第5章 密钥分配与密钥管理

5.1 单钥加密体制的密钥分配

5.1.1 密钥分配的基本方法

两个用户（主机、进程、应用程序）在用单钥密码体制进行保密通信时，首先必须有一个共享的秘密密钥，而且为防止攻击者得到密钥，还必须时常更新密钥。因此，密码系统的强度也依赖于密钥分配技术。两个用户 A 和 B 获得共享密钥的方法有以下几种：

（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 C 分别有一个保密信道，则 C 为 A、B 选取密钥后，分别在两个保密信道上发送给 A、B。

前两种方法称为人工发送。在通信网中，若只有个别用户想进行保密通信，密钥的人工发送还是可行的。然而如果所有用户都要求支持加密服务，则任一对希望通信的用户都必须有一共享密钥。如果有 n 个用户，则密钥数目为 $n(n-1)/2$。因此当 n 很大时，密钥分配的代价非常大，密钥的人工发送是不可行的。

对第三种方法，攻击者一旦获得一个密钥就可获取以后所有的密钥；再者对所有用户分配初始密钥时，代价仍然很大。

第四种方法比较常用，其中的第三方通常是一个负责为用户分配密钥的密钥分配中心。这时每一用户必须和密钥分配中心有一个共享密钥，称为主密钥。通过主密钥分配给一对用户的密钥称为会话密钥，用于这一对用户之间的保密通信。通信完成后，会话密钥即被销毁。如上所述，如果用户数为 n，则会话密钥数为 $n(n-1)/2$。但主密钥数却只需 n 个，所以主密钥可通过物理手段发送。

5.1.2 一个实例

图 5-1 是密钥分配的一个实例。假定两个用户 A、B 分别与密钥分配中心（Key Distribution Center，KDC）有一个共享的主密钥 K_A 和 K_B，A 希望与 B 建立一个共享的一次性会话密钥，可通过以下几步：

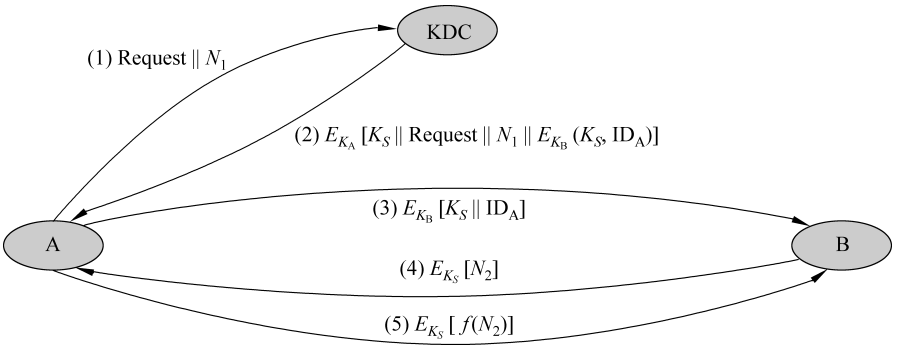

图 5-1　密钥分配实例

(1) A 向 KDC 发出会话密钥请求。表示请求的消息由两个数据项组成：一是 A 和 B 的身份；二是这次业务的唯一识别符 N_1，称 N_1 为一次性随机数，可以是时间戳、计数器或随机数。每次请求所用的 N_1 都应不同，且为防止假冒，应使敌手对 N_1 难以猜测。因此用随机数作为这个识别符最为合适。

(2) KDC 为 A 的请求发出应答。应答是由 K_A 加密的消息，因此只有 A 才能成功地对这一消息解密，并且 A 可相信这一消息的确是由 KDC 发出的。消息中包括 A 希望得到的两项：

- 一次性会话密钥 K_S；
- A 在(1)中发出的请求，包括一次性随机数 N_1，目的是使 A 将收到的应答与发出的请求相比较，看是否匹配。

因此 A 能验证自己发出的请求在被 KDC 收到之前，未被他人篡改。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的过去的应答。

此外，消息中还有 B 希望得到的两项：

- 一次性会话密钥 K_S；
- A 的身份(例如 A 的网络地址)ID_A；

这两项由 K_B 加密，将由 A 转发给 B，以建立 A、B 之间的连接并用于向 B 证明 A 的身份。

(3) A 存储会话密钥，并向 B 转发 $E_{K_B}[K_S \| ID_A]$。因为转发的是由 K_B 加密后的密文，所以转发过程不会被窃听。B 收到后，可得会话密钥 K_S，并从 ID_A 可知另一方是 A，而且还从 E_{K_B} 知道 K_S 的确来自 KDC。

这一步完成后，会话密钥就安全地分配给了 A、B。然而还能继续以下两步：

(4) B 用会话密钥 K_S 加密另一个一次性随机数 N_2，并将加密结果发送给 A。

(5) A 以 $f(N_2)$作为对 B 的应答，其中 f 是对 N_2 进行某种变换(例如加 1)的函数，并将应答用会话密钥加密后发送给 B。

这两步可使 B 相信第(3)步收到的消息不是一个重放。

注意：第(3)步就已完成密钥分配，第(4)、(5)两步结合第(3)步执行的是认证功能。

5.1.3　密钥的分层控制

网络中如果用户数目非常多而且分布的地域非常广，一个 KDC 就无法承担为用户分配密钥的重任。问题的解决方法是使用多个 KDC 的分层结构。例如，在每个小范围（如一个 LAN 或一个建筑物）内，都建立一个本地 KDC。同一范围的用户在进行保密通信时，由本地 KDC 为他们分配密钥。如果两个不同范围的用户想获得共享密钥，则可通过各自的本地 KDC，而两个本地 KDC 的沟通又需经过一个全局 KDC。这样就建立了两层 KDC。类似地，根据网络中用户的数目及分布的地域，可建立三层或多层 KDC。

分层结构可减少主密钥的分布，因为大多数主密钥是在本地 KDC 和本地用户之间共享。再者，分层结构还可将虚假 KDC 的危害限制到一个局部区域。

5.1.4　会话密钥的有效期

会话密钥更换得越频繁，系统的安全性就越高。因为敌手即使获得一个会话密钥，也只能获得很少的密文。但另一方面，会话密钥更换得太频繁，又将延迟用户之间的交换，同时还造成网络负担。所以在决定会话密钥的有效期时，应权衡矛盾的两个方面。

对面向连接的协议，在连接未建立前或断开时，会话密钥的有效期可以很长。而每次建立连接时，都应使用新的会话密钥。如果逻辑连接的时间很长，则应定期更换会话密钥。

无连接协议（如面向业务的协议）无法明确地决定更换密钥的频率。为安全起见，用户每进行一次交换，都用新的会话密钥。然而这又失去了无连接协议主要的优势，即对每个业务都有最少的费用和最短的延迟。比较好的方案是在某一固定周期内或对一定数目的业务使用同一会话密钥。

5.1.5　无中心的密钥分配

用密钥分配中心为用户分配密钥时，要求所有用户都信任 KDC，同时还要求对 KDC 加以保护。如果密钥的分配是无中心的，则不必有以上两个要求。然而如果每个用户都能和自己想与之建立联系的另一用户安全地通信，则对有 n 个用户的网络来说，主密钥应多达 $n(n-1)/2$ 个。当 n 很大时，这种方案无实用价值，但在整个网络的局部范围却非常有用。

无中心的密钥分配时，两个用户 A 和 B 建立会话密钥需经过以下 3 步，见图 5-2。

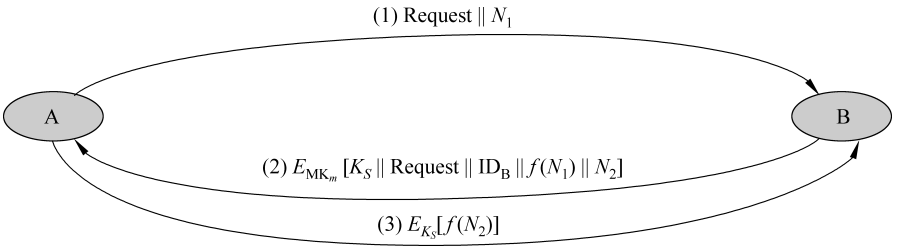

图 5-2　无中心的密钥分配

(1) A向B发出建立会话密钥的请求和一个一次性随机数 N_1。

(2) B用与A共享的主密钥 MK_m 对应答的消息加密,并发送给A。应答的消息中有B选取的会话密钥、B的身份、$f(N_1)$和另一个一次性随机数 N_2。

(3) A使用新建立的会话密钥 K_S 对 $f(N_2)$加密后返回给B。

5.1.6 密钥的控制使用

因为密钥可根据其不同用途分为会话密钥和主密钥两种类型,所以还希望对密钥的使用方式加以某种控制,会话密钥又称为数据加密密钥,主密钥又称为密钥加密密钥。

如果主密钥泄露了,则相应的会话密钥也将泄露,因此主密钥的安全性应高于会话密钥的安全性。一般在密钥分配中心以及终端系统中,主密钥都是物理上安全的,如果把主密钥当作会话密钥注入加密设备,那么其安全性则降低。

单钥体制中的密钥控制技术有:

1. 密钥标签

用于DES的密钥控制,将DES的64比特密钥中的8个校验位作为控制使用这一密钥的标签。标签中各比特的含义如下:

- 一个比特表示这个密钥是会话密钥还是主密钥。
- 一个比特表示这个密钥是否能用于加密。
- 一个比特表示这个密钥是否能用于解密。
- 其他比特无特定含义,留待以后使用。

由于标签是在密钥之中的,所以在分配密钥时,标签与密钥一起被加密,因此可对标签起到保护。本方案的缺点一是标签的长度被限制为8比特,限制了它的灵活性和功能。二是由于标签是以密文形式传送,只有解密后才能使用,因而限制了对密钥使用的控制方式。

2. 控制矢量

这一方案比上一方案灵活。方案中对每一会话密钥都指定一个相应的控制矢量,控制矢量分为若干字段,分别用于说明在不同情况下密钥是被允许使用还是不被允许使用,且控制矢量的长度可变。控制矢量是在KDC产生密钥时加在密钥之中的,过程由图5-3(a)所示。首先由一个哈希函数将控制矢量压缩到与加密密钥等长,然后与主密钥异或后作为加密会话密钥的密钥,即

$$H = h(\mathrm{CV})$$
$$K_{\mathrm{in}} = K_m \oplus H$$
$$K_{\mathrm{out}} = E_{K_m \oplus H}[K_S]$$

其中,CV是控制矢量,h 是哈希函数,K_m 是主密钥,K_S 是会话密钥。会话密钥的恢复过程由图5-3(b)所示,表示为

$$K_S = D_{K_m \oplus H}[E_{K_m \oplus H}[K_S]]$$

KDC在向用户发送会话密钥时,同时以明文形式发送控制矢量。用户只有使用与KDC共享的主密钥以及KDC发送来的控制矢量才能恢复会话密钥,因此还必须保留会

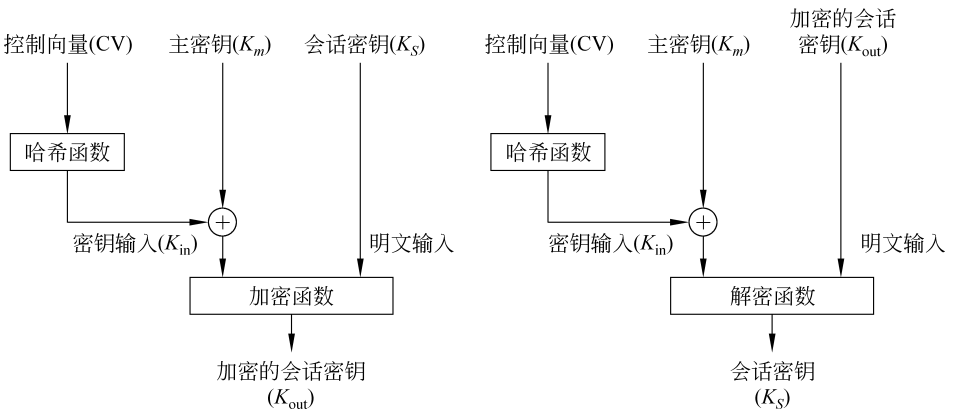

图 5-3　控制矢量的使用方式

话密钥与其控制矢量之间的对应关系。

与使用 8 比特的密钥标签相比，使用控制矢量有两个优点：第一，控制矢量的长度没有限制，因此可对密钥的使用施加任意复杂的控制；第二，控制矢量始终是以明文形式存在，因此可在任一阶段对密钥的使用施加控制。

5.2　公钥加密体制的密钥管理

5.1 节介绍了单钥密码体制中的密钥分配问题，而公钥加密的一个主要用途是分配单钥密码体制使用的密钥。本节介绍两个内容：一是公钥密码体制所用的公开密钥的分配；二是如何用公钥体制来分配单钥密码体制所需的密钥。

5.2.1　公钥的分配

公钥的分配方法有以下几类。

1. 公开发布

公开发布指用户将自己的公钥发给每一其他用户，或向某一团体广播。例如，PGP (Pretty Good Privacy)中采用了 RSA 算法，它的很多用户都是将自己的公钥附加到消息上，然后发送到公开(公共)区域，如因特网邮件列表。

这种方法虽然简单，但有一个非常大的缺点，即任何人都可伪造这种公开发布。如果某个用户假装是用户 A 并以 A 的名义向另一用户发送或广播自己的公开钥，则在 A 发现假冒者以前，这一假冒者可解读所有意欲发向 A 的加密消息，而且假冒者还能用伪造的密钥获得认证。

2. 公用目录表

公用目录表指建立一个公用的公钥动态目录表，目录表的建立、维护以及公钥的分布由某个可信的实体或组织承担，称这个实体或组织为公用目录的管理员。与第一种分配

方法相比,这种方法的安全性更高。该方案有以下一些组成部分:

(1) 管理员为每个用户都在目录表中建立一个目录,目录中有两个数据项:一是用户名,二是用户的公开钥。

(2) 每一用户都亲自或以某种安全的认证通信在管理者那里为自己的公开钥注册。

(3) 用户如果由于自己的公开钥用过的次数太多或由于与公开钥相关的秘密钥已被泄露,则可随时用新密钥替换现有的密钥。

(4) 管理员定期公布或定期更新目录表。例如,像电话号码本一样公布目录表或在发行量很大的报纸上公布目录表的更新。

(5) 用户可通过电子手段访问目录表,这时从管理员到用户必须有安全的认证通信。

本方案的安全性虽然高于公开发布的安全性,但仍易受攻击。如果敌手成功地获取了管理员的秘密钥,就可伪造一个公钥目录表,以后既可假冒任一用户,又能监听发往任一用户的消息,且公用目录表还易受到敌手的窜扰。

3. 公钥管理机构

如果在公钥目录表中对公钥的分配施加更严密的控制,安全性将会更强。与公用目录表类似,这里假定有一个公钥管理机构来为各用户建立、维护动态的公钥目录,但同时对系统提出以下要求,即:每个用户都可靠地知道管理机构的公开钥,而只有管理机构自己知道相应的秘密钥。公开钥的分配步骤如下(见图5-4)。

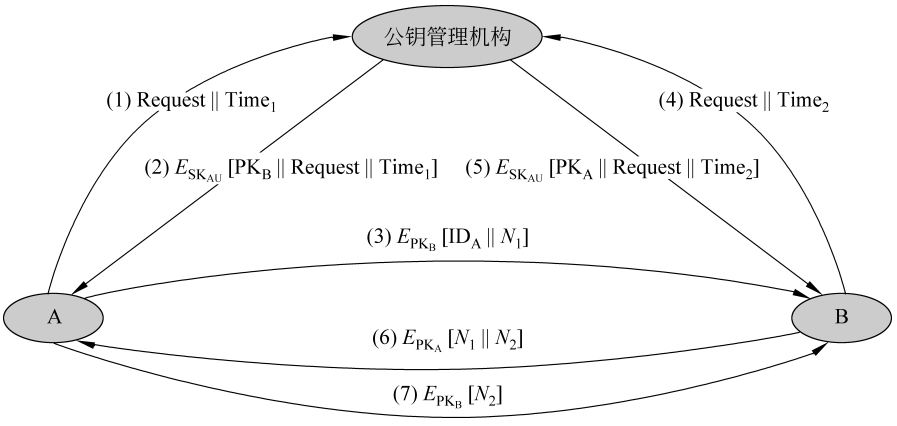

图5-4 公钥管理机构分配公钥

(1) 用户A向公钥管理机构发送一个带时间戳的消息,消息中有获取用户B的当前公钥的请求。

(2) 管理机构对A的请求作出应答,应答由一个消息表示,该消息由管理机构用自己的秘密钥SK_{AU}加密,因此A能用管理机构的公开钥解密,并使A相信这个消息的确是来源于管理机构。

应答的消息中有以下几项。

- B的公钥PK_B,A可用它对将发往B的消息加密。
- A的请求,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

初发出的请求在被管理机构收到以前未被篡改。

- 最初的时间戳，以使 A 相信管理机构发来的消息不是一个旧消息，因此消息中的公开钥的确是 B 当前的公钥。

(3) A 用 B 的公开钥对一个消息加密后发往 B，这个消息有两个数据项，一是 A 的身份 ID_A，二是一个一次性随机数 N_1，用于唯一地标识这次业务。

(4)、(5) B 以相同方式从管理机构获取 A 的公开钥。这时，A 和 B 都已安全地得到了对方的公钥，所以可进行保密通信。然而，他们也许还希望有以下两步，以认证对方。

(6) B 用 PK_A 对一个消息加密后发往 A，该消息的数据项有 A 的一次性随机数 N_1 和 B 产生的一个新一次性随机数 N_2。因为只有 B 能解密(3)的消息，所以 A 收到的消息中的 N_1 可使其相信通信的另一方的确是 B。

(7) A 用 B 的公开钥对 N_2 加密后返回给 B，可使 B 相信通信的另一方的确是 A。

以上过程共发送了 7 个消息，其中前 4 个消息用于获取对方的公开钥。用户得到对方的公开钥后存下可供以后使用，这样就不必再发送前 4 个消息了，然而还必须定期地通过密钥管理中心获取通信对方的公开钥，以免对方的公开钥更新后无法保证当前的通信。

4. 公钥证书

上述公钥管理机构分配公开钥时也有缺点，由于每一用户要想和他人联系都需求助于管理机构，所以管理机构有可能成为系统的瓶颈，而且由管理机构维护的公钥目录表也易被敌手窜扰。

分配公钥的另一方法是公钥证书，用户通过公钥证书相互之间交换自己的公钥而无须与公钥管理机构联系。公钥证书由证书管理机构(Certificate Authority，CA)为用户建立，其中的数据项有与该用户的秘密钥相匹配的公开钥及用户的身份和时间戳等，所有的数据项经 CA 用自己的秘密钥签字后就形成证书，即证书的形式为 $C_A = E_{SK_{CA}}[T, ID_A, PK_A]$，其中，$ID_A$ 是用户 A 的身份，PK_A 是 A 的公钥，T 是当前时间戳，SK_{CA} 是 CA 的秘密钥，C_A 即是为用户 A 产生的证书。产生过程如图 5-5 所示。用户可将自己的公开钥通过公钥证书发给另一用户，接收方可用 CA 的公钥 PK_{CA} 对证书加以验证，即

$$D_{PK_{CA}}[C_A] = D_{PK_{CA}}[E_{SK_{CA}}[T, ID_A, PK_A]] = (T, ID_A, PK_A)$$

因为只有用 CA 的公钥才能解读证书，接收方从而验证了证书的确是由 CA 发放的，且也获得了发方的身份 ID_A 和公开钥 PK_A。时间戳 T 为接收方保证了收到的证书的新鲜性，用于防止发方或敌方重放一个旧证书。因此时间戳可被当作截止日期，证书如果过旧，则被吊销。

5.2.2　用公钥加密分配单钥密码体制的密钥

公开钥分配完成后，用户就可用公钥加密体制进行保密通信了。然而由于公钥加密的速度过慢，以此进行保密通信不太合适，但用于分配单钥密码体制的密钥却非常合适。

1. 简单分配

图 5-6 表示简单使用公钥加密算法建立会话密钥的过程，如果 A 希望与 B 通信，则可通过以下几步建立会话密钥：

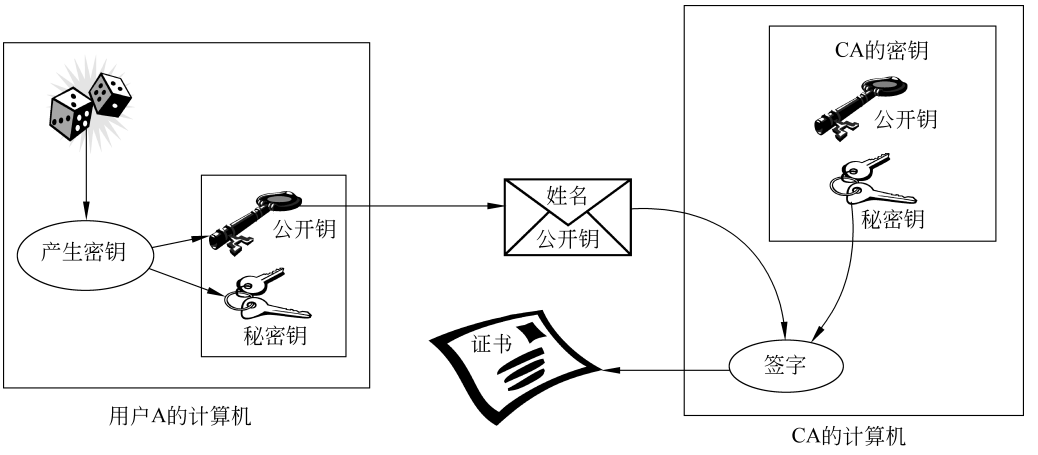

图 5-5　证书的产生过程

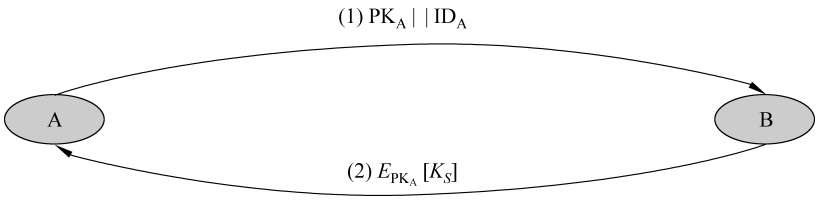

图 5-6　简单使用公钥加密算法建立会话密钥

(1) A 产生自己的一对密钥$\{PK_A, SK_A\}$,并向 B 发送 $PK_A||ID_A$,其中,ID_A 表示 A 的身份。

(2) B 产生会话密钥 K_S,并用 A 的公开钥 PK_A 对 K_S 加密后发往 A。

(3) A 由 $D_{SK_A}[E_{PK_A}[K_S]]$恢复会话密钥。因为只有 A 能解读 K_S,所以仅 A、B 知道这一共享密钥。

(4) A 销毁$\{PK_A, SK_A\}$,B 销毁 PK_A。

A、B 现在可以用单钥加密算法以 K_S 作为会话密钥进行保密通信,通信完成后,又都将 K_S 销毁。这种分配法尽管简单,但却由于 A、B 双方在通信前和完成通信后,都未存储密钥,因此,密钥泄露的危险性为最小,且可防止双方的通信被敌手监听。

这一协议易受到主动攻击,如果敌手 E 已接入 A、B 双方的通信信道,就可以如下不被察觉的方式截获双方的通信:

(1) 与上面的(1)相同。

(2) E 截获 A 的发送后,建立自己的一对密钥$\{PK_E, SK_E\}$,并将 $PK_E||ID_A$ 发送给 B。

(3) B 产生会话密钥 K_S 后,将 $E_{PK_E}[K_S]$发送出去。

(4) E 截获 B 发送的消息后,由 $D_{SK_E}[E_{PK_E}[K_S]]$解读 K_S。

(5) E 再将 $E_{PK_A}[K_S]$发往 A。

现在 A 和 B 知道 K_S,但并未意识到 K_S 已被 E 截获。A、B 在用 K_S 通信时,E 就可

以实施监听。

2. 具有保密性和认证性的密钥分配

图 5-7 所示的密钥分配过程具有保密性和认证性，因此既可防止被动攻击，又可防止主动攻击。

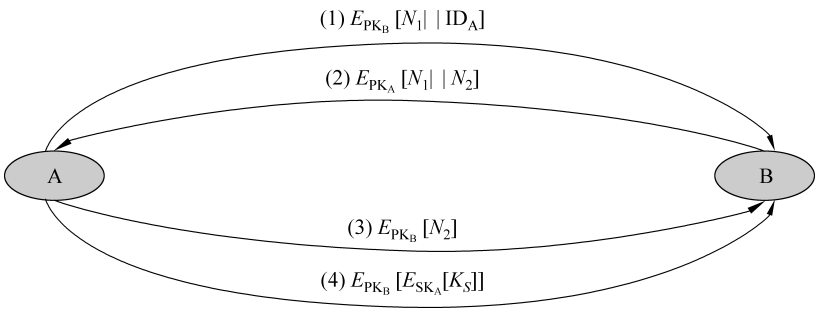

图 5-7　具有保密性和认证性的密钥分配

假定 A、B 双方已完成公钥交换，可按以下步骤建立共享会话密钥：

(1) A 用 B 的公开钥加密 A 的身份 ID_A 和一个一次性随机数 N_1 后发往 B，其中，N_1 用于唯一地标识这一业务。

(2) B 用 A 的公开钥 PK_A 加密 A 的一次性随机数 N_1 和 B 新产生的一次性随机数 N_2 后发往 A。因为只有 B 能解读(1)中的加密消息，所以 B 发来的消息中 N_1 的存在可使 A 相信对方的确是 B。

(3) A 用 B 的公钥 PK_B 对 N_2 加密后返回给 B，以使 B 相信对方的确是 A。

(4) A 选取一个会话密钥 K_S，然后将 $M=E_{PK_B}[E_{SK_A}[K_S]]$发给 B，用 B 的公开钥加密是为保证只有 B 能解读加密结果，用 A 的秘密钥加密是保证该加密结果只有 A 能发送。

(5) B 以 $D_{PK_A}[D_{SK_B}[M]]$恢复会话密钥。

5.2.3　Diffie-Hellman 密钥交换

Diffie-Hellman 密钥交换是 W. Diffie 和 M. Hellman 于 1976 年提出的第一个公钥密码算法，已在很多商业产品中得到应用。该算法的唯一目的是使得两个用户能够安全地交换密钥，得到一个共享的会话密钥，算法本身不能用于加、解密。

算法的安全性基于求离散对数的困难性。

图 5-8 表示 Diffie-Hellman 密钥交换过程，其中，p 是大素数，a 是 p 的本原根，p 和 a 作为公开的全程元素。用户 A 选择一个保密的随机整数 X_A，并将 $Y_A=a^{X_A} \bmod p$ 发送给用户 B。类似地，用户 B 选择一个保密的随机整数 X_B，并将 $Y_B=a^{X_B} \bmod p$ 发送给用户 A。然后 A 和 B 分别由 $K=Y_B^{X_A} \bmod p$ 和 $K=Y_A^{X_B} \bmod p$ 计算出的就是共享密钥，这是因为

$$Y_B^{X_A} \bmod p=(a^{X_B} \bmod p)^{X_A} \bmod p=(a^{X_B})^{X_A} \bmod p=a^{X_B X_A} \bmod p$$
$$=(a^{X_A})^{X_B} \bmod p=(a^{X_A} \bmod p)^{X_B} \bmod p=Y_A^{X_B} \bmod p$$

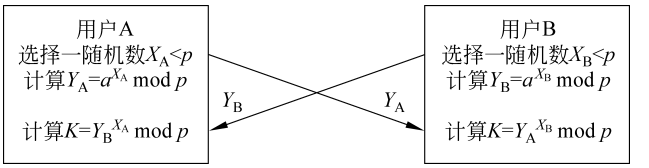

图 5-8 Diffie-Hellman 密钥交换

因为 X_A、X_B 是保密的,敌手只能得到 p、a、Y_A、Y_B,要想得到 K,则必须得到 X_A、X_B 中的一个,这意味着需要求离散对数。因此敌手求 K 是不可行的。

【例 5-1】 $p=97$,$a=5$,A 和 B 分别秘密选 $X_A=36$、$X_B=58$,并分别计算 $Y_A=5^{36} \bmod 97=50$,$Y_B=5^{58} \bmod 97=44$。在交换 Y_A,Y_B 后,分别计算

$$K=Y_B^{X_A} \bmod 97=44^{36} \bmod 97=75, \quad K=Y_A^{X_B} \bmod 97=50^{58} \bmod 97=75$$

5.3 随机数的产生

随机数在密码学中起着重要的作用。本节首先介绍随机数在密码学中的作用,然后介绍产生随机数的一些方法。

5.3.1 随机数的使用

很多密码算法都需使用随机数,例如:

- 相互认证,如在图 5-1、图 5-2、图 5-4 和图 5-7 所示的密钥分配中,都使用了一次性随机数来防止重放攻击。
- 会话密钥的产生,用随机数作为会话密钥。
- 公钥密码算法中密钥的产生,用随机数作为公钥密码算法中的密钥,如图 5-5 所示;或以随机数来产生公钥密码算法中的密钥,如图 5-8 所示。

在随机数的上述应用中,都要求随机数序列满足随机性和不可预测性。

1. 随机性

以下两个准则常用来保障数列的随机性:

(1) 均匀分布——数列中每个数出现的频率应相等或近似相等。

(2) 独立性——数列中任意一个数都不能由其他数推出。

数列是否满足均匀分布可通过检测得出,而是否满足独立性则无法检测。相反却有很多检测方法能证明数列不满足独立性。通常检测数列是否满足独立性的方法是在对数列进行了足够多次检测后都不能证明不满足独立性,就可比较有把握地相信该数列满足独立性。

在设计密码算法时,由于真随机数难以获得,经常使用似乎是随机的数列,这样的数列称为伪随机数列,这样的随机数称为伪随机数。

2. 不可预测性

在诸如相互认证和会话密钥的产生等应用中,不仅要求数列具有随机性,而且要求对