

## RSA算法

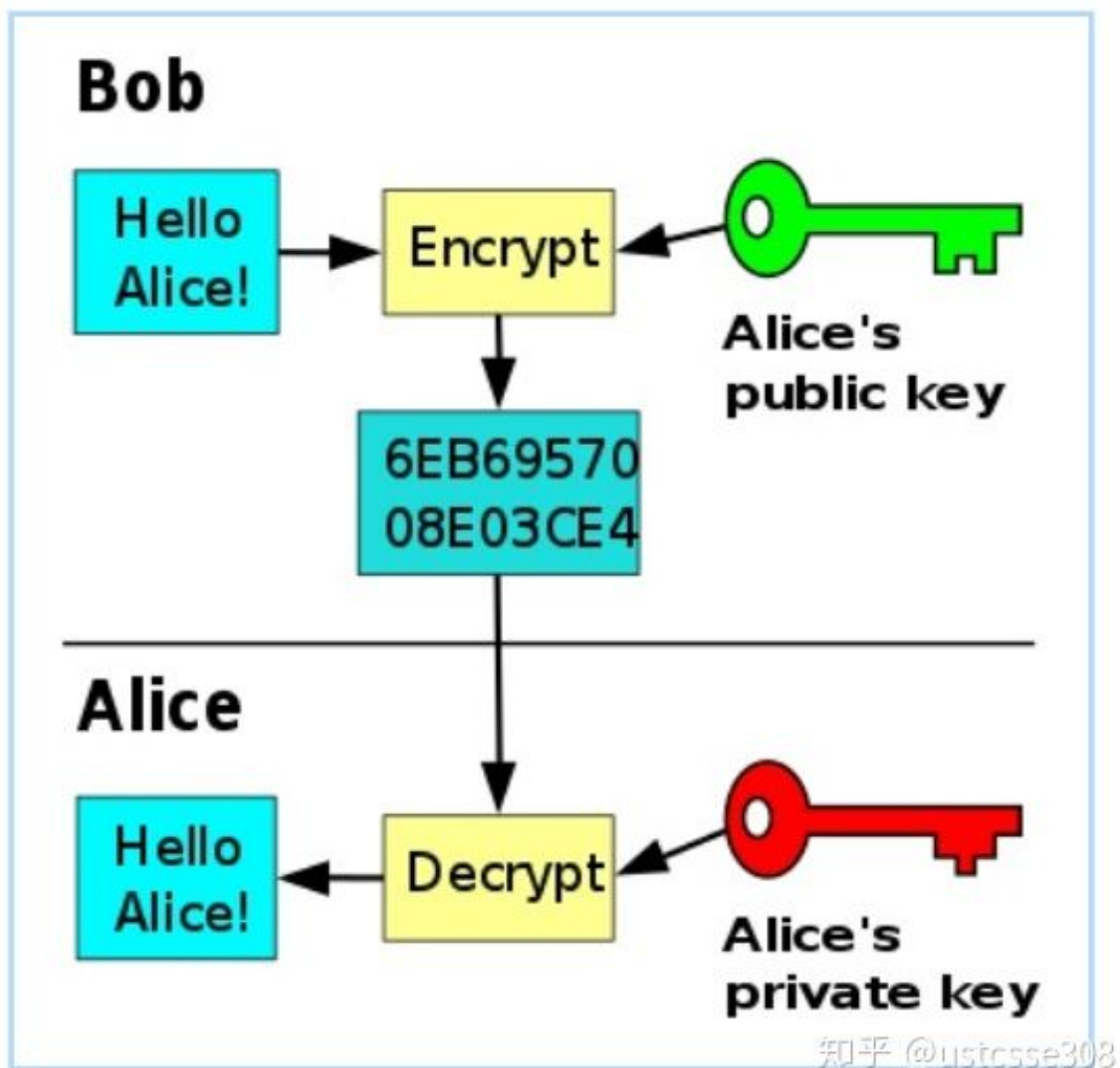
- RSA算法的流程

首先看一下RSA算法的流程。

RSA算法分为密钥生成和密钥使用。



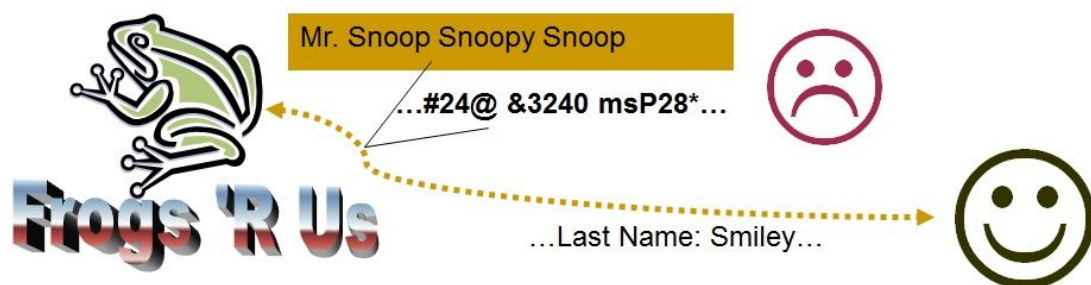
Alice可以使用大质数为基础生成一对密钥，分别是公钥和私钥，一般使用公钥来加密（encryption），私钥来解密(decryption)，所以一般用e表示公钥，d表示私钥。



在Alice生成公私钥对之后，公钥公开，所有人可见，如果Bob想给Alice发送加密信息，就可以使用Alice的公钥对信息进行加密；当Alice收到密文之后，使用自己的私钥进行解密。

在这个过程中，Alice的公钥和Bob加密之后的密文都是对攻击者Eve可见的。

这里举一个具体的例子。



知乎 @ustcsse308

Frog公司提供了在线服务，用户希望向Frog提供一条消息，“Last Name: Smiley”。为了防止被窃听，消息需要被加密。



Frog公司公开了一对数字  $(N, e)$  。

$N = 4559, e = 13.$

$m^e \bmod N$



Smiley Transmits: "Last name Smiley"

- ♦ L A S T N A M E S M I L E Y
- ♦ 1201 1920 0014 0113 0500 1913 0912 0525
- ♦  $1201^{13} \bmod 4559, 1920^{13} \bmod 4559, \dots$
- ♦ 2853 0116 1478 2150 3906 4256 1445 2462

知乎 @ustcsse308

RSA加密过程  $4559 = 97 \times 47$

$$N = 4559, d = 3397$$

- 2853 0116 1478 2150 3906 4256 1445 2462
- $2853^{3397} \bmod 4559, 0116^{3397} \bmod 4559, \dots$
- |      |      |      |       |      |      |      |      |
|------|------|------|-------|------|------|------|------|
| 1201 | 1920 | 0014 | 0113  | 0500 | 1913 | 0912 | 0525 |
| L A  | S T  |      | N A M | E    | S M  | I L  | E Y  |



~~知平 @ustcsse303~~

## RSA解密过程

### RSA过程:

具体的例子：

1. 挑选两个质数，如  $p=61$  和  $q=53$
2. 计算  $N = p \cdot q = 3233$
3. 计算  $(p-1)(q-1) = 60 \cdot 52 = 3120$  【这一步可以计算  $(p-1)$  和  $(q-1)$  的最小公倍数，从而使得计算的  $d$  比较小；17 关于 780 的模逆是 413，比 2753 要小】
4. 选择与 3120 互质的一个数  $e = 17$
5. 计算得出  $d$ ，使得  $d$  是  $e$  关于 3120 的模逆，得出  $d = 2753$ （模逆可以使用 Euclid 扩展算法，证明略）
6. 如果明文是 5，那么密文是  $5^{17} \pmod{3233} = 3086$
7. 解密， $3086^{2753} \pmod{3233} = 5$

实际中的RSA公钥长这样：



```
String publicKey =  
"3082010a0282010100c70e6c3f23937fcc70a59d20  
c30e533f7ec04ec29849ca47d523ef03348574c8a30  
22e465c0b7dc9889d4f8bf0f89c6c8c5535dbbffa2b3  
eafbe356e74a46d91322ca36d59bc1a8e3964393f20  
cbce6f9e6e899c86348787f5736691a191d5ad1d47d  
c29cd47fe18012ae7aea88ea57d8ca0a0a3a1249a26  
2197a0d24f737ebb473927b05239b12b5ceeb29dfa4  
1402b901a5d4a69c436488def87efee3f51ee5fedca  
3a8e46631d94c25e918b9895909aee99d1c6d370f4a  
1e352028e2afd4218b01c445ad6e2b63ab926b610a4  
d20ed73ba7ccfe16b5db9f80f0d68b6cd908794a4f  
7865da92bcbe35f9b3c4f927804eff9652e60220e10  
773e95d2bbdb2f10203010001"
```

这一大串数字是N

e = 0x10001 = 65537 知乎 @ustcsse308

- RSA不被破解的原因

为了解密，关键是要找出私钥。如果已知  $(p-1)(q-1)$ ，那么就很容易算出来私钥。而为了获得  $(p-1)(q-1)$ ，就需要知道p和q的值。为了获得p和q的值，就必须对N进行因式分解。

1874年，William Stanley Jevons就在自己的书《科学的原则》中写道：

读者中有人能发现是哪两个数的乘积为8616460799吗？我想这个答案只有我自己知道。

书中他描述了单向函数（one-way function）与密码学的关系，还提出了因子分解问题可以用作创建trapdoor函数。

到目前为止，关于RSA可靠性的描述：

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。

假如有人找到一种快速因数分解的算法，那么RSA的可靠性就会极度下降。但找到这样的算法的可能性是非常小的。今天只有短的RSA密钥才可能被暴力破解。到2008年为止，世界上还没有任何可靠的攻击RSA算法的方式。

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。

12301866845301177551304949

58384962720772853569595334

79219732245215172640050726

36575187452021997864693899

56474942774063845925192557

32630345373154826850791702

61221429134616704292143116

02221240479274737794080665

351419597459856902143413

这是人类已经分解的最大整数（768）个二进制位，它等于：

33478071698956898786044169

84821269081770479498371376  
85689124313889828837938780  
02287614711652531743087737  
814467999489

x

36746043666799590428244633  
79962795263227915816434308  
76426760322838157396665112  
79233373417143396810270092  
798736308917

比它更大的因数分解，还没有被报道过。