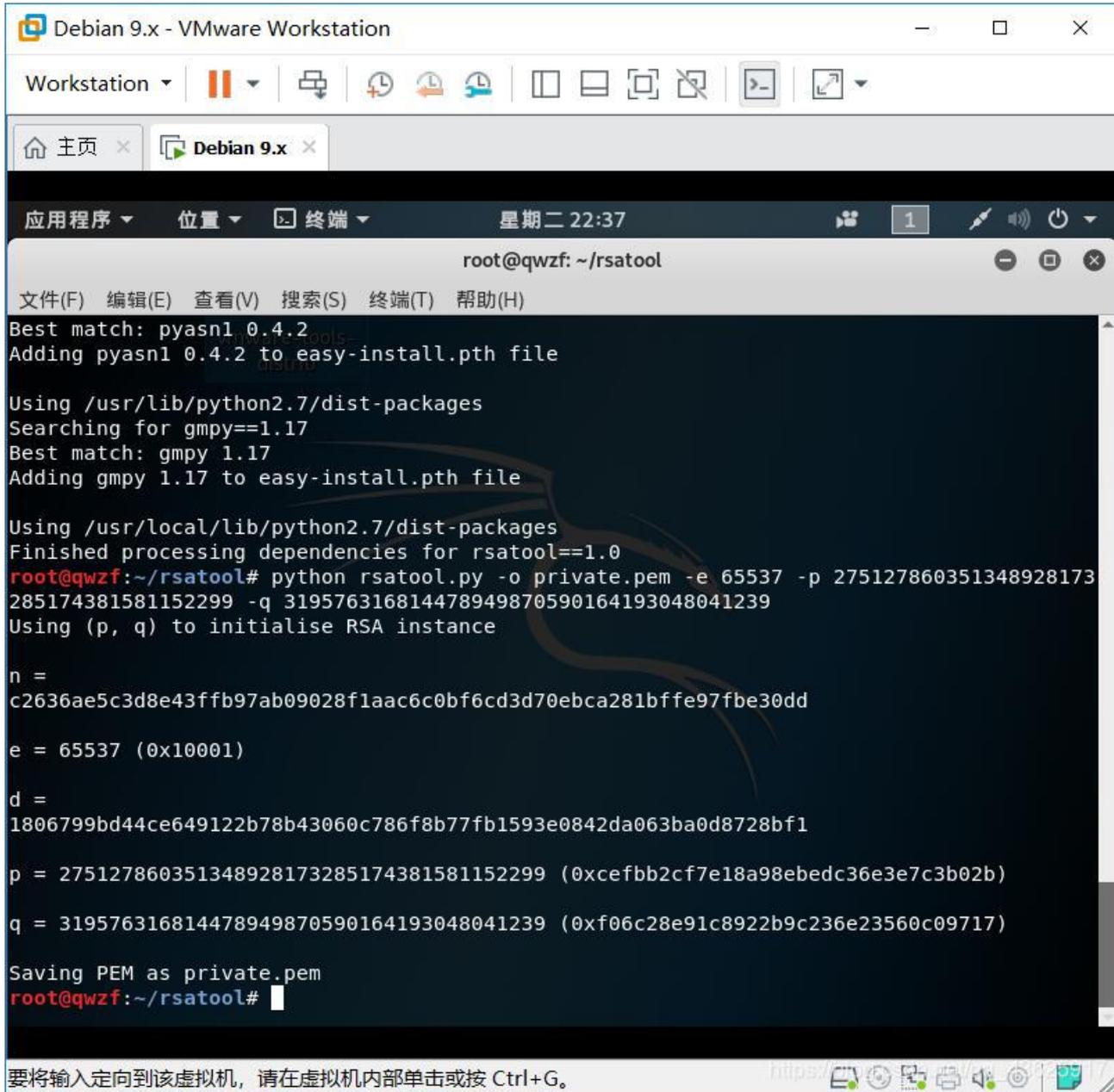
[https://blog.csdn.net/qq\\_43625917](https://blog.csdn.net/qq_43625917)

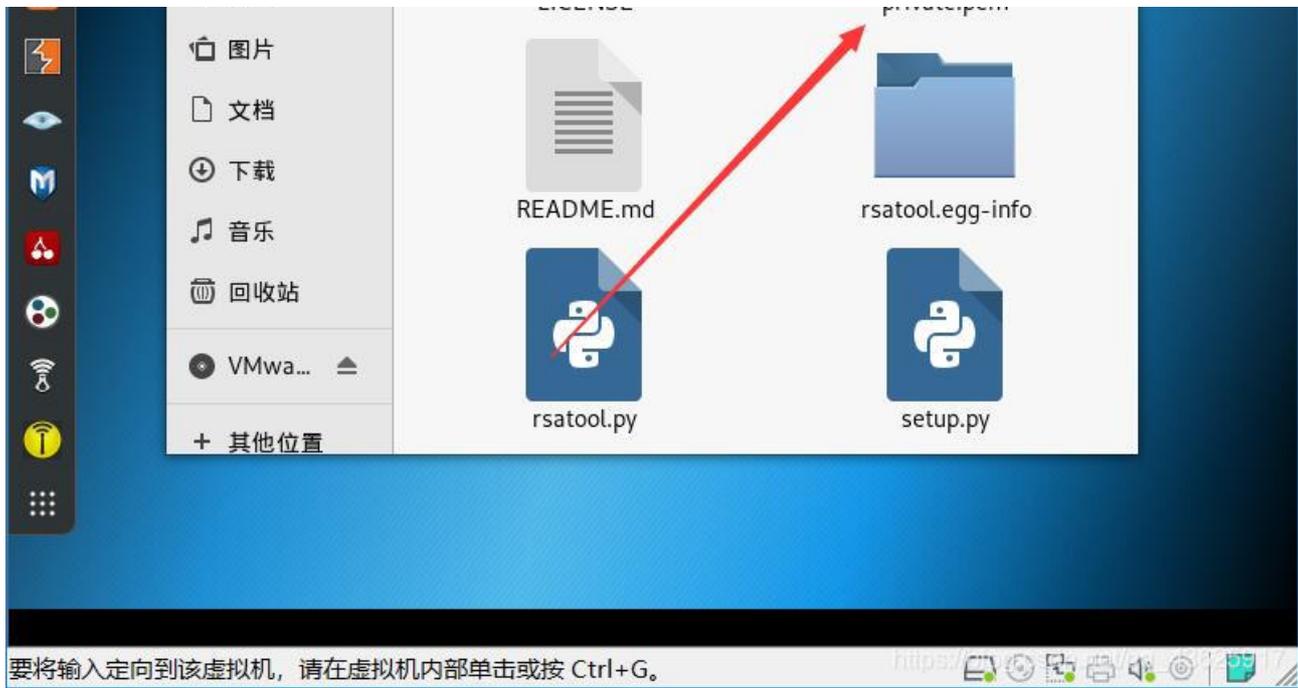
即p=275127860351348928173285174381581152299、q=319576316814478949870590164193048041239

## 3、用RSAtools生成私钥文件private.pem

命令：

```
python rsatool.py -o private.pem -e 65537 -p 275127860351348928173285174381581152299 -q 319576316814478949870590164193048041239
```





#### 4、用生成的private.pem和OpenSSL对flag.enc文件进行解密

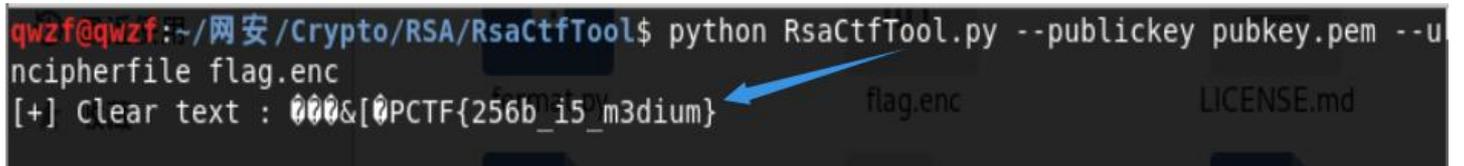
命令：

```
openssl rsautl -decrypt -in flag.enc -inkey private.pem
```



得到flag

当然也有个强大的工具RsaCtfTool



### 感悟

学习了RSA加密算法之后，对RSA加密解密有了认识。同时为了能顺利使用rsatool，特意又搭了Kali虚拟机。

果然Kali真的是强大的一批。小白进阶ing