问题1：314159265358979 × 314159265358979等于多少？

问题2：3912571506419387090594828508241的平方根是多少？

如果给一张纸和一支笔，加上一大杯冰激凌作为正确答案的奖励，那么大多数六年级学生可以在一两个小时内做出问题1的答案。而如果给一般成年人纸和笔，并许诺回答出正确答案可以免去终身50%税收的话，大多数人还是不能在没有计算器、计算机或其他外界帮助的条件下解答出问题2的答案。虽然平方和求平方根互为逆运算，但是它们在计算的复杂性上却有很大差异。这种不对称性构成了公钥密码体系的基础。在公钥密码体系中，加密运算比较简单，而没有密钥的解密运算却十分繁琐。

一种叫做RSA的公钥机制表明：对计算机来说，大数乘法比对大数进行因式分解要容易得多，特别是在使用取模算法进行运算且每个数字都有上百位时（Rivest等人，1978）。这种机制广泛应用于密码领域。其他广泛使用的还有离散对数（El Gamal，1985）。公钥机制的主要问题在于运算速度要比对称密钥机制慢数千倍。

当我们使用公钥密码体系时，每个人都拥有一对密钥（公钥和私钥）并把其中的公钥公开。公钥是加密密钥，私钥是解密密钥。通常密钥的运算是自动进行的，有时候用户可以自选密码作为算法的种子。在发送机密信息时，用接收方的公钥将明文加密。由于只有接收方拥有私钥，所以也只有接收方可以解密信息。

9.2.3 单向函数

在接下来的许多场合里，我们将看到有些函数f，其特性是给定f和参数x，很容易计算出$y = f(x)$。但是给定$f(x)$，要找到相应的x却不可行。这种函数采用了十分复杂的方法把数字打乱。具体做法可以首先将y初始化为x。然后可以有一个循环，进行多次迭代，只要在x中有1位就继续迭代，随着每次迭代，y中的各位的排列以与迭代相关的方式进行，每次迭代时添加不同的常数，最终生成了彻底打乱位的数字排列。这样的函数叫做*加密散列函数*。

9.2.4 数字签名

经常性地使用数字签名是很有必要的。例如，假设银行客户通过发送电子邮件通知银行为其购买股票。一小时后，定单发出并成交，但随后股票大跌了。现在客户否认曾经发送过电子邮件。银行当然可以出示电子邮件作为证据，但是客户也可以声称是银行为了获得佣金而伪造了电子邮件。那么法官如何来找到真相呢？

通过对邮件或其他电子文档进行数字签名可以解决这类问题，并且保证了发送方日后不能抵赖。其中的一个通常使用的办法是首先对文档运行一种单向散列运算（hashing），这种运算几乎是不可逆的。散列函数通常独立于原始文档长度产生一个固定长度的结果值。最常用的散列函数有MD5（Message Digest 5），一种可以产生16个字节结果的算法（Rivest，1992）以及SHA-1（Secure Hash Algorithm），一种可以产生20个字节结果的算法（NIST，1995）。比SHA-1更新版本有SHA-256和SHA-512，它们分别产生32字节和64字节的散列结果，但是迄今为止，这两种加密算法依然没有得到广泛使用。

下一步假设我们使用上面讲过的公钥密码。文件所有者利用他的私钥对散列值进行运算得到D(散列值)。该值称为签名块（signature block），它被附加在文档之后传送给接收方，如图9-3所示。对散列值应用D有些像散列解密，但这并不是真正意义上的解密，因为散列值并没有被加密。这不过是对散列值进行的数学变换。

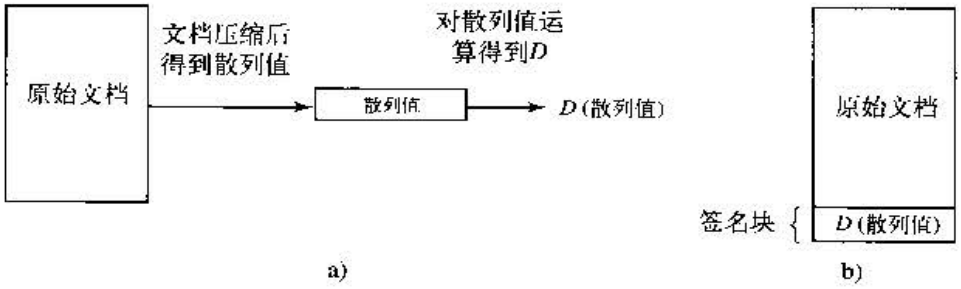

图9-3 a) 对签名块进行运算；b) 接收方获取的信息

接收方收到文档和散列值后，首先使用事先取得一致的MD5或SHA算法计算文档的散列值，然后