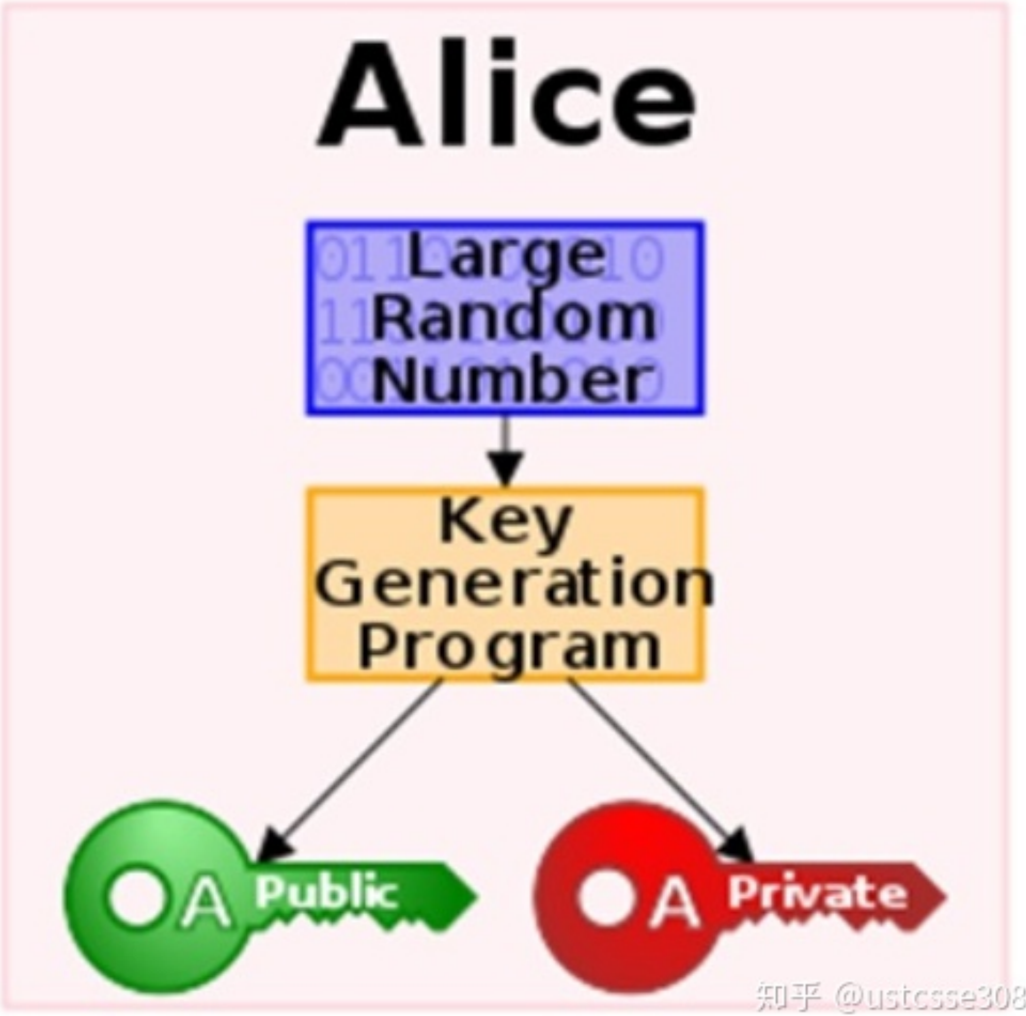
RSA算法

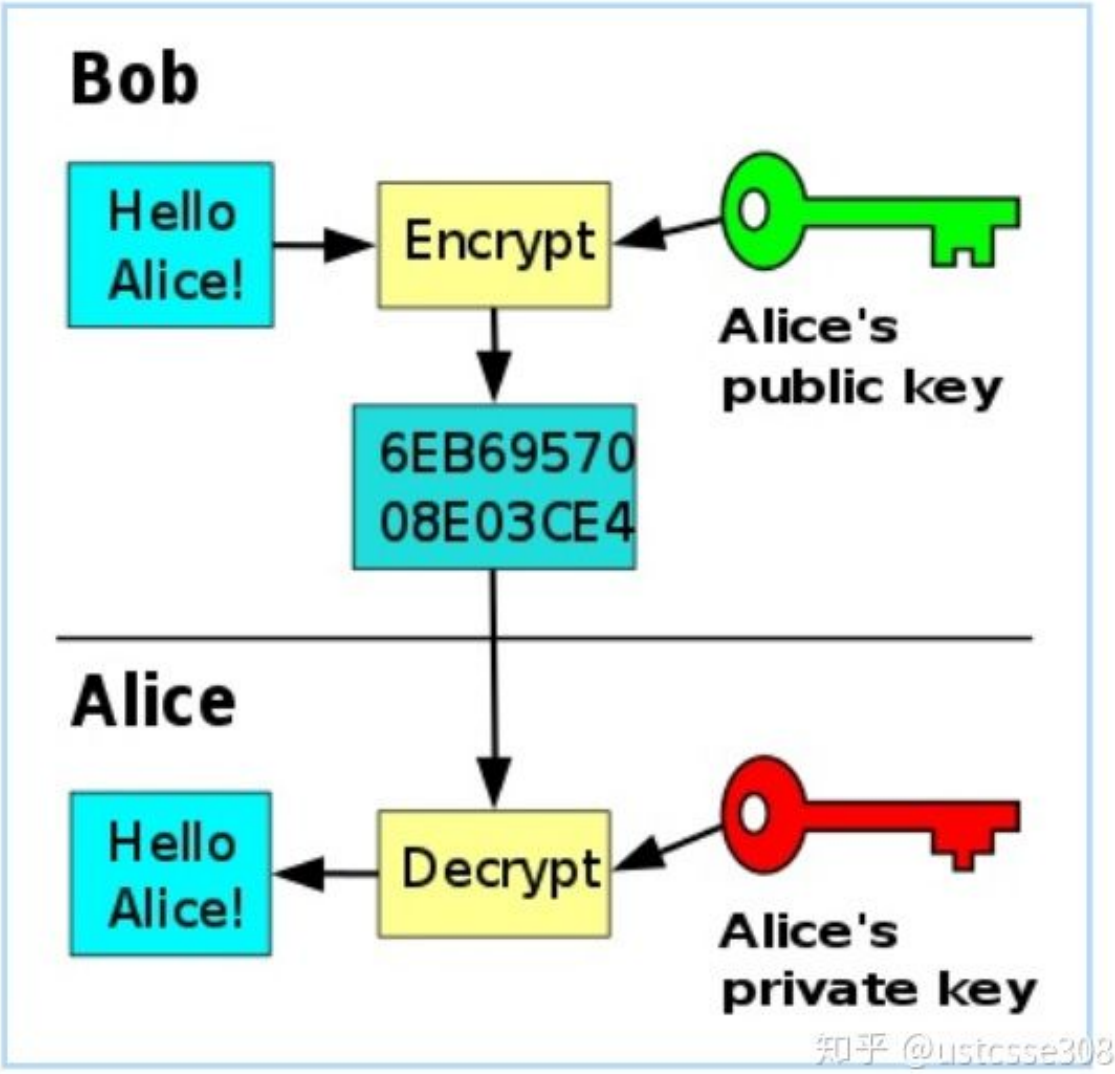
- RSA算法的流程

  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。



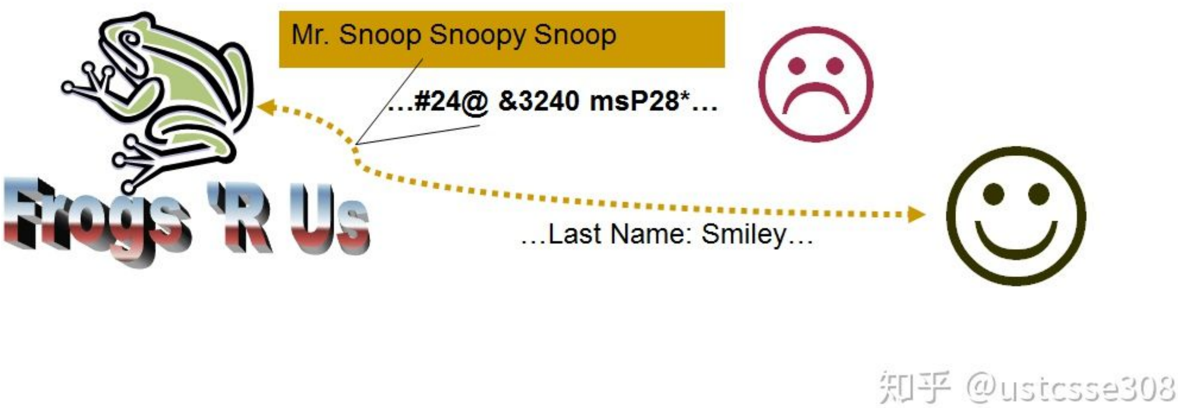
Alice可以使用大质数为基础生成一对密钥，分别是公钥和私钥，一般使用公钥来加密（encryption），私钥来解密(decryption)， 所以一般用e表示公钥，d表示私钥。



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。

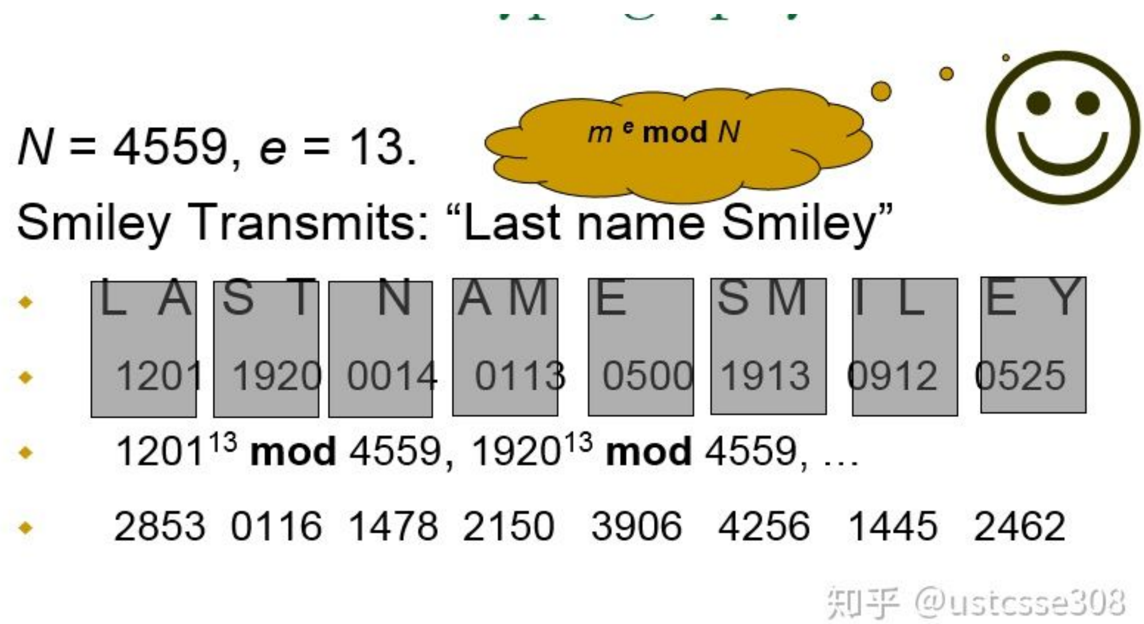
在这个过程中，Alice的公钥和Bob加密之后的密文都是对攻击者Eve可见的。

这里举一个具体的例子。

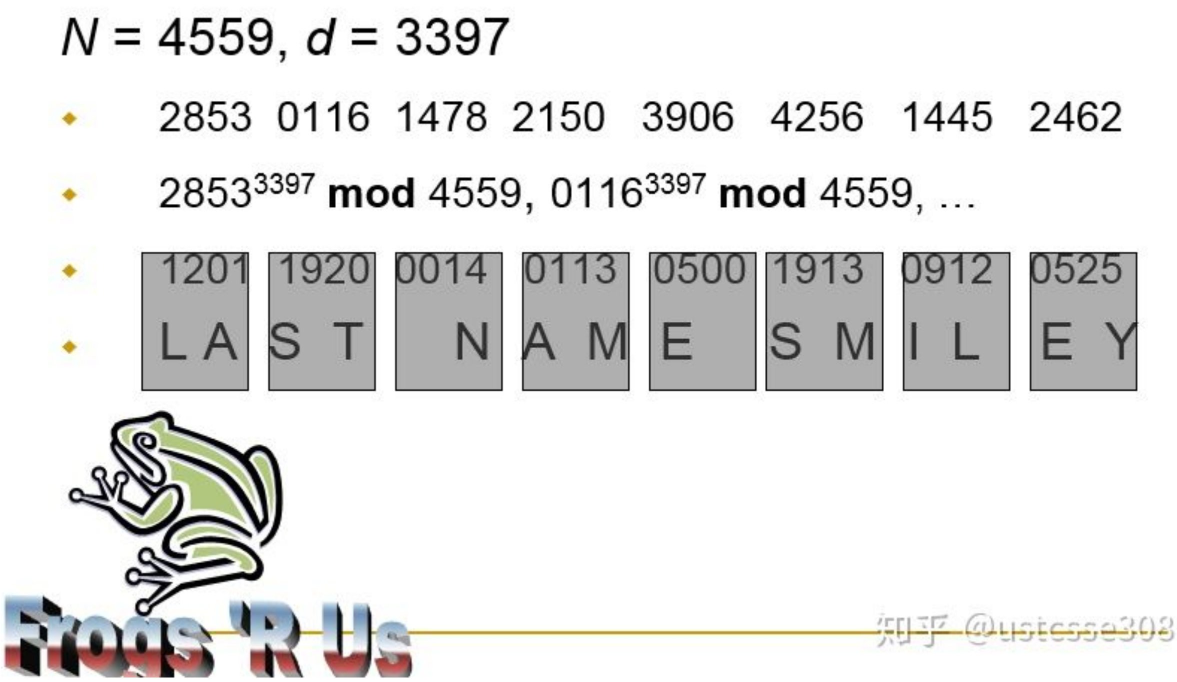


Frog公司提供了在线服务，用户希望向Frog提供一条消息，“Last Name: Smiley”。为了防止被窃听，消息需要被加密。

Frog公司公开了一对数字（N，e）。



RSA加密过程 4559 = 97*47



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*q = 3233
3. 计算（p-1）(q-1) = 60*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 (mod 3233) = 3086
7. 解密，3086^2753 (mod 3233) = 5

实际中的RSA公钥长这样：

- 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

×

36746043666799590428244633
79962795263227915816434308
76426760322838157396665112
79233373417143396810270092
798736308917

比它更大的因数分解，还没有被报道过。