第5章　公钥密码

5.1　简介

前面介绍了传统密码体制中的对称加密体制，本章要讲述另一种重要的密钥体制——公钥密码体制。公钥密码学的发展是整个密码学发展历史中最伟大的一次革命，也许可以说是唯一的一次革命。在公钥密码体制产生和应用之前的整个密码学史中，所有的密码算法基本上都是基于代换和置换这两种方法。几千年来，对算法的研究主要是通过手工计算来完成的。随着转轮加密/解密机器的出现，传统密码学有了很大进展，利用电子机械转轮可以开发出极其复杂的加密系统，利用计算机甚至可以设计出更加复杂的系统，最著名的就是数据加密标准 DES。转轮机和 DES 是密码学发展的重要标志，但它们都基于代换和置换这些初等方法之上。

而公钥密码学与以前的密码学完全不同。公钥密码的发展提供了新的理论和技术基础，同时也是密码学发展的里程碑。算法的基本工具突破传统的代换和置换，公钥密码使用的基本工具是数学函数；另一方面公钥密码是非对称的，使用两个独立的密钥，两个密钥的使用对机密性、密钥分配、数字签名等都有划时代的意义。

公钥密码体制的概念是在解决对称密码体制中最难解决的两个问题时提出的，这两个问题是密钥分配和数字签名。

对称密码体制在进行密钥分配时，要求通信双方或者已经有一个共享的密钥，或者可以借助一个密钥分配中心来分配密钥。对前者的要求，常常可用人工方式传送双方最初共享的密钥，但是这种方法成本很高，而且还要依赖于通信过程的可靠性，这同样是一个安全隐患；对于第二个要求则完全依赖于密钥分配中心的可靠性，同时密钥中心往往需要很大的存储容量来处理大量的密钥。

第二个问题数字签名考虑的是如何为数字化的消息或文件提供一种类似于为书面文件手写签字的方法。这个对于对称密码体制是一个非常难以解决的问题。而随着社会的发展，尤其电子商务的发展，数字签名问题必须得到好的解决。W. Diffie 和 M. Hellman 为解决上述两个问题，从而提出了公钥密码体制。

对称加密和公开加密的一个显著区别体现在密钥协商上：使用对称加密的双方需要实时协商密钥；而使用公开加密时，用户的公私钥对是收发双方事先按照某种算法产生的，所以完全避开了密钥协商的步骤，方便陌生人进行加密通信。

对称加密的优点是加解密速度快，尤其针对长报文时；但它的缺点也比较明显，系统中的密钥数量多而且难以安全分发，同时不能对报文进行发送端鉴别。这些不足说明需要产生新的加密算法。

与对称加密算法比较而言，公钥密码的优点是：系统中需要产生密钥个数少，而且可

以公开分发密钥，通信双方事先不需要通过保密信道交换密钥；同时可以实现数字签名进行发送端鉴别。这使得公钥密码特别适用于互联网这种大规模通信的环境。但是它也有显著的缺点：加解密速度慢，不适合直接加密明文。

公钥密码是密码学上的重大突破，但初学者会对公钥密码有一些误解，这里先提前说明。一种误解是，从密码分析的角度看，公钥密码比传统密码更安全。事实上，任何加密方法的安全性依赖于密钥的长度和破译密文所需要的计算量。从抗密码分析的角度看，原则上不能说传统密码优于公钥密码，也不能说公钥密码优于传统密码。

第二种误解是，公钥密码是一种通用的方法，所以传统密码已经过时。其实正相反，由于现有的公钥密码方法所需的计算量大，所以取代传统密码似乎不太可能。就像公钥密码的发明者之一 W. Diffie 所说的："公钥密码学仅限于用在密钥管理和签名这类应用中，这几乎是已被广泛接受的事实。"

第三种误解是，对称密码中与密钥分配中心的握手是一件异常麻烦的事情，与之相比，用公钥密码实现密钥分配则非常简单。事实上，使用公钥密码也需要某种形式的协议，该协议通常包含一个中心代理，并且它所包含的处理过程既不比对称密码中的那些过程更简单，也不比之更有效。

本章先介绍公钥密码设计的基本思想与概念，接着介绍 Diffie-Hellman 密钥分配协议。密钥分配协议不是一个公钥密码的加解密算法，但很多人认为它是公钥密码思想的起源，它是密码学中一个惊人的成就，该算法的作者也因此获得计算机理论研究的最高奖项——图灵奖。然后讨论 RSA 算法，它是目前公钥密码中最重要的一种切实可行的加解密算法，RSA 的三位作者也获得了图灵奖。最后我们介绍椭圆曲线密码算法的一些知识，该领域是公钥密码研究的一个热点。

5.1.1 公钥密码体制的设计原理

公钥密码算法最重要的特点是采用两个相关的密钥将加密与解密能力分开，其中一个密钥是公开的，称为公开密钥，用来加密；另一个密钥是用户专用，是保密的，称为私有密钥，用于解密。加密和解密使用的密钥是不同的，因此公钥密码体制也称为双钥密码体制。公钥密码算法的重要特性：已知密码算法和加密密钥，想得到解密密钥在计算上是不可行的。

图 5.1 是公钥密码体制加密的框图，加密过程主要有以下几步。

(1) 系统中要求接收消息的端系统，产生一对用来加密和解密的密钥，如图 5.1 中的接收者 B，需要产生一对密钥 PK_B 和 SK_B，其中 PK_B 是公钥，SK_B 是私钥。

(2) 端系统 B 将公钥 PK_B 放在一个公开的寄存器或文件中，通常放入存放密钥的密钥中心中。另一私钥 SK_B 则被用户保存。

(3) A 如果要想向 B 发送消息 m，则首先必须得到并使用 B 的公钥 PK_B 加密 m，表示为

$$c = E_{PK_B}[m]$$

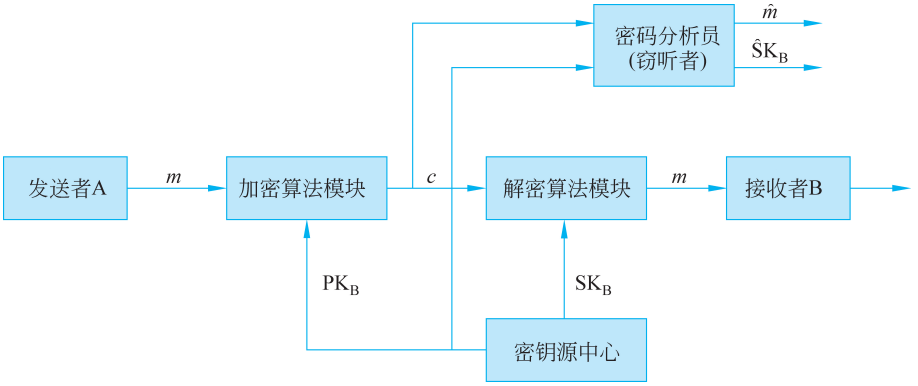

图 5.1　公钥密码体制加密的框图

其中,c 是密文;E 是加密算法。

(4) B 收到 A 的加密密文 c 后,用自己的私钥 SK_B 解密得到明文信息,表示为

$$m = D_{SK_B}[c]$$

其中,D 是解密算法。

因为整个过程中只有 B 知道 SK_B,所以其他人都无法对 c 解密,从而信息得到机密性保护。作为密码分析员可以观察到密文 c 并且可以得到公钥 PK_B,但是他不能访问私钥 SK_B 或者明文 m,所以密码分析者的目的是恢复私钥 SK_B 或者明文 m。如果密码分析者知道加密(E)和解密(D)算法,如果他只关心 m 这一个明文消息,那么他会集中精力试图通过生成明文估计值来恢复 $\hat{m}$。但是通常密码分析者也希望能获得其他消息,所以他会试图破解私钥 SK_B。

公钥密码不仅能用于保密通信,还可以提供不可否认性,这一点是对称密码由于自身特点而无法实现的。

首先解释一下不可否认,在使用对称密码保护的情况下,假设 A 从她的股票经纪人 B 处订购了 100 股股票,为了保护订单的机密性和完整性,A 使用共享对称密钥 K_{AB} 加密,假设 A 下了订单不久后并在向 B 付钱之前,股票暴跌 90%。这时 A 宣布她从没有下过订单,也就是说 A 否认了这笔交易。

那么 B 能否证实 A 曾经下过订单呢,不,他不能。因为 B 也知道对称密钥 K_{AB},他可以假冒 A 在订单上放置伪造的消息。因此,尽管 B 知道 A 确实下了订单,但是却不能证明这一点。如何防止 A 否认这笔交易呢?可以使用数字签名,数字签名就是通过某种密码运算生成一系列符号及代码组成电子密码进行签名,来代替书写签名或印章,对于这种电子签名还可进行技术验证。

公钥密码可以实现数字签名,信息发送者可使用公钥密码产生别人无法伪造的一段数据串。事实上,确保数据机密性只是公钥体系的用途之一,它还有一个非常重要的用途就是对信息进行数字签名,防止信息发送者抵赖或第三方冒充发送者发送信息。对称加密算法不能实现这个功能,那为什么公开加密机制可以实现此功能呢?很简单,还是使用了“公钥加密,只有私钥能解密;私钥加密,只有公钥能解密”的原理。

发送者用自己的私钥加密数据后传给接收者，接收者用发送者的公钥解开数据后，就可以确定消息来自于谁，同时也是对发送者消息真实性的一个证明，发送者对所发消息不可否认。如图 5.2 所示，用户 A 用自己的私钥 SK_A 对明文 m 进行加密，过程表示为

$$c = E_{SK_A}[m]$$

将密文 c 发送给 B。B 用 A 提供的公钥 PK_A 对 c 进行解密，该过程可以表示为

$$m = D_{PK_A}[c]$$

因为从 m 得到 c 是经过 A 的私钥 SK_A 加密，也只有 A 才能做到，因此 c 可以当作 A 对 m 的数字签名。另一方面，任何人只要得不到 A 的私钥 SK_A 就不能篡改 m，所以以上过程实现了对消息来源和消息完整性的认证功能。

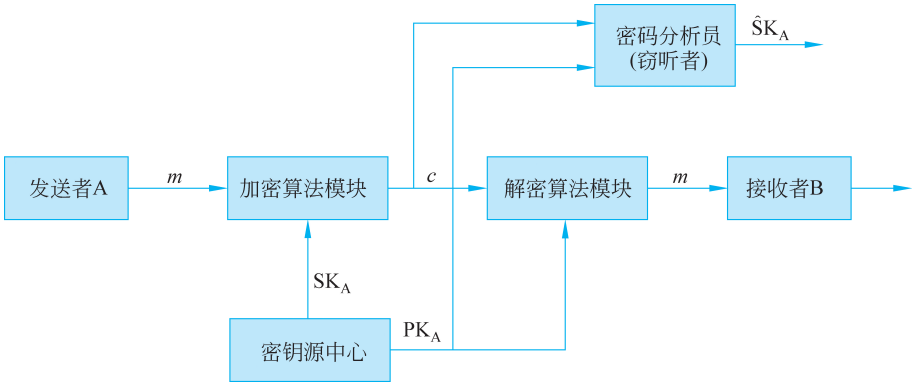

图 5.2 公钥密码体制数字签名原理框图

在这个数字签名的过程中，发送者使用私钥加密实现签名的功能，接收者使用公钥解密实现验证的功能，这与前面提到的加密过程相反。使用公钥密码加密时，发送者使用公钥加密，接收者使用私钥解密。在许多公钥算法中，两个密钥中任何一个都可用来加密，而另一个用来解密，可分别实现加解密和数字签名的功能。

在以上数字签名过程中，由于消息是由用户自己的私钥加密的，所以消息不可能被他人篡改，但却能很容易被他人窃听，这是由于任何人都能使用用户的公钥对消息解密。因此为了同时提供认证功能和保密性，可采用双重加、解密。原理图如图 5.3 所示。

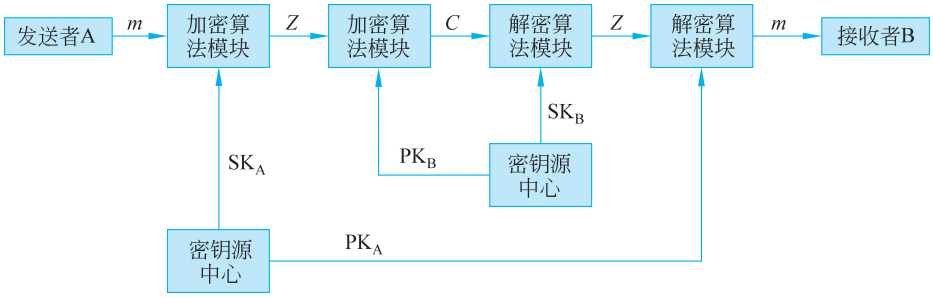

图 5.3 公钥密码体制的认证、保密原理图

发送方首先用自己的私钥 SK_A 对消息 m 进行加密,用于提供数字签名功能。然后用接收方的公钥 PK_B 进行第二次加密操作,表示为 $c=E_{PK_B}[E_{SK_A}[m]]$,解密过程为 $m=D_{PK_A}[D_{SK_B}[c]]$,即接收方用自己的私钥和发送方的公钥对收到的密文进行两次解密操作。

一般来讲,公钥加密算法应满足以下几点基本要求。

(1) 接收方 B 产生密钥对(公钥 PK_B 和私钥 SK_B)是很容易计算得到的。

(2) 发送方 A 用收到的公钥对消息 m 加密以产生密文 c,即 $c=E_{PK_B}[m]$,很容易通过计算得到。

(3) 接收方 B 用自己的私钥对密文 c 解密,即 $m=D_{SK_B}[c]$ 在计算上是容易的。

(4) 密码分析者或者攻击者由 B 的公钥 PK_B 求私钥 SK_B 在计算上是不可行的。

(5) 密码分析者或者攻击者由密文 c 和 B 的公钥 PK_B 恢复明文 m 在计算上是不可行的。

(6) 加密、解密操作的次序可以互换,也就是 $E_{PK_B}[D_{SK_B}(m)]=D_{SK_B}[E_{PK_B}(m)]$。

以上要求中最后一条虽然非常有用,但并不是对所有算法都有此要求。

以上要求的本质是要求一个陷门单向函数。所谓陷门单向函数是两个集合 X、Y 之间的一个映射,使得 Y 中每一元素 y 都有唯一的一个原像 $x \in X$,且由 x 易于计算它的像 y。但是由 y 计算它的原像 x 在计算上是不可行的。这里所说的易于计算是指函数值能在其输入长度的多项式时间内求出,即如果输入长 n 比特,则求函数值的计算时间是 n^a 的某个倍数,其中 a 是一个固定常数。这时认为求函数值的算法属于可计算,否则就是不可行的。注意这里的可计算和不可行两个概念与计算复杂性理论中复杂度的概念非常相似,同时存在着本质的区别。在复杂性理论中,算法复杂度是用算法在最坏的情况下或平均情况时的复杂度来度量的。而这里所说的两个概念是指算法在几乎所有情况下的情景。称一个函数是陷门单向函数,是指该函数是易于计算的,但求它的逆过程是不可行的,除非在已知某些附加信息的前提下。当附加信息给定后,求逆可在一定时间内完成。

所以总结为:陷门单向函数是一族可逆函数 f_k,但是满足以下 3 个条件。

(1) $Y=f_k(X)$ 易于计算(当 k 和 X 已知时)。

(2) $X=f_k^{-1}(Y)$ 易于计算(当 k 和 Y 已知时)。

(3) $X=f_k^{-1}(Y)$ 在计算上是不可行的(当 Y 已知但 k 未知时)。

5.1.2 公钥密码分析

与对称密码一样,公钥密码也易受穷举攻击,其解决方法也是使用长密钥。但同时也应考虑使用长密钥的利弊,公钥体制使用的是某种可逆的数学函数,计算函数值的复杂性可能不是密钥长度的线性函数,而是比线性函数增长更快的函数。因此,为了抗穷举攻击,密钥必须足够长;同时为了便于实现加密和解密,密钥必须足够短。在实际中,现在使用的密钥长度确实可以抗穷举攻击,但是它也使加密/解密速度太慢,所以公钥密码目前主要用于密钥管理和签名中。

对公钥密码的另一种攻击方法是，找出一种给定的公钥计算出私钥的方法。到目前为止，还未在数学上证明对一特定公钥算法这种攻击是不可行的，所以包括已被广泛使用的 RSA 在内的任何算法都是值得怀疑的。密码分析的历史表明，同一个问题从一个角度看是不可解的，但从另一个不同的角度来看则可能是可解的。

最后，还有一种攻击形式是公钥体制中所特有的，这种攻击本质上就是对消息的穷举攻击。例如，假定要发送的消息是 56 位的 DES 密钥，那么攻击者可以用公钥事先对所有可能的密钥加密，并与截获的密文匹配，从而可破解任何消息。因此，无论公钥体制的密钥有多长，这种攻击都可以转化为对 56 位密钥的穷举攻击。这是一种针对公开加密算法的类似彩虹表的攻击。对抗这种攻击的方法是，在要发送的消息后附加上一个随机数。

5.2 Diffie-Hellman 密钥交换

1976 年，Whitfield Diffie 和 Martin Hellman 在《密码学的新方向》一文中提出了著名的 Diffie-Hellman 密钥交换算法，标志着公钥密码体制的出现。Diffie 和 Hellman 第一次提出了不需要使用保密信道就可以安全分发对称密钥，这就是 Diffie-Hellman 算法的重大意义所在。不仅如此，公钥加密本身就是一个重大创新，因为它从根本上改变了加密和解密的过程。

密钥交换问题是对称加密的难题之一，Diffie-Hellman 密钥交换算法可以有效地解决这个问题。这个机制的巧妙在于需要安全通信的双方可以用这个方法确定对称密钥。然后可以用这个密钥进行加密和解密。

需要注意的是，Diffie-Hellman 密钥交换算法不是一种加密算法，不能进行消息的加密和解密。只是一种能使双方共享的对称密钥不需要在网络上传递的算法。双方确定要用的密钥后，则使用该密钥用某种对称加密算法实现加密和解密消息。Diffie-Hellman 算法第一次尝试了不需要基于保密信道的密钥分发，它的目的是使两个用户在公共网络平台上安全地交换一个对称密钥，以便用于随后的报文加密。Diffie-Hellman 算法的吸引力主要在于对称密钥只在需要的时候才会进行计算，之前密钥不需要保存，所以不会有泄密的危险。其次，它也不需要 PKI 的支持，除对全局参数的约定外，密钥交换不需要事先存在的任何条件。所以目前许多商业产品还在使用这种密钥交换技术，如 SSL、IPSec 等。

Diffie-Hellman 算法的安全性建立在离散对数问题的计算困难性之上。假定给定 g 和 $x = g^k$，那么为了求解 k 需要进行通常的对数运算 $\log_g(x)$。现在给定 g、p 和 $g^k \bmod p$，求解 k 的问题与对数问题类似，不同的是进行的是离散值的计算，这个问题称为离散对数。尽管同因子分解一样未被证明是 NP 完全问题，但求解离散对数问题也是非常困难的。简而言之，可以如下定义离散对数：首先定义一个素数 p 的原根，为其各次幂产生从 $1 \sim p-1$ 的所有整数根，也就是说，如果 a 是素数 p 的一个原根，那么数值

$$a \bmod p,\ a^2 \bmod p,\ \cdots,\ a^{p-1} \bmod p$$

是各不相同的整数，并且以某种排列方式组成了 $1 \sim p-1$ 的所有整数。

对于一个整数 b 和素数 p 的一个原根 a，可以找到唯一的指数 i，使得

$$b = a^i \bmod p, \quad 0 \leqslant i \leqslant p-1$$

指数 i 称为 b 的以 a 为基数的模 p 的离散对数。该值被记为 $\text{ind}_{a,p}(b)$。

基于此背景知识，可以定义 Diffie-Hellman 密钥交换算法。该算法的实现原理如图 5.4 所示，描述如下。

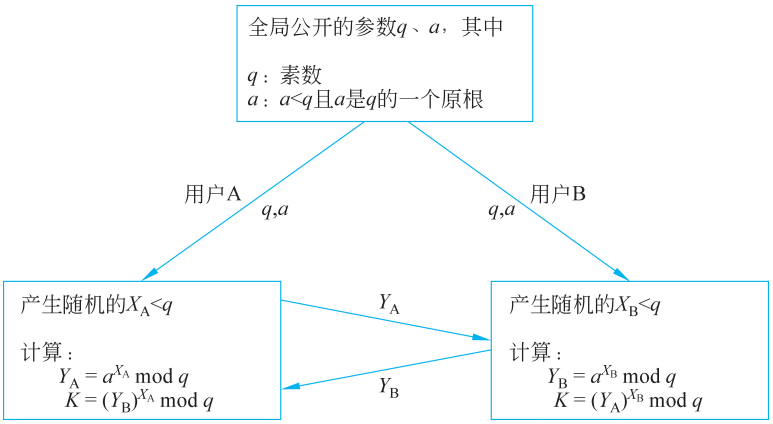

图 5.4　Diffie-Hellman 密钥交换算法的实现原理

（1）有两个全局公开的参数，一个素数 q 和一个整数 a，a 是 q 的一个原根。

（2）设用户 A 和 B 希望交换一个密钥，用户 A 选择一个作为私有密钥的随机数 $X_A < q$，并计算公开密钥 $Y_A = a^{X_A} \bmod q$。A 对 X_A 的值保密存放而使 Y_A 能被 B 公开获得。类似地，用户 B 选择一个私有的随机数 $X_B < q$，并计算公开密钥 $Y_B = a^{X_B} \bmod q$。B 对 X_B 的值保密存放而使 Y_B 能被 A 公开获得。

（3）用户 A 产生共享对称密钥的计算式是 $K = (Y_B)^{X_A} \bmod q$。同样，用户 B 产生共享对称密钥的计算式是 $K = (Y_A)^{X_B} \bmod q$。这两个计算产生相同的结果（根据取模运算规则得到）：

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (a^{X_B} \bmod q)^{X_A} \bmod q \\ &= (a^{X_B})^{X_A} \bmod q \\ &= a^{X_B X_A} \bmod q \\ &= (a^{X_A})^{X_B} \bmod q \\ &= (a^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned}$$

因此，相当于双方已经交换了一个相同的对称密钥 K。

（4）因为 X_A 和 X_B 是保密的，一个攻击方可以利用的参数只有 q、a、Y_A 和 Y_B。因此攻击方被迫求离散对数来确定用户私钥。例如，要获取用户 B 的对称密钥 K，攻击方必须先计算

$$X_B = \mathrm{ind}_{a,q}(Y_B)$$

然后他才可以像用户 B 那样计算出对称密钥 K。

下面给出的例子中，素数 $q=97$ 和它的一个原根 $a=5$，A 和 B 分别选择私钥 $X_A=36$ 和 $X_B=58$，并计算相应的公钥：

$$Y_A = 5^{36} \bmod 97 = 50 \text{（A 计算）}$$

$$Y_B = 5^{58} \bmod 97 = 44 \text{（B 计算）}$$

A 和 B 交换公钥后，双方均可单独计算出对称密钥 K：

$$K = (Y_B)^{X_A} \bmod 97 = 44^{36} \bmod 97 = 75 \text{（A 计算）}$$

$$K = (Y_A)^{X_B} \bmod 97 = 50^{58} \bmod 97 = 75 \text{（B 计算）}$$

我们假定攻击者能够得到下列信息：$q=97$，$a=5$，$Y_A=50$，$Y_B=44$。在这个简单的例子中，用穷举攻击确定 $K=75$ 是可能的。但当所有数字都足够大时，上述攻击方法实际是不可行的，即使使用大型的并行机也是如此。

然而 Diffie-Hellman 算法也存在一些不足。

（1）没有提供双方身份的任何信息，容易遭受中间人攻击。

（2）它是计算密集型的，因此容易遭受拒绝服务攻击，即攻击者请求大量的密钥，被攻击者花费了相当多的计算资源来求解无用的幂系数而不是在做真正的工作。

（3）没办法防止重放攻击。

在 Diffie-Hellman 密钥交换算法中，并没有把通信双方的身份包含进去，即它不能鉴别通信双方的身份，所以非常容易遭受中间人攻击。中间人攻击过程如图 5.5 所示。

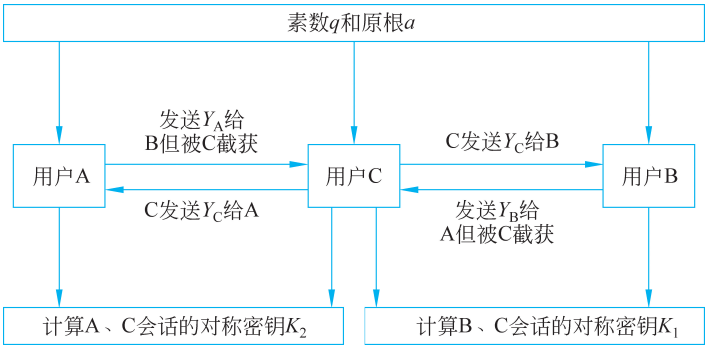

图 5.5　中间人攻击示意图

（1）B 在给 A 的报文中发送他的公钥 Y_B。

（2）C 截获 Y_B 保存下来并给 A 发送报文，该报文具有 B 的用户 ID 但使用 C 的公钥 Y_C，仍按照好像来自 B 的样子被发送出去。A 收到 C 的报文后，将 Y_C 和 B 的用户 ID 存储在一块。类似地，C 使用 Y_C 向 B 发送好像来自 A 的报文。

（3）B 基于私钥 X_B 和 Y_C 计算对称密钥 K_1。A 基于私钥 X_A 和 Y_C 计算对称密钥 K_2。C 使用私钥 X_C 和 Y_B 计算 K_1，并使用私钥 X_C 和 Y_A 计算 K_2。

（4）从现在开始，C 就可以转发 A 发给 B 的报文或转发 B 发给 A 的报文，在途中根据需

要修改。使得A和B都不知道它们是在和C直接通信,它们之间的通信都是通过C中转的。

OAKLEY算法是对Diffie-Hellman密钥交换算法的改进,它保留了后者的优点,同时克服了其弱点。OAKLEY算法具有5个重要特征。

(1) 它采用64位随机的cookie的机制来对抗拒绝服务攻击。cookie是为双方提供一种较弱的源地址认证,cookie交换可以在它执行协议中复杂的运算(大整数求乘幂)之前完成。如果源地址是伪造的,则攻击者不能得到该cookie,也就不能攻击成功。

(2) 它使得双方能够协商一个全局参数,基本上与Diffie-Hellman的全局参数一样。

(3) 它增加了"现时"机制来抗重放攻击。

(4) 它能够交换Diffie-Hellman的公开密钥。

(5) 它对Diffie-Hellman中公钥的交换进行了认证,所以能够认证交换中双方的身份以抵抗中间人攻击。

5.3 RSA

MIT的Ron Rivest、Adi Shamir和Len Adleman于1977年提出并于1978年首次发表一种用数论构造的RSA算法,可以说是最早提出的满足要求的公钥算法之一,它是迄今为止在理论上最为成熟完善的公钥密码体制,该体制已经得到广泛的应用和实践。

RSA的明文和密文均是$0\sim n-1$的整数,通常n的大小为2048位二进制数或617位十进制数,也就是说,n小于2^{2048}。本节将详细讨论RSA算法,首先给出算法描述,然后讨论RSA算法的计算问题和安全问题。

5.3.1 算法描述

1. RSA算法的密钥产生

(1) 选两个保密的大素数p和q。

(2) 计算$n=pq$,$\varphi(n)=(p-1)(q-1)$,其中$\varphi(n)$是n的欧拉函数值。

(3) 选一整数e,满足$1<e<\varphi(n)$,且$\gcd(\varphi(n),e)=1$。

(4) 计算d,满足$d\cdot e=1 \bmod \varphi(n)$,即$d$是$e$在模$\varphi(n)$下的乘法逆元,因为$e$与$\varphi(n)$互素,由模运算可知,它的乘法逆元一定存在。

(5) 以$\{e,n\}$为公钥,$\{d,n\}$为私钥。

2. RSA算法的加密

(1) 将明文比特串分组,使得每个分组对应的十进制数小于n,即分组长度小于$\log_2 n$。

(2) 对每个明文分组m,做加密运算:

$$c=m^e \bmod n$$

3. RSA 算法的解密

对密文分组的解密运算为

$$m = c^d \bmod n$$

下面将证明 RSA 算法中解密过程的正确性(相关的运算参见有关数论书籍)。

证明：从加密过程知 $c = m^e \bmod n$，所以

$$c^d \bmod n \equiv m^{ed} \bmod n \equiv m^{1 \bmod \varphi(n)} \bmod n \equiv m^{k\varphi(n)+1} \bmod n$$

以下分为两种情况。

(1) 当 m 和 n 互素时，则由 Euler 定理：

$$m^{\varphi(n)} \equiv 1 \bmod n, \quad m^{k\varphi(n)} \equiv 1 \bmod n, \quad m^{k\varphi(n)+1} \equiv m \bmod n$$

即 $c^d \bmod n \equiv m$。

(2) 当 $\gcd(m,n) \neq 1$ 时，先看 $\gcd(m,n)=1$ 的含义，由于 $n=pq$，所以 $\gcd(m,n)=1$，这意味着 m 不是 p 的倍数也不是 g 的倍数，因此 $\gcd(m,n)\neq 1$ 意味着 m 是 p 的倍数或者是 q 的倍数，假设 $m=cp$，其中 c 为一个正整数。此时必有 $\gcd(m,q)=1$，否则 m 也是 q 的倍数，从而是 pq 的倍数，与 $m<n=pq$ 矛盾。

由 $\gcd(m,n)=1$ 及 Euler 定理得 $m^{\varphi(q)} \equiv 1 \bmod q$，所以 $m^{k\varphi(q)} \equiv 1 \bmod q$，$[m^{k\varphi(q)}]^{\varphi(p)} \equiv 1 \bmod q$，$m^{k\varphi(n)} \equiv 1 \bmod q$，因此存在一整数 r，使得 $m^{k\varphi(n)} = 1 + rq$，两边同时乘以 $m=cp$，得

$$m^{k\varphi(n)+1} = m + rcpq = m + rcn$$

即

$$m^{k\varphi(n)+1} = m \bmod n$$

所以

$$c^d \bmod n = m$$

例如，选 $p=7$，$q=17$，求 $n=p\times q=119$，$\varphi(n)=(p-1)(q-1)=96$。取 $e=5$，满足 $1<e<\varphi(n)$，且 $\gcd(\varphi(n),e)=1$。确定满足 $d\times e=1 \bmod 96$ 且小于 96 的 d，因为 $77\times 5=385=4\times 96+1$，所以 d 为 77，因此公钥为{5,119}，密钥为{77,119}。设明文 $m=19$，则由加密过程得密文为

$$c = 19^5 \bmod 119 = 2\ 476\ 099 \bmod 119 = 66$$

解密为

$$66^{77} \bmod 119 = 19$$

5.3.2 RSA 算法中的计算问题

1. 加密、解密过程

RSA 的加密、解密过程都是求一个整数的整数次幂，然后取模。如果按其含义直接计算，则中间结果运算量非常大，有可能超出计算机所允许的整数取值范围。而如果采用模运算的性质：

$$(a\times b) \bmod n = [(a \bmod n)\times(b \bmod n)] \bmod n$$

就可以减小中间结果；再来考虑如何提高加、解密运算中指数运算的有效性。例如，求