第3章 chapter 3

数字签名

在生活中，经常需要在文件上签名。签名无非出于以下3种目的：

(1) 认证。如果某人拟了一份文件，希望其他人确信该文件来自他，他可以在文件上签名。

(2) 批准和负责。例如办理某些银行业务(如刷卡消费、支取存款)时，营业员会要求经办人签名，这是为了防止经办人以后抵赖，因为一旦签名就表明该项业务得到了经办人的批准，并且由经办人承担责任。

(3) 有效。人们有时请求领导或上级对某份文件签名或签章，用来表明该份文件是有效力和权威的，以获得机构内其他人的认可。

3.1 数字签名概述

对签名的基本要求是：①无法伪造；②容易认证；③不可抵赖。手写签名一般通过某人特有的笔迹实现以上要求。例如，有些领导为了防止别人伪造签名，也为了让其他人容易鉴别，一般把签名设计得很有特色，其他人模仿不出。而数字签名和手写签名的功能非常类似，好的数字签名比手写签名更能够防止伪造。因此，我国制定了《电子签名法》，承认数字签名和手写签名具有同等的法律效力。

不仅如此，通过数字签名还能实现认证机制。如果一份消息附带某人的数字签名，那么可以确信该消息确实是从该用户处发出的，而不是其他人伪造的。因此，可以说数字签名是连接加密技术和认证技术的桥梁。

3.1.1 数字签名的特点

传统签名有4个基本特点：①与被签的文件在物理上不可分割；②签名者不能否认自己的签名；③签名不能被伪造；④签名容易被验证。

而数字签名是传统签名的数字化，它也具有传统签名的这4个特点：①签名能与所签文件绑定；②签名者不能否认自己的签名；③签名不能被伪造；④签名容易被自动验证。

而进行数字签名通常也是为了确认以下两点：

第一，信息是由签名者发送的。

第二，信息自签发后到收到为止未经任何修改。

总体来说，数字签名应具备以下几个特点：

(1) 签名是可以被确认的，即接收方可以确认或证实签名确实是由发送方完成的。

(2) 签名是不可伪造的，即接收方和第三方都不能伪造签名。

(3) 签名不可重用，即签名是和消息绑定的，不能把签名移到其他消息(文件)上。

(4) 签名是不可抵赖的，即发送方不能否认他签发的消息。

(5) 第三方可以确认收发双方之间的消息传送，但不能篡改消息。

3.1.2 数字签名的过程

要实现数字签名，最简单的方法是：发送方将整个消息用自己的私钥加密；接收方用发送方的公钥解密，解密成功就可验证该消息经过发送方的签名。

但这种方法有一个缺陷，就是被签名的消息可能很长，由于公钥加密的运算速度慢，导致加密会非常耗时而不可行。因此，在实际应用中，数字签名是先对消息用单向散列函数求消息摘要(散列值)，然后发送方用其私钥加密该散列值，这个被发送方用私钥加密的散列值就是发送方的**数字签名**。发送方将其附在消息后，一起发送给接收方，就可以让其验证签名了。

验证签名时，接收方先用发送方的公钥解密数字签名，然后将提取的散列值与自己计算的散列值比较，如果相同，就表明该签名是有效的，整个过程如图 3.1 所示。这样，攻击者虽然能截获并阅读消息(消息是明文形式)，但不能修改消息内容或将消息换成别的消息，因为别的消息的散列值和该消息的散列值是不同的，接收方能通过验证签名发现。

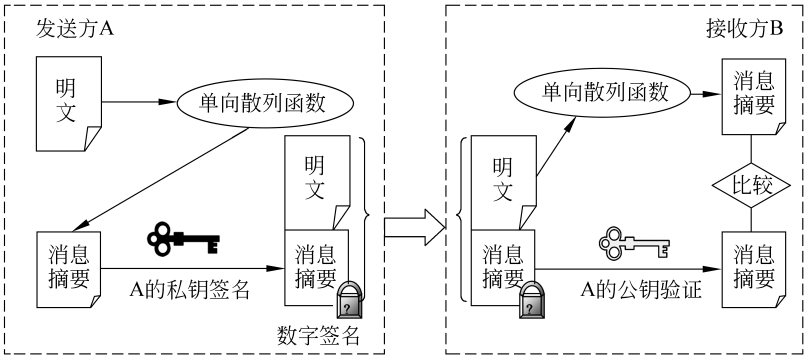

图 3.1 数字签名的过程

图 3.1 的数字签名方案虽然解决了公钥密码体制加密长消息速度慢的问题，但又产生了一个新的问题，那就是消息将以明文形式传输，无法实现机密性。如果对消息有机密性需求，则可以将明文和数字签名的组合体先用一个对称密钥加密，再将加密后的组合体以及对称密钥的数字信封发送给接收方，如图 3.2 所示(省略了数字签名过程)。这种方式将数字签名与数字信封技术结合在一起，实现了能满足消息机密性需求的数字签名。

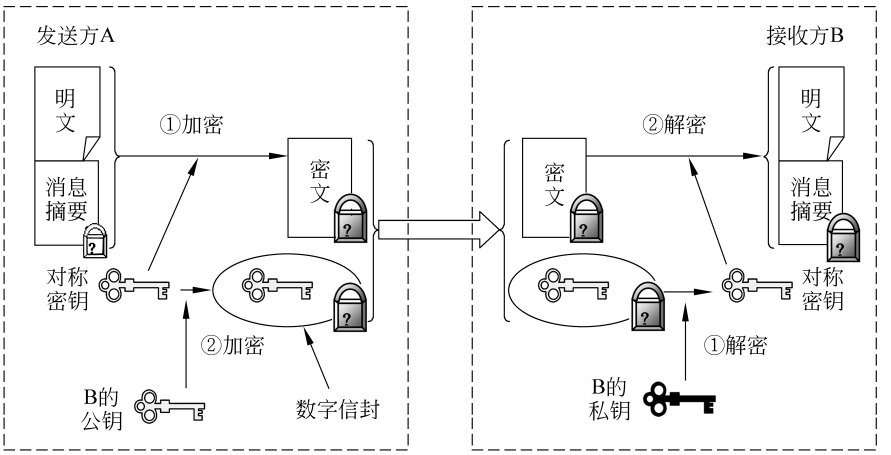

图 3.2　能满足机密性需求的数字签名方案

提示：能满足机密性需求的数字签名使用了两对公私钥。这是因为，公钥密码体制如果用于数字签名，则无法同时实现机密性；反之，如果用于加密，则无法同时实现数字签名。如果要用公钥密码体制同时实现数字签名和加密，则需要使用两次公钥密码算法，一次用于加密，另一次用于数字签名。这需要两对公私钥才能实现，一对是发送方的，另一对是接收方的。

3.2　数字签名的算法实现

理论上，只要是双向可逆的公钥加密算法都可用于数字签名，常见的数字签名算法有 RSA、ElGamal、Schnorr 等。下面分别介绍这 3 种数字签名算法。

3.2.1　RSA 数字签名算法

设 RSA 数字签名算法的私钥为 d，公钥为 e，则 RSA 数字签名算法的思想就是：签名者用自己的私钥 d 加密文件摘要，其他人用签名者的公钥 e 就可以验证签名。

1. 签名过程

用户 A 对消息 m 进行签名，他先计算 m 的摘要 $H(m)$，再用自己的私钥 d 计算 S_A：

$$S_A = \mathrm{Sign}(H(m)) = (H(m))^d \bmod n$$

然后将 S_A 附在消息 m 后作为用户 A 对消息 m 的签名。

2. 验证签名过程

如果其他用户要验证 A 对消息 m 的签名，则要用 A 的公钥 e 计算散列值：

$$M' = S_A^e \bmod n$$

如果 M' 与 $H(M)$ 相等，则相信签名确实是用户 A 的。可见，RSA 数字签名算法的计算过程就是 RSA 加密算法的逆过程。

3. RSA 数字签名的注意事项

如果用 RSA 数字签名算法实现数字签名，一定要先求出消息摘要，再用私钥签名；或者将一对公私钥专门用于签名，而用另一对公私钥对进行加解密。

这是因为，公钥 e 和 n 是公开的，攻击者如果截获别人发给接收方的密文 c（c 是别人用接收方的公钥 e 加密得到的，即 $c=m^e \bmod n$），则攻击者可以任意选择一个小于 n 且与 n 互素的数 r，计算

$$x=r^e \bmod n, \quad y=xc \bmod n$$

将 y 发给接收方请求签名。如果接收方随随便便就用自己的私钥 d 给攻击者发来的 y 签名，即

$$u=y^d \bmod n$$

则攻击者得到签名 u 后，就可以轻而易举地恢复出 c 对应的明文 m。他首先计算 r 的乘法逆元 t，即 $t=r^{-1} \bmod n$，再把 t 和 u 相乘即得到 m，这是因为

$$\begin{aligned} t \times u &= r^{-1}y^d \bmod n \\ &= r^{-1}(xc)^d \bmod n = r^{-1}x^d c^d \bmod n \\ &= r^{-1}r^{ed}c^d \bmod n = r^{-1}r^{k\phi(n)+1}c^d \bmod n \\ &= r^{-1}rc^d \bmod n = c^d \bmod n = m \end{aligned}$$

而如果先对消息 y 求消息摘要 $H(y)$ 再签名则不存在该问题，或者签名和加密使用不同的密钥对也能避免该问题。

3.2.2 ElGamal 数字签名算法

ElGamal 数字签名算法是一种非确定性的签名方案，它需要使用随机数，但 ElGamal 数字签名算法的运算过程并不是 ElGamal 加密算法的逆过程。

1. 选择密钥

系统先选取一个大素数 p 及 p 的本原根 a，用户 A 选择一个随机数 $x(1 \leqslant x \leqslant p-1)$ 作为自己的私钥，计算 $y=a^x \bmod p$，将 y 作为自己的公钥。整个系统公开的参数有大素数 p、本原根 a 以及每个用户的公钥，而每个用户的私钥 x 则严格保密。

2. 签名过程

给定消息 m，用户 A 通过下述计算实现签名。

(1) 选择随机数 $k \in \mathbf{Z}_p^*$，且 k 与 $p-1$ 互素（注意，随机数 k 需要保密）。

(2) A 对消息 m 进行散列压缩后得到消息散列值 $H(m)$，再计算

$$r=a^k \bmod p$$

$$s=(H(m)-xr)k^{-1} \bmod (p-1)$$

将 (r,s) 作为用户 A 对消息 m 的数字签名，与消息 m 一起发送给接收方。

3. 验证签名的过程

接收方 B 在收到消息 m 与数字签名 (r,s) 后，先计算消息 m 的散列值 $H(m)$。然后计算

$$y^r r^s \bmod p = a^{H(m)} \bmod p$$

如果上式成立，则可确信(r,s)为有效签名；否则认为签名是伪造的。

4. 证明验证签名的正确性

若(r,s)为合法用户采用ElGamal数字签名算法对消息m的签名，则

$$y^r r^s=(a^x)^r(a^k)^s=a^{xr+ks} \bmod p$$

又因为

$$s=(H(m)-xr)k^{-1} \bmod(p-1)$$

两边乘k再移项得

$$ks+xr=H(m) \bmod(p-1)$$

根据模运算规则有

$$a^{xr+ks}=a^{H(m) \bmod(p-1)} \bmod p$$

由费马定理的推论，$a^k \equiv a^{k \bmod(p-1)} \bmod p$，将$k$替换成$H(m)$，有

$$a^{xr+ks}=a^{H(m)} \bmod p$$

因此有

$$y^r r^s=a^{H(m)} \bmod p$$

5. ElGamal数字签名算法举例

用户A对消息m进行签名。设系统选取素数$p=19$，本原根$a=13$。用户A选择整数$x=10$作为自己的私钥，经计算可得用户A的公钥$y=6$。

如果用户A需要对消息m的散列值$H(m)=15$进行签名，首先选择一个随机数$k=11$，然后求出k的乘法逆元：

$$k^{-1}=5 \bmod 19$$

然后计算r：

$$r=a^k \bmod p=13^{11} \bmod 19=2$$

接着计算s：

$$s=(H(m)-xr)k^{-1} \bmod(p-1)=5\times(15-10\times2)\bmod 18=11$$

用户A把元组$(r,s)=(2,11)$作为自己对$H(m)=15$的数字签名。

接收方B验证签名时只须计算并验证

$$y^r r^s \bmod p=6^2\times2^{11} \bmod 19=8$$

$$a^{H(m)} \bmod p=13^{15} \bmod 19=8$$

两者相等，则认为$(2,11)$是用户A对消息m的有效签名。

6. ElGamal数字签名算法的安全性

关于ElGamal数字签名算法的安全性，有以下几点需要注意：

(1) ElGamal数字签名算法是一个非确定性的算法，对同一个消息m所产生的签名依赖于随机数k。

(2) 由于用户的签名私钥x是保密的，攻击者要从公钥y推导出私钥x等价于求解离散对数的困难性，因此ElGamal数字签名算法的安全性是建立在求解离散对数的困难性上的。

(3) 在签名时使用的随机数k绝对不能被泄露。这是因为，当攻击者知道了随机数

k 后，就可以通过公式

$$s=(H(m)-xr)k^{-1} \bmod (p-1)$$

推出

$$x=(H(m)-ks)r^{-1} \bmod (p-1)$$

从而得到用户的私钥 x，这样整个签名算法便被攻破。

(4) 随机数 k 不能被重用。有研究指出，如果随机数 k 被重用，则攻击者可根据得到的两个不同的签名求出签名私钥 x。

ElGamal 数字签名算法还有一些变种，如 DSA。它是一种单向不可逆的公钥密码体制，只能用于数字签名，而不能用于加解密和密钥分配。与 ElGamal 类似，DSA 在每次签名的时候也要使用随机数，所以对同一个消息，每次签名的结果是不同的，所以称 DSA 为随机化数字签名算法。而 RSA 为确定性数字签名算法。由于 RSA 存在共模攻击，用 RSA 签名时每次都要使用不同的 n，而 DSA 没有这个要求，因此在实际中使用 DSA 进行数字签名比 RSA 更加方便。

3.2.3 Schnorr 数字签名算法

Schnorr 数字签名算法的安全性建立在求解离散对数的困难性上。对于相同的安全级，Schnorr 的签名长度比 RSA 短(对 140 位长的 q，Schnorr 签名长度仅为 212 位，比 RSA 签名长度短一半，比 ElGamal 签名短得多)。而且产生签名所需的大部分计算都可在预处理阶段完成，进一步提高了该签名体制的速度。由于其签名运算的高效率，Schnorr 数字签名算法已被广泛应用于许多电子现金协议和公平盲签名协议中。

Schnorr 数字签名算法的签名过程如图 3.3 所示。

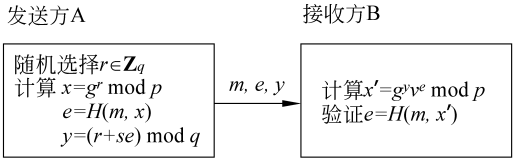

图 3.3 Schnorr 数字签名算法的签名过程

1. 初始过程

初始时算法完成以下工作：

(1) 选择大素数 p、q，$p\geqslant 2^{512}$，$q\geqslant 2^{160}$，并且 q 是 $p-1$ 的一个素因子，即 $q\mid(p-1)$。

(2) 选择 $g\in\mathbf{Z}_p{}^*$，满足 $g^q\equiv 1 \bmod p$。

(3) 选择一个小于 q 的随机数 s，计算 $v=g^{-s} \bmod p$。

(4) 将 p、q、a、v 公开，将 s 保密，其中 v 是公钥，s 是私钥。

2. 签名过程

签名过程如下：

(1) 签名方 A 选取一个小于 q 的随机整数 r，并计算 $x=g^r \bmod p$。

(2) A 将消息 m 与 x 连接起来，计算其散列值 $e=H(m,x)$。

（3）A计算 $y=(r+se) \bmod q$，(e,y)即为签名。A将消息 m 和签名(e,y)传送给B。

其中，$H(\cdot)$是一个单向散列函数，m 是消息。

3. 验证过程

验证方B收到 m 和(e,y)后，计算 $x'=g^y v^e \bmod p$，然后验证 $e=H(m,x')$，如果通过验证，则认为该签名有效。这是因为 $y=(r+se) \bmod q$，$v=g^{-s} \bmod p$，$x=g^r \bmod p$。所以

$$x'=g^y v^e \bmod p=g^{(r+se)} v^e \bmod p=g^r g^{se} g^{-se} \bmod p=g^r \bmod p=x$$

由于每次计算得到的签名与选择的随机数 r 有关，因此Schnorr数字签名算法也是非确定性算法。

3.3 前向安全数字签名

对于数字签名来说，其安全性涉及两方面：一是签名算法的安全性，数字签名使用的算法要能够抵抗各种密码分析，即算法不被破解；二是签名私钥的安全性，即私钥不会被窃取，或者即使被窃取了损失也不大。一般来说，私钥由签名者自己生成后保存在自己的系统中，不会经过网络传输，一般很难被窃取，因此普通数字签名算法都假设私钥是绝对安全的（这个假设隐含着：如果私钥泄露，则责任完全由签名者承担，验证者不需承担任何责任）。如果不这样假设，则签名者无论自己的私钥是否被盗，他都可以声称自己的私钥被盗了，是窃取者用他的私钥签的名，从而可对自己签名的任何消息进行抵赖。

但是私钥不在网络上传输并不表示私钥是绝对安全的，因为攻击者还可能会攻入签名者的系统并窃取私钥。一旦签名私钥被泄露，则攻击者可使用该私钥随意冒用签名，这将给整个系统带来灾难性的后果。银行或CA等比较重要的机构必须考虑签名私钥泄露这种风险的存在。

1. 前向安全的概念和方法

基于这样的背景，1997年，Anderson首次提出了前向安全（forward secrecy）的概念。其主要思想是：将一个密码学系统的整个生命周期分为若干时间段，系统的私钥值在每个时间段都不断地变化。这样，即使当前时间段的私钥值泄露了，也不会影响以前时间段私钥的安全性，这意味着以前的签名仍然是有效的。因此，前向安全数字签名方案能有效地降低因私钥泄露而造成的损失。这种思想的本质是对数字签名安全性的风险控制，即将签名私钥泄露后造成的损失尽可能减少。

前向安全数字签名与一般数字签名相比多了一个私钥自动更新的环节。这使它具有前向安全性：如果时间段 i 的私钥泄露，则攻击者只可以伪造时间段 i 以后的签名，而不能伪造时间段 i 之前的签名，也就是说，时间段 i 之前的签名仍然有效。

实现前向安全数字签名的关键是私钥可以自动更新，但验证签名的公钥却要求始终不变，这样，无论私钥怎样变化，验证者总能用固定的公钥和时间段编号对签名进行验证。因此，私钥可以用单向散列函数（例如散列链）实现，即允许签名者由昨天的私钥计

算出今天的私钥，但不能由今天的私钥计算出昨天的私钥，以此保证：即使当前的私钥暴露了，过去的私钥仍然是安全的。自动更新是单向的，所以自动更新函数是单向散列函数，为了便于验证及提高效率，对应的公钥必须始终保持不变。

为了实现前向安全，可以将签名的私钥按时间段进行自动更新，并用不同的私钥生成签名，而相应的公钥并没有变，任何验证者都可以使用固定的公钥和时间段编号验证签名。前向安全数字签名的私钥自动更新过程如图 3.4 所示。用户先注册一个公钥 PK，同时保存相应的私钥 SK。然后将公钥的有效时间分为 n 个时间段，记为 $T_1, T_2, \cdots, T_n$，每个时间段的私钥记为 $SK_1, SK_2, \cdots, SK_n$。存在这样的单向散列函数 f，它可以将私钥 SK_1 更新为 SK_2，即 $SK_2 = f(SK_1)$。因为单向散列函数具有单向性，即，由 SK_1 计算 SK_2 非常容易，而反过来由 SK_2 计算 SK_1 则非常困难。

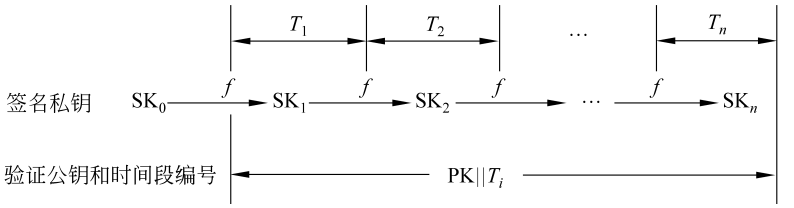

图 3.4 前向安全数字签名的私钥自动更新过程

2. 基于 ElGamal 的前向安全数字签名方案

通常一个前向安全数字签名方案包括 4 个算法：公私钥生成算法、私钥更新算法、数字签名算法和数字签名验证算法，也就是说比普通数字签名方案多了一个私钥更新算法。

基于 ElGamal 的前向安全数字签名方案的 4 个算法说明如下。

(1) 公钥生成算法。

系统先选取一个大素数 p 及 p 的本原根 g，用户 A 选择一个随机数 $x(1 \leqslant x \leqslant p-1)$ 作为自己的初始私钥 SK_0，确定私钥的更新次数 T_i，并根据 $PK = g^{SK_0^{-1}} \bmod p$ 计算出公钥 PK。系统公开初始化参数$\{p, g, PK, T_i\}$。

(2) 私钥更新算法。

前向安全技术的关键是根据设定的时间段不断地计算出新的私钥，并用新的私钥替换旧的私钥。设 i 表示时间段 i，则私钥更新算法如下：

根据 SK_i 计算 SK_{i+1}，计算公式为 $SK_{i+1} = SK_i^2 \bmod (p-1)$，其中 $i \in [1, n+1]$。

显然，如果想由 SK_{i+1} 计算 SK_i，则等价于求解离散对数的难题，因此非常困难。

(3) 数字签名算法。

对消息 m 进行签名时，签名者首先选择一个随机数 k（k 与 $p-1$ 互素），然后计算

$$a = g^k \bmod p$$

$$b = (H(m)) - SK_i^{2a+1-i} a)k^{-1} \bmod p$$

此时对消息 m 的签名结束，$\{a, b, i\}$时间段 i 对消息 m 的签名。

（4）数字签名验证算法。

验证者在接收到签名后通过以下等式进行验证：

$$PK^a a^b = g^{H(m)} \bmod p$$

若上式为真，则签名有效；反之则签名无效。

3. 强前向安全的概念

前向安全数字签名仍然是有安全漏洞的，因为它没有办法阻止攻击者在窃取私钥后在未来的时间段内进行同样的私钥更新。即，如果攻击者获得了时间段 i 的私钥，并且签名方也没有发觉自己的私钥已经被窃取，那么攻击者就可以和签名方一样进行私钥的更新，得到时间段 i 以后的所有私钥。有了这些私钥就可以伪造时间段 i 及以后的所有签名，直到被签名者发现。也就是说，前向安全签名无法保证签名在未来的安全性（即后向安全性）。为此，2001 年，Burmester 提出了强前向安全数字签名的概念，即在保证签名是前向安全的同时，不应该让攻击者具有和合法签名者同样的私钥更新能力，也就是说，即使攻击者获得了时间段 i 的私钥，它也不能伪造时间段 i 以前的签名和时间段 i 以后的签名，这样的安全性称为强前向安全性，或称为双向安全性。

3.4 特殊的数字签名

在电子商务、电子投票、电子现金等领域，产生了几种特殊的数字签名方式，如盲签名、群签名、群盲签名、门限签名、数字时间戳等。


盲签名

3.4.1 盲签名

1. 盲签名的特点

在一般的数字签名中，文件的签名者都知道他们签署的文件内容，甚至该文件就是签名者自己撰写的。但有时可能需要某人对一个文件签名，却又不想让他知道文件的内容。例如，立遗嘱时，通常将遗嘱写好并用信封密封好后，由公证人签名盖章，为了防止公证人未到时候就私下将遗嘱的内容泄露出去，要求公证人看不到遗嘱内容，但又必须让公证人签名，这样验证者才能确信遗嘱是真实的。这里公证人对遗嘱的签名就是一种盲签名。

盲签名最主要的用途是实现电子现金的匿名性。用户自己生成了一些电子现金（包含序列号），把电子现金提交给银行签名（当然有办法让银行能大体知道它签署的是什么，只不过不准确而已），这样电子现金才会变得有效，但用户又不想让银行知道自己提交的电子现金是哪些，以防止银行对用户的消费状况进行跟踪，侵犯用户隐私。因此，不能让银行看到待签名文件（电子现金）的具体内容（如序列号），这就需要采用盲签名技术。

盲签名操作涉及三方，分别是请求签名者、签名者和签名验证者。

为了实现盲签名，一种自然的想法就是先将消息加密（称为盲化），再把加密的消息发送给签名者。这样，签名者就无法阅读消息的内容了，而只能进行签名。请求签名者可先将签名解密（脱盲），然后把消息明文和脱盲的签名发送给验证者进行验证。该过程

如图 3.5 所示。

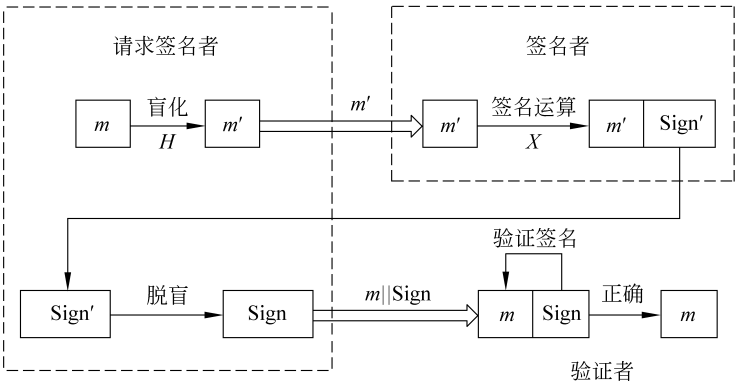

图 3.5 盲签名的过程

提示：脱盲的签名就相当于签名者直接对消息明文 m 进行的签名。

由此可见，盲签名的基本原理是两个可交换的算法的应用：第一个是加密算法，用来隐藏消息，实现盲化处理；第二个是签名算法，用来对消息进行签名。只有这两个算法是可交换的，即 $\mathrm{Sign}_k(mh)=(\mathrm{Sign}_k(m))h$，其中 h 为盲化因子，盲签名才能有效。

如果这两个算法不能交换，则文件拥有者（即请求签名者）无法进行脱盲运算，不能由 Sign′得到 Sign，而只能解密 Sign′得到 m'。虽然文件拥有者也可以把 m、m' 和 Sign′提交给验证者验证 Sign′确实是从 m 得来的签名，但这又要将盲化因子 h 告诉验证者，而一旦盲化因子公开，则签名者也能用盲化因子解密得知明文了。

综上，盲签名与普通数字签名相比，有两个显著的特点，即两个条件：

（1）消息的内容对签名者是不可见的。例如，图 3.5 中签名者不知道 m。

（2）在签名消息被接收者公开后，签名者不能追踪签名，即盲签名具有不可追踪性。例如，图 3.5 中签名者即使看到 Sign，仍然不能把它和 m' 联系起来。

上述盲签名方案又称为强盲签名。如果盲签名方案满足条件（1），但不满足条件（2），即，在签名消息被接收者公开后，签名者能够追踪签名，则称为弱盲签名或公平盲签名。

盲签名可通过 RSA 算法和离散对数等数学难题实现。

2. RSA 的盲签名体制

RSA 盲签名的步骤如下：

（1）参数选择。系统随机选取两个大素数 p 和 q，计算 $n=pq$，再计算 n 的欧拉函数 $\phi(n)=(p-1)(q-1)$，计算完后，n 可以公开。然后选择一个与 $\phi(n)$ 互素的整数 e 作为某用户的公钥（这样 e 才会有乘法逆元）。求出 e 的乘法逆元，将该结果作为该用户的私钥 d，即 $de=1 \bmod \phi(n)$。将 d 保密，将 (d,n) 作为私钥；将 e 公开，将 (e,n) 作为公钥。p、q 和 $\phi(n)$ 都需要保密。

（2）签名过程。用户（请求签名者）选择待签名的消息 $m\in\mathbf{Z}_n^*$ 和一个随机数 $r\in\mathbf{Z}_n$ 作为盲化因子，并用签名方的公钥 e 对原消息进行盲化，即计算

$$m'=mr^e \bmod n$$

然后把盲化的消息 m' 发送给签名者进行签名。

签名者收到 m' 后，用自己的私钥 d 对其进行签名，即计算

$$\mathrm{Sign}(m')=(m')^d \bmod n$$

可见，签名过程和普通 RSA 签名完全一致，然后把 $\mathrm{Sign}(m')$ 作为 m' 的签名发送给用户。

(3) 脱盲过程。验证者收到 $\mathrm{Sign}(m')$ 后，对其进行脱盲运算，即计算

$$\mathrm{Sign}(m)=\mathrm{Sign}(m')/r \bmod n$$

$\mathrm{Sign}(m)$ 就是对原消息 m 的直接签名，即 $\mathrm{Sign}(m)=m^d \bmod n$。这是因为

$$\mathrm{Sign}(m)=\mathrm{Sign}(m')/r=(m')^d/r=(mr^e)^d/r=m^d r^{ed}/r \bmod n=m^d r/r \bmod n=m^d \bmod n$$

(4) 验证签名。由于 $\mathrm{Sign}(m)$ 就是对原消息 m 的直接签名，因此验证者可以用签名者的公钥 e 像验证普通 RSA 签名一样验证 $\mathrm{Sign}(m)$，即验证如下等式是否成立：

$$m=(\mathrm{Sign}(m))^e \bmod n$$

例 3.1 取 $p=3, q=11$，则 $n=33, \phi(n)=20$。再取公钥 $e=3$，经计算得 $d=7$。设明文 $m=6$，任取随机数 $r=5$。求 m 的盲签名，并对盲签名进行验证。

解：

$$m'=6\times 5^3 \bmod 33=750 \bmod 33=24$$

$$\mathrm{Sign}(m')=24^7 \bmod 33=18$$

$$\mathrm{Sign}(m)=18\times 5^{-1} \bmod 33$$

$$\mathrm{Sign}(m)=30$$

验证如下：

$$m=6, (\mathrm{Sign}(m))^e \bmod n=30^3 \bmod 33=6$$

两者相等，说明签名是有效的。

3. ElGamal 的盲签名体制

ElGamal 盲签名的步骤如下：

(1) 系统先选取一个大素数 p 及 p 的本原根 a，然后选择一个随机数 x，$2\leqslant x\leqslant p-2$，再计算 $y=a^x \bmod p$，以 (y,a,p) 作为用户的公钥，而以 x 作为用户的私钥。

(2) 盲化过程。请求签名者选择随机数 $h\in \mathbf{Z}_p^*$ 作为盲化因子，然后计算

$$\beta=a^h \bmod p$$

$$m'=mh \bmod (p-1)$$

将二元组 (β, m') 发送给签名者。

(3) 签名过程。签名者收到 (β, m') 后，选择随机数 $k\in \mathbf{Z}_{p-1}^*$，并用自己的私钥 x 对 m' 进行签名，即计算

$$r=\beta k \bmod p$$

$$s=xr+m'k \bmod (p-1)$$

并将 (r,s) 作为对消息 m 的签名发送给验证者。

(4) 验证过程。验证者收到 (r,s) 后，用签名者的公钥 y 进行验证

$$a^s=r^m y^r \bmod p$$

如果该式成立，则说明签名有效。

3.4.2 群签名、群盲签名和门限签名

1. 群签名

群签名是指：一个群中的任意一个成员可以以匿名的方式代表整个群对消息进行签名，验证者可以确认签名来自该群，但不能确认是群中的哪一个成员进行的签名。但是当出现争议时，借助于一个可信的机构或群成员的联合就能识别出群中的签名者。

与其他数字签名一样，群签名也是可以公开验证的，而且可以用单个的群公钥来验证。

一个群签名体制由以下几个算法和协议组成：

(1) 创建算法。一个用以产生群公钥和私钥的多项式时间概率算法。

(2) 加入协议。一个用户和群管理员之间的交互协议。执行该协议可以使用户成为群成员，群管理员得到群成员的秘密成员管理密钥，并产生群成员的私钥和群成员证书。

(3) 签名算法。一个概率算法，当输入一个消息和一个群成员的私钥后，输出对消息的签名。

(4) 验证算法。一个在输入对消息的签名及群公钥后确定签名是否有效的算法。

(5) 打开算法。一个在给定一个签名及群公钥的条件下确定签名人身份的算法。

一个好的群签名方案应满足以下的安全性要求：

(1) 匿名性。给定一个群签名后，对除了唯一的群管理员之外的任何人来说，确定签名人的身份在计算上是不可行的。

(2) 不关联性。在不打开群签名的情况下，确定两个不同的群签名是否为同一个群成员所签在计算上是困难的。

(3) 防伪造性。只有群成员才能产生有效的群签名。

(4) 可跟踪性。群管理员在必要时可以打开一个群签名，以确定签名人的身份，而且签名人不能阻止一个合法群签名的打开。

(5) 防陷害攻击。包括群管理员在内的任何人都不能以其他群成员的名义产生合法的群签名。

(6) 抗联合攻击。即使一些群成员串通在一起也不能产生一个合法的不能被跟踪的群签名。

2. 群盲签名

1998 年，Lysyanskaya 和 Ramzan 有效结合群签名和盲签名提出了群盲签名的概念。大多数电子现金系统都基于由单个银行发行电子现金的模型，所有的用户与商家在同一家银行拥有账户。而在现实世界中，电子现金可能是在一个中央银行监控下由一群银行发行的。Camenisch 和 Stadler 利用群盲签名构造了一个多个银行参与发行电子现金的、匿名在线的电子现金方案，为研究电子现金系统开辟了一个新的方向。

在该方案中有多个银行参与，每个银行都可以安全地发行电子现金，这些银行形成一个群，受中央银行的控制，中央银行担当群管理员的角色。该方案具有以下性质：

(1) 任何银行都不能跟踪自己发行的电子现金。

(2) 商家只需要用单个群公钥验证其收到的电子现金的有效性，而不关心该电子现

金是哪个银行发行的。

(3) 所有银行组成的群只有一个公钥，该公钥与参与银行的个数无关，而且有银行加入时，该公钥也不需要改变。

(4) 给定一个合法的电子现金，除中央银行以外的任何银行都不能辨别该电子现金是哪个银行发行的，为用户和银行提供了匿名性。

(5) 包括中央银行在内的任何银行都不能以其他银行的名义发行电子现金。

3. 门限签名

在有 n 个成员的群中，至少有 t 个成员才能代表群对文件进行有效的数字签名。例如，银行金库大门的打开申请需要一个正行长和一个副行长同时签名或者 3 个副行长同时签名才能生效。这就需要门限签名。门限签名可通过共享密钥的方法实现，它将密钥分为 n 份，只有超过 t 份子密钥组合在一起才能重构出密钥。

3.4.3 数字时间戳

在某些电子交易中，交易时间是非常重要的信息。例如，股票、期货的交易时间直接影响交易商品的价格。因此，需要一个可信任的第三方——时间戳权威(Time Stamp Authority，TSA)提供可信赖的且不可抵赖的时间戳服务。TSA 的主要功能是证明某份文件(交易信息)在某个时间(或以前)存在，防止用户在这个时间后伪造数据进行欺诈。

数字时间戳(Digital Time-Stamp，DTS)产生的一般过程是：用户首先对需要加时间戳的文件用散列函数计算其摘要，然后将摘要发送给 TSA。TSA 将收到文件摘要时的日期和时间信息附加到文件中，再用 TSA 的私钥对该文件进行加密(TSA 的数字签名)，然后将其返回给用户，整个过程如图 3.6 所示。

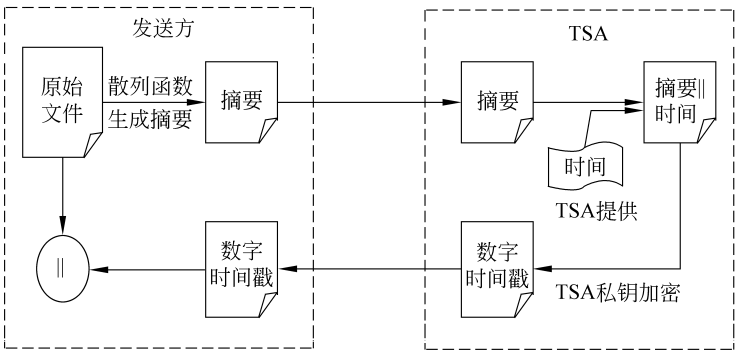

图 3.6 数字时间戳的产生过程

用户收到数字时间戳后，可以将其与原始文件一起发送给接收方，供接收方验证时间。

可见，数字时间戳是一个经 TSA 签名后形成的凭证文档，它包括 3 部分：

(1) 需加时间戳的文件摘要。

(2) TSA 收到文件的日期和时间。

(3) TSA 的数字签名。

习 题

1. 要实现电子现金的匿名性，可以使用(　　)。

A. 盲签名　　B. 群签名

C. 门限签名　　D. 前向安全数字签名

2. 盲签名操作涉及的三方分别是_____、签名者和验证者。

3. 为了解决签名私钥可能被窃取的问题，人们提出了_____数字签名。

4. 如果要实现带有机密性的数字签名，需要使用发送方的_____和接收方的_____。(填公钥或私钥)

5. 简述数字签名的过程和应用。

6. 小强想出了一种数字签名的新方案，他用一个随机的对称密钥加密要签名的明文，得到密文，再用自己的私钥加密该对称密钥(签名)，然后把密文和加密后的对称密钥一起发送给接收方，接收方如果能解密得到明文，就表明验证签名成功。请问用该方案能够对明文签名吗？为什么？

第4章 chapter 4

密钥管理与密钥分配

在现代密码学中，加密算法的安全性完全依赖于密钥，因此密钥是现代密码学的核心，密钥管理是整个密码系统中最重要的环节。密钥管理作为现代密码学的一个重要分支，是在授权各方之间建立和维护密钥关系的一整套技术，它是现代密码学中最重要、最困难的部分。

4.1 密钥管理

密钥管理是一种综合性技术，它除了技术因素外，还包括管理因素，例如密钥的行政管理制度和人员的素质密切相关。再好的技术，如果失去必要的管理支持，也将毫无意义。密码系统的安全强度总是由系统中最薄弱的环节决定的。但是，作为一个好的密钥管理系统，应尽量不依赖于人的因素，为此，密钥管理系统的一般应满足以下要求：

(1) 密钥难以被窃取。

(2) 在一定条件下，即使窃取了密钥也没有用。

(3) 密钥的分配和更换过程在用户看来是透明的，用户不一定要亲自掌握密钥。

4.1.1 密钥的层次结构

如果一个密码系统的功能很简单，可以使用单层密钥体制，即所有的密钥都直接用来加密或解密数据，但这种密钥体制的安全性不高。一个完善的密码系统通常要求密钥能够定期更换、自动生成和分配等各种附加功能，为此，就需要设计成多层密钥体制。

多层密钥体制的基本思想是用密钥保护密钥。在多层密钥体制中，密钥可分为会话密钥、密钥加密密钥、主密钥3个层次。密钥的层次结构如图4.1所示。其中，f_n 表示加解密变换函数。

- 会话密钥(session key)是最底层的密钥，直接对数据进行加密和解密。
- 密钥加密密钥(key encrypting key)是最底层以上的所有密钥(除主密钥外)，用于对下一层密钥进行加密保护。
- 主密钥(master key)是最高层的密钥，是密钥系统的核心，通常受到严格的保护，用于对密钥加密密钥进行保护。

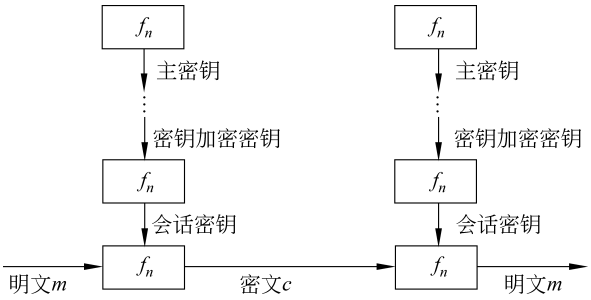

图 4.1 密钥的层次结构

多层密钥体制的优点如下：

(1) 安全性大大提高，下层的密钥被破解不会影响上层密钥的安全。

(2) 为密钥管理自动化带来了方便。除主密钥需由人工装入以外，其他各层密钥均可由密钥管理系统实行动态的自动更新和维护。

4.1.2 密钥的生命周期

密钥管理涉及密钥的生成、存储、更新、备份与恢复、销毁以及撤销等，涵盖了密钥的整个生命周期。

1. 密钥的产生

密钥必须在安全环境中生成，以防止对密钥的非授权访问。密钥的生成有两种方式：一种是由密钥分配中心集中生成，称为集中式；另一种是由客户端分散生成，称为分散式。这两种方式各有优缺点，如表 4.1 所示。

表 4.1 两种密钥生成方式的对比

方　式	集 中 式	分 散 式
代表	密钥分配中心	个人客户端
生产者	在中心统一进行	用户
用户数量	用户数量受限制	用户数量不受限制
特点	密钥质量高，方便备份	需第三方认证
安全性	需安全的私钥传输通道	安全性高，只需将公钥传送给 CA

为了保证安全，避免弱密钥，防止密钥被猜测或分析出来，密钥的一个基本要求是具有足够的随机性，这包括长周期性、非线性、统计意义上的等概率性以及不可预测性等。但是，一个真正的随机序列是无法用计算机模拟产生的，目前常采用物理噪声源方法产生具有足够的随机性的伪随机序列。

对密钥的另一个基本要求是密钥要足够长。决定密钥长度时需要考虑多方面的因素，包括数据价值有多大、数据需要多长的安全期、攻击者的资源情况怎样等。应该注意到，计算机的计算能力和加密算法的发展也是决定密钥长度时要考虑的重要因素。

2. 密钥的存储

密钥的安全存储是密钥管理中的一个重要环节，也是比较困难的一个环节。所谓密钥的安全存储是指要确保密钥在存储状态下的机密性、真实性和完整性。安全可靠的存储介质是密钥安全存储的物质条件，安全严密的访问控制机制是密钥安全存储的管理条件。

密钥安全存储的原则是不允许密钥以明文形式出现在密钥管理设备之外。例如，可以将密钥以明文形式存储在安全的 IC 卡或智能卡中，由专人保管，使用时插入设备中。如果无法做到时，必须用另一个密钥对密钥进行加密以保护该密钥，或由一个可信方分发密钥。

3. 密钥的更新

密钥更新是密钥管理的基本要求，无论密钥是否泄露，都应该定期更新，更新时间取决于给定时间内待加密数据的数量、加密的次数和密钥的种类。会话密钥应当频繁更换，以防止攻击者在长时间内通过截获大量的密文分析出密钥；密钥加密密钥无须频繁更换；而主密钥可有更长的更换时间。

4. 密钥的备份与恢复

为了进一步确保密钥和加密数据的安全，防止密钥遭到毁坏并造成数据丢失，可利用备份的密钥恢复原来的密钥或被加密的数据，密钥的备份本质上也是一种密钥的存储形式。密钥备份有以下几个原则：

（1）备份的密钥应当受到与存储的密钥同样的保护。

（2）为了减少明文形式的密钥的数量，一般都采用高级密钥保护低级密钥的密文形式进行密钥备份。

（3）对于高级密钥，不能采用密文形式备份，一般采用多个密钥分量的形式备份，即把密钥通过门限方案分割成几部分，将每个密钥分量备份到不同的设备或地点，并且指定专人负责。

（4）密钥的备份应当考虑方便恢复，密钥的恢复应当经过授权而且要遵循安全的规章制度。

5. 密钥的销毁

对任何密钥的使用都必须设置有效期。没有哪个密钥能够无限期地使用，否则会带来不可预料的后果，这是因为：

（1）密钥使用时间越长，它泄露的可能性就越大。

（2）如果密钥已经泄露，又没有被使用者察觉，那么密钥使用越久，损失就会越大。

（3）密钥使用越久，对攻击者来说花费精力破译它的诱惑力就越大，甚至会采取穷举法进行攻击。

（4）对同一密钥加密的多个密文进行密码分析相对来说更容易。

因此，当密钥超过有效期或停止使用后，应该对该密钥进行销毁，彻底清除所有踪迹，包括将所有明文、密钥及其他没受保护的重要保密参数全部清除，以禁止攻击者通过观察数据或从废弃的设备中确定旧密钥值。

6. 密钥的撤销

密钥的撤销是从法律上取消密钥与密钥拥有者之间的关联，解除实体在密钥使用过程中应承担的义务。密钥的撤销往往意味着密钥同时也被销毁。

密钥管理的各个过程都要记录日志，方便以后进行审计。

4.2 密钥的分配

密钥分配，通俗地说，就像把钥匙传递给对方。在现实生活中，这应该是一个很简单的问题，因为人们可以面对面地把钥匙交到对方手里；但是在网络环境中，人们不能见面，只能通过网络把密钥发送给对方。而在这中间可能会遭受各种各样的攻击，如窃取密钥或伪造密钥等。如果密钥被敌方掌握了，那么设计得再好的密码系统也没用了，因此密钥分配是密钥管理最重要的环节。

密钥分配是指将密钥安全地分发给通信双方的过程。由于密钥是整个密码系统安全的核心，所以攻击者很可能通过窃取密钥来攻破密码系统。许多情况下，出现安全问题不是因为密码算法被破解，而是因为密钥分配系统被攻破。需要注意的是，对称密码体制和公钥密码体制的密钥分配方式是不同的。

4.2.1 对称密码体制的密钥分配

两个用户 A 和 B 获得共享的对称密钥有如下几种方法：

(1) 密钥由 A 选取并通过物理手段发送给 B。

(2) 密钥由第三方选取并通过物理手段发送给 A 和 B。

(3) 如果 A、B 事先已有一个对称密钥，则其中一方选取新密钥后，用已有的密钥加密新密钥并发送给另一方。

(4) 如果 A 和 B 与第三方分别有一条保密信道(即第三方与每个用户事先共享一个对称密钥)，则第三方为 A、B 选取密钥后，分别在两条保密信道上将密钥发送给 A、B。

提示：如果有 n 个用户，需要两两拥有共享密钥，那么一共需要 $n(n-1)/2$ 个密钥，而采用第(4)种方法，只需要 n 个密钥。

第(4)种方法称为集中式密钥分配方案，它是指由密钥分配中心(Key Distribution Center，KDC)负责密钥的产生并分配给通信双方。在这种情况下，用户不需要保存大量的会话密钥，只需要保存和 KDC 通信的加密密钥。其缺点是通信量大，同时要求具有较好的鉴别功能以鉴别 KDC 和通信双方。图 4.2 是集中式密钥分配方案的实现，称为 Needham-Schroeder 协议。

该密钥分配方案的具体过程如下：

(1) A 向 KDC 发出会话密钥请求，请求的消息由两部分组成，一是 A 和 B 的身份 ID_A 和 ID_B，二是本次业务的唯一标识符 N_1。每次请求的 N_1 都应不同，常用一个时间戳、一个计数器或一个随机数作为这个标识符。A 发给 KDC 的请求可表示为

$$A \rightarrow KDC: ID_A \| ID_B \| N_1$$

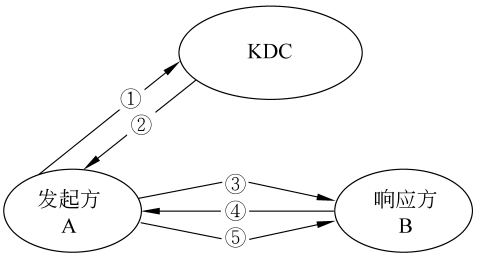

图 4.2　集中式密钥分配方案的实现

提示：‖表示连接符，例如 abc‖fg=abcfg。

(2) KDC 对 A 的请求作出应答。应答是由 KDC 与 A 共享的密钥 K_a 加密的信息，因此只有 A 才能成功地对这一信息进行解密，并且 A 能相信信息的确是由 KDC 发出的。

$$\text{KDC}\rightarrow\text{A}: E_{K_A}[K_S \| \text{ID}_A \| \text{ID}_B \| N_1 \| E_{K_B}[K_s \| \text{ID}_A]]$$

应答中包括 A 希望得到的两项数据：①一次性会话密钥 K_s；②A 在第(1)步中发出的请求，包括一次性随机数 N_1，其目的是使 A 将收到的应答与发出的请求比较，看是否匹配。因此，A 能印证自己发出的请求在被 KDC 收到之前未被篡改，而且 A 还能根据一次性随机数相信自己收到的应答不是重放对过去请求的应答。

KDC 到 A 的应答中含有 A 和 B 的身份 $\text{ID}_A \| \text{ID}_B$，可防止攻击者将 KDC 发往其他用户的应答转向，重放给 A。

此外，应答中还有 B 希望得到的两项数据：①一次性会话密钥 K_s；②A 的身份 ID_A。这两项由 K_B 加密，将由 A 转发给 B，以建立 A 和 B 之间的连接并用于向 B 证明 A 的身份。

(3) A 收到 KDC 的响应后，将会话密钥 K_s 保存起来，同时将经过 KDC 与 B 的共享密钥加密过的消息传送给 B。B 收到后，得到会话密钥 K_s，并从 ID_A 可知对方是 A，而且还从 E_{K_B} 知道 K_s 确实来自 KDC。由于 A 转发的是加密后的密文，所以转发内容不会被窃听。

$$\text{A}\rightarrow\text{B}: E_{K_B}[K_S \| \text{ID}_A]$$

(4) B 用会话密钥加密另一个随机数 N_2，将加密结果发送给 A，并告诉 A，B 当前是可以通信的。

$$\text{B}\rightarrow\text{A}: E_{K_s}[N_2]$$

(5) A 响应 B 发送的消息 N_2，并对 N_2 进行某种函数变换(以防止攻击者将 B 发送给 A 的消息反向重放)，同时用会话密钥 K_s 进行加密，然后将其发送给 B。

$$\text{A}\rightarrow\text{B}: E_{K_s}[f(N_2)]$$

实际上第(3)步已经完成了密钥的分配，第(4)、(5)步结合第(3)步执行的是认证功能，使 B 能够确认其收到的消息不是一个重放。

4.2.2 公钥密码体制的密钥分配

虽然公钥密码体制中使用的公钥可以公开，但必须保证公钥的真实性。公钥的发布一般有以下 4 种方法。

1. 公开发布

用户 A 将自己的公钥公开发布给其他每一个用户。这种方法最简单，但没有认证性，因为任何人都可以伪造 A 的这种公开发布。如果某个用户假装是用户 A，并以 A 的名义向其他用户发送或广播自己的公钥，则在 A 发现假冒者以前，这一假冒者可解密所有发给 A 的加密消息（因为它拥有与该假冒公钥对应的私钥），而且假冒者还能用伪造的密钥获得认证。

2. 公钥目录表

建立一个动态可访问的公钥目录表。公钥目录表的建立、维护以及公钥的发布由可信的实体或组织承担。管理员为每个用户在公钥目录表里建立一个目录项，目录项中包括两个数据项：一是用户名，二是用户的公钥。每一用户都亲自或以某种安全的认证通信向管理员注册自己的公钥，用户可以随时替换自己的公钥。管理员定期公布或定期更新目录。其他用户可以通过公开的渠道访问该公钥目录表以获取公钥。

这种方法比用户公开发布公钥更安全。但它也存在两个缺点：①一旦管理员的私钥被攻击者窃取，则攻击者可以修改公钥目录表，传递伪造的公钥；②用户必须知道这个公钥目录表的位置且信任该公钥目录表。

3. 公钥管理机构（在线服务器方式）

公钥管理机构为用户建立和维护动态的公钥目录。每个用户知道公钥管理机构的公钥，只有公钥管理机构知道自己的私钥。这种方案称为带认证的公钥分配，如图 4.3 所示，步骤如下：

（1）A 发送一个带有时间戳的消息给公钥管理机构 M，以请求 B 的公钥。

（2）M 给 A 发送一个用其私钥 SK_M 签名的包括 B 的公钥 PK_B 在内的消息，A 用 M 的公钥 PK_M 解密得到 B 的公钥 PK_B。

（3）A 用 B 的公钥加密 $ID_A \| N_1$ 并发送给 B，表示请求和 B 通信，B 用其私钥 SK_B 解密成功，就同意通信，然后 B 以同样的方法从 M 处检索到 A 的公钥。

其中，M 应答的消息（如②）中的 Request 用于 A 验证收到的应答的确是对相应请求的应答，且还能验证自己最初发送的请求在被 M 收到之前是否被篡改。最初的时间戳 $Time_1$ 使 A 相信 M 发来的消息不是一个旧消息。消息⑥和⑦使 A、B 能相互确认对方身份，因为只有 B 才能得到 N_1，只有 A 才能得到 N_2。

该方案安全性很高，但也有缺点。只要用户与其他用户通信，就必须向公钥管理机构申请对方的公钥，故可信服务器必须在线，这导致可信服务器可能成为性能的瓶颈。

4. 公钥证书（离线服务器方式）

为解决公钥管理机构的性能瓶颈问题，可以通过公钥证书实现公钥的发布，即使用公钥证书进行公钥分配，这样就不要求与公钥管理机构直接通信。公钥证书由认证机构