第 5 章

身份认证与网络安全

本章要点：

☑ 身份认证技术概述
☑ 基于口令的身份认证
☑ 双因素认证技术
☑ 基于 X509 证书的身份认证
☑ 安全认证协议
☑ USB Key 认证
☑ 基于生物特征的身份认证
☑ 零知识认证技术
☑ 网络认证与授权管理

5.1 身份认证技术

认证(Authentication)指的是对某人或某物与其所声称的是否相符或有效进行确认的一个过程。认证的基本思想是验证被验证者的一个或多个参数是否真实与有效(比如验证他是谁,他具有哪些特征,他有什么可以识别他的东西),以此达到认证的目的,而认证的主要目的是为其他安全措施(如访问控制和审计等)提供相关的鉴别依据。本节介绍身份认证的概念、身份认证系统的特征及分类。

5.1.1 身份认证技术简介

G. J. Simmons 在 1984 年提出了认证系统的信息理论。他将信息论用于研究认证系统的理论安全性和实际安全问题,也指出了认证系统的性能极限以及设计认证码所必须遵循的原则。他在认证系统中的地位与香农的信息理论在保密系统中的地位一样重要,他为研究认证系统奠定了理论基础。认证理论的主要目标有两个:一个是推导欺骗者成功的概率的下界;另一个是构造欺骗者成功的概率尽可能小的认证码。

认证技术是信息安全中的一个重要内容,信息安全包括两种主要的认证技术:消息认证与身份认证,消息认证用于保证信息在传送过程中的完整性和信息来源的可靠性,身份认证是指计算机及网络系统确认操作者身份的过程,身份认证则用于鉴别用户身份,限制非法用户访问网络资源。身份认证用于解决访问者的物理身份和数字身份的一致性问

题，给其他安全技术提供权限管理的依据。在网上商务日益火爆的今天，在某些应用场合，认证技术可能比信息加密本身更加重要。在信息安全理论与技术中，认证技术是很重要的一个方面。在安全系统中，身份认证是保障信息安全的第一道关卡，是保护网络安全的一道重要防线。

5.1.2 身份认证系统的特征

身份认证系统一般需要具有以下特征：

- 验证者正确识别合法用户的概率极大。
- 攻击者伪装成合法用户骗取验证者信任的成功率极小。
- 通过重放认证信息进行欺骗和伪装的成功率极小。
- 计算有效性：实现身份认证的算法计算量足够小。
- 通信有效性：实现身份认证所需的通信量足够小。
- 秘密参数能够安全存储。
- 第三方的可信赖性高。
- 可证明安全性。

5.1.3 用户身份认证的分类

根据被认证方证明身份所使用秘密的不同，认证用户身份的方法大体有三种，这三种方法可以单独使用或联合使用。

1. 用户知道的秘密

用户知道的秘密，如口令、个人识别码（Personal Identification Number，PIN）或密钥等。最常见的鉴别和认证方式是个人识别码加上口令。口令系统工作时需要用户的识别码及其口令。系统将口令和该用户预存在系统中的口令进行比较，如果口令匹配，用户就被认证并获准访问。在通常情况下，用户名是公开的，因此作为身份唯一标志的口令就显得格外重要。但是，由于这种口令是可以重复使用的，攻击者有足够的时间来获取口令。

2. 用户拥有的令牌

用户拥有的令牌如：银行卡或智能卡，用户为了鉴别和认证的目的所拥有的物体称为令牌，令牌包括记忆令牌和智能令牌。记忆令牌存储但不处理信息。对令牌的读写通过专用读写器完成。最常见的记忆令牌是磁卡，磁卡表面封装有磁性薄条。在计算机系统中使用记忆令牌进行认证的常见应用是自动提款机。通常，使用智能令牌时还需要用户输入 PIN 用来为智能令牌“解锁”以便使用。这种认证方式是一种双因素的认证方式（PIN＋智能令牌），即使 PIN 或智能令牌被窃取，用户仍不会被冒充。

3. 用户本身的生物特征

用户本身的生物特征，如语音特征、面部特征或指纹等。生物识别认证技术利用各人独一无二的特征（或属性）对人的身份进行识别。这包括生理属性（如指纹、手掌几何形状或视网膜图案）或行为特征（如语音模式和笔迹签署）。生物识别系统可以提升计算机系统的安全性，但是生物识别技术的缺陷源于测量和抽取生物特征的技术的复杂性和生物属性的自然变化，这些特征在某些情况下会发生变化。

上述几种认证方式在应用中存在一定的缺点，当前国内外将认证技术的研究重点逐步转移到了基于 X.509 数字证书的认证技术和为 TCP/IP 网络提供可信第三方鉴别的 Kerberos 协议认证技术，下面我们将分别予以介绍。

5.2　基于口令的身份认证

对于身份认证技术来说，基于口令的认证方式是最常用的一种技术。基于口令的认证方式就是用户输入自己的口令，计算机验证并给予用户相应的权限。鉴别用户身份最常见也是最简单的方法就是口令认证：系统为每一个合法用户建立一个用户名/口令对，当用户登录系统或使用某项功能时，提示用户输入自己的用户名和口令，系统通过核对用户输入的用户名、口令与系统内已有的合法用户的用户名/口令对是否匹配，如与某一项用户名/口令对匹配，则该用户的身份得到了认证。这种方法有如下缺点：其安全性仅仅基于用户口令的保密性，而用户口令一般较短且容易猜测，因此这种方案不能抵御口令猜测攻击；另外，攻击者可能窃听通信信道或进行网络窥探(Sniffing)，口令的明文传输使得攻击者只要能在口令传输过程中获得用户口令，系统就会被攻破，尽管有许多漏洞，这种方法在非网络环境下还是经常被采用的。

5.2.1　口令的存储

基于口令的认证方式需要解决的一个重要问题是口令的存储。一般有以下两种方法进行口令存储。

1. 直接明文存储口令

这种方式有很大风险，任何人只要得到存储口令的数据库，就可以得到全体人员的口令，比如攻击者可以设法得到一个低优先级的账号和口令，进入系统后得到存储口令的文件，因为是明文存储，这样，他就可以得到全体人员的口令，包括管理员的口令，然后以管理员的身份进入系统，进行非法操作。

2. 散列存储口令

散列函数的目的是为文件、报文或其他分组数据产生类似于“指纹”的特征信息。对于每一个用户，系统存储账号和散列值对在一个口令文件中，当用户登录时，用户输入口令 x，系统计算 $H(x)$，然后与口令文件中相应的散列值进行比对，成功则允许登录，否则拒绝登录。在文件中存储口令的散列值而不是口令的明文，优点在于黑客即使得到口令文件，通过散列值想要计算出原始口令在计算上也是不可能的，这就相对增加了安全性。

5.2.2　口令机制

1. 口令传递

最简单的口令机制是以明文的形式把口令从用户传送到服务器。为了验证口令，服务器中存储口令文件，其中包含了口令的明文形式(附于用户名)或口令在单向函数下的映射。后者是 UNIX 系统的经典方法，且还用于 FTP 和 Telnet 的远程认证。在远程认证的情况下，这种机制的缺陷很明显，因为口令会很容易地被窃听者从网络上侦听下来。

2. 激励-响应

激励-响应是更为安全的口令认证形式。这种情况下，口令从不以明文的形式传送，而是用来对每次认证时认证服务器所选取的激励进行秘密的函数计算。这提供了认证的新鲜性。但也使口令易受到口令猜测攻击，这种攻击是指假设入侵者拥有个相对较小的口令字典，其中包含了许多普通的口令。入侵者首先记录包含了激励和响应的认证会话，然后用一些可能的口令对激励进行计算，看能否得到同样的响应。如果能的话，说明这是个合法的用户口令。

3. 一次性口令

激励-响应机制的一种变形称为一次性口令机制，在这种机制下，用户每次验证自己身份时使用不同的口令。如果这些一次性口令是从一个用户记忆的口令推导而来的，那么这个用户记忆的口令仍易受到口令猜测的攻击。另外一种方法是把这些一次性口令全部写在纸片上让用户保存，这样就叫防止上述攻击，且口令不会被重用。但对于用户来说，需携带大批的口令并保证这些口令的安全和保密，且每次都要输入相对复杂的字符串，这是非常不方便的。为了防止重传攻击，可在每次提交的口令中增加随机数，时间标签等因素，从而提供一次一密的功能，提高了整个认证系统的安全性。

5.2.3 对口令协议的基本攻击

对口令协议的基本攻击包括以下几种。

1. 窃听

入侵者搭线窃听，试图从正在进行的通信中获得有用的信息。

2. 重放

入侵者记录过去通信中的消息并在以后的通信中重放它们。

3. 中间人攻击

入侵拦截各主体之间的消息，并用自己的消息来取代它们。在向服务器发送的消息中它假冒用户的身份，同样，在向用户发送的消息中它假冒服务器的身份。

4. 口令猜测

攻击假设入侵者拥有一个相对较小的口令字典，其中包含了许多普通的口令。利用该口令字典，入侵者主要用以下两种方法进行攻击，它们是：

（1）离线攻击入侵者记录过去的通信，然后遍历口令字典，查找与所记录的通信相一致的口令。如果发现了这样的口令，那么入侵者就能够断定这是某个用户的口令。

（2）在线攻击入侵者重复地从口令字典中选取口令并用它来假冒合法用户。如果假冒失败了，入侵者就把这个口令从口令字典中删除，再用其他的口令进行尝试。在实践中，防止这种在线攻击的标准方法是限制口令到期之前允许用户登录失败的次数，或降低允许用户登录的频率。

5. 内部人员辅助攻击

很多时候入侵者可能得到内部人员的帮助进行攻击。实际上，入侵者往往就是系统中的一个用户，因此我们应该考虑到这种可能性，即入侵者拥有自己的账户及相应的合法口令。我们应该确保在这种情况下，协议能够防止入侵者用合法的账户来攻击别人的账户。

6. 秘密揭露

入侵者可能会偶尔得到应该被参与协议的主体保持为秘密的敏感数据(如当服务器或用户被损害时)。在这种情况下,协议的目标就是使得密钥或文件的泄露对整个系统的影响最小化。特别是,在口令机制的环境中,我们需要考虑泄露的密钥对导出的会话密钥(反之亦然)的影响以及损害口令文件或服务器密钥所产生的影响。

5.2.4　口令认证的安全性

设计安全口令机制的困难之处在于口令空间通常较小,比随机密钥更易受到攻击。特别是离线口令猜测攻击之类的穷尽搜索攻击非常有效;且当使用口令作为加密密钥时,总是假设即使口令是从个很小的口令集中选取,加密函数仍是安全的。但这些假设非同寻常,想形式化地证明现有协议安全性是非常困难的。

总体来说,基于口令的认证方式虽然较为常用,但它存在严重的安全问题。它是一种单因素的认证,即仅通过口令一个因素来进行认证,也就是说它的安全性仅依赖于口令,口令一旦泄露,用户即可被冒充。为了提高安全性,往往会使用双因素认证技术。

5.3　双因素认证技术

身份认证技术的应用信息系统中,通过一个条件的符合来证明一个人的身份称之为单因素认证,由于仅使用一种条件判断用户的身份容易被仿冒,可以通过组合两种不同条件来证明一个人的身份,称之为双因素认证。

双因素身份认证机制,通常是在静态密码的基础上,增加一个物理因素,从而构成一个他人无法复制和识破的安全密码。最常见的物理因素有生物特征和智能卡。前者因技术及设备的复杂性而只在某些特殊的领域中使用;后者是目前应用的最广泛的双因素身份验证机制,又称动态口令验证机制。

动态口令技术是一种让用户密码按照时间或使用次数不断变化、每个密码只能使用一次的技术。它采用一种称为动态令牌的专用硬件,内置电源、密码生成芯片和显示屏,密码生成芯片运行专门的密码算法,根据当前时间或使用次数生成当前密码并显示在显示屏上。认证服务器采用相同的算法计算当前的有效密码。用户使用时只需要将动态令牌上显示的当前密码输入客户端计算机,即可实现身份认证。由于每次使用的密码必须由动态令牌来产生,只有合法用户才持有该硬件,所以只要通过密码验证就可以认为该用户的身份是可靠的。而用户每次使用的密码都不相同,即使黑客截获了一次密码,也无法利用这个密码来仿冒合法用户的身份,从而提高登录过程的安全性。

5.3.1　双因素认证原理

双因素身份认证系统主要由三个部分组成:动态口令产生算法、客户端软件代理和管理服务器。这三个部分协同工作,管理服务器在选定的网络节点之间(通过签发代理主机证书)建立一个保护的环境。每个受保护的网络节点都是一个客户端,必须运行客户端软件代理。用户访问受保护网络节点时,客户端软件代理要求用户输入验证信息(用户

名、PIN 码和动态密码），并同时将客户端的节点密文传输到管理服务器。管理服务器在接收到验证信息后，首先根据节点密文确定客户端是否为可信节点；然后根据用户名在用户信息数据库中取出用户的初始密钥，并在当前时间前后的一定时间段内生成一系列动态口令，如果用户提交的动态口令在这组口令中得以匹配，并且 PIN 码也相符，则可以认定该用户为合法用户，接受用户的验证请求。这种动态口令和用户 PIN 码相结合的验证方式称为双因素验证方式。

5.3.2 动态口令的产生

动态口令产生算法，一般是采用特定的运算函式或流程，即基本函数，加上具有变动性的一些参数，即基本元素。利用基本元素经过基本函数的运算流程得到结果，产生的内容再转换为使用的密码，由于基本元素具有每次变化的特性，因此每次产生的密码都会不相同，所以称为动态口令。动态口令产生算法一般可以以软件形式、硬件形式或智能卡形式实现。其处理过程如图 5-1 所示。

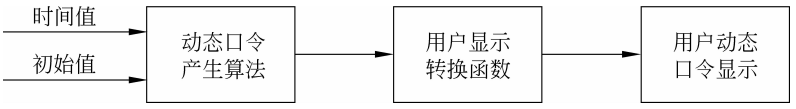

图 5-1 动态口令的产生与显示

5.3.3 客户端软件代理

客户端软件代理是实现认证功能的中间部分，它部署在应用服务器上，用来实施动态口令的安全策略。客户端软件代理主要功能是将具体应用的身份认证请求通过安全的通道传输给管理服务器，并通知用户验证结果。被保护的服务器安装了客户端软件代理后，用户无论是通过本地还是远程访问该服务器时，在获取它的访问权前，访问请求将被截取，通过安全的通道向管理服务器证明身份。以此来控制用户权限，决定用户被授权后，能够访问和不能访问哪些资源。

5.3.4 管理服务器

管理服务器是网络中认证引擎。其主要作用为：验证用户口令的有效性，向用户签发口令令牌，签发可信代理主机证书，实时监控，创建日志信息等。客户端软件负责把用户信息传递给指定的管理服务器，然后对返回的响应进行操作。管理服务器负责接收用户连接请求。在管理服务器中设立一个中心数据库，这个中心数据库包括用户身份认证信息（比如用户名、口令），根据这个中心数据库来认证用户。通常管理服务会采用 Radius 服务器。Radius 是一种客户机及服务器协议。Radius 服务器可支持许多种方法来认证用户的身份。

5.3.5 双因素身份验证系统的几个要点

双因素身份验证系统包含以下几个要点。

1. 时间同步机制

在系统中，用户令牌和管理服务器共享相同的动态口令生成算法，不同的令牌（用户）拥有唯一的初始密钥，动态口令生成算法根据初始密钥和当前时间产生动态口令。由于在同一时刻管理服务器和令牌需要计算出相同的动态口令。而动态口令的产生依赖于时间值和初始密钥，所以时钟的同步很重要。对于可能分布在不同地区的管理服务器和令牌，采用全球同步时间保证两者之间的时钟始终一致。这样在硬件精度能够保证的前提下，令牌和管理服务器之间便能保证时间同步。

2. 时间漂移的处理

受到硬件制作、地域温度等各种因素的影响，令牌时钟会出现微小的漂移，久而久之就有可能造成令牌和服务器之间时钟不一致，影响用户验证处理。所以将管理服务器设计在某一时间窗口的基础上进行验证（通常为 3 分钟），即当用户提供的动态口令与服务器根据当前时间产生的口令不一致时，如果用户名和 PIN 码正确，则服务器同时将用户动态口令与前一分钟和后一分钟的口令相比较，如果匹配，则仍然认为口令有效，同时将偏移量记录。

3. 输入错误的处理

系统中设置有一个阀值，当用户密码连续错误的次数超过这个阀值时，必须要求用户连续两次输入正确的密码。另外，动态口令验证系统是根据时间同步原理来实现令牌和服务器之间的同步，它们之间有可能存在时间漂移。用户即使输入令牌显示的正确口令，该口令在服务器端也可能是过期的。此时也会要求用户输入下一个令牌值。如果两次口令均正确，则可以确认用户身份，并根据两次输入校正时间偏移。

双因素认证系统以较少的投资、较少的工程实施量，可以有效地解决企业在身份认证方面的安全隐患，并且提供更有效更方便的用户管理。利用动态口令，可以解决单一固定口令容易被他人窃取和冒名使用的问题；利用 PIN 码和动态口令相结合，可以解决由于令牌丢失而口令泄露的问题。

5.4　基于 X509 证书的身份认证

5.4.1　X509 证书的格式及含义

X.509 是定义目录服务建议 X.500 系列的一部分，X.500 目录中的条目称为目录信息树（Directory Information Tree，DIT）的层次树形结构来组织，持有此证书的用户就可以凭此证书访问那些信任（Certificate Authority，CA）的服务器。基于 X.509 证书的认证技术依赖于共同信赖的第三方来实现认证，这里可信赖的第三方是指 CA 的认证机构。该认证机构负责证明用户的身份并向用户签发数字证书，主要职责包括证书颁发、证书更新、证书废除、证书和证书撤销列表（Certificate Revocation List，CRL）的公布、证书状态的在线查询、证书认证和制定政策等。其中，证书颁发主要实现申请者在 CA 的注册机构（Registration Authority，RA）进行注册，申请数字证书。使用此数字证书，通过运用对称和非对称密码体制等密码技术建立一套严密的身份认证系统，从而保证：信息除了发送

方和接收方外不被其他人窃取；信息在传输过程中不被篡改；发送方能够通过数字证书来确定接收方的身份；发送方对于自己的信息不能抵赖等。X.509 数字证书就是其中一种被广泛使用的数字证书，是国际电信联盟-通信（International Telecommunication Union-Telecommunication，ITU-T）部分标准和国际标准化组织的证书格式标准。它是随 PKI 的形成而新发展起来的安全机制，支持身份的鉴别与识别（认证）、完整性、保密性及不可否认性等服务。通用的 X.509 数字证书的格式和吊销列表的格式如图 5-2 所示。

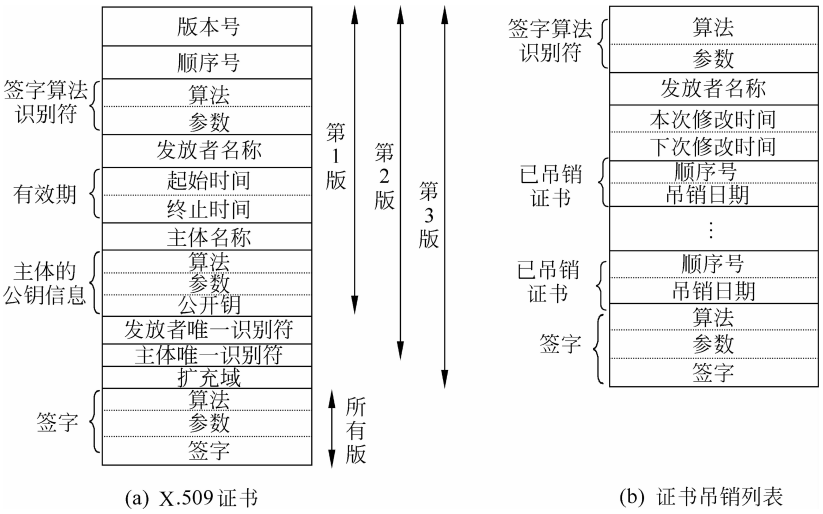

图 5-2　X.509 数字证书的格式和吊销列表的格式

X.509 证书标准文件数据格式如下。

- **版本号**——X.509 版本号，这将最终影响证书中包含的信息的类型和格式。
- **顺序号**——证书序列号是赋予证书的唯一整数值，它用于将本证书与同一 CA 颁发的证书区分开来。
- **签字算法识别符**——产生证书算法的识别符。
- **颁发者名称**——签发证书实体的唯一名，通常为某个 CA。
- **有效期**——证书仅在有限的时间段内有效。该域表示两个日期的序列，即证书有效期开始的日期及证书有效期结束的日期。
- **主体名称**——该域包含与存储在证书的主体公钥信息域的公钥相关联的实体的 DN。
- **主体公钥信息**——该域含有与主体相关联的公钥及该公钥用于何种密码算法的算法标识符。
- **颁发者唯一标识符**——可选域。它包含颁发者的唯一标识符。将该域包括在证书中的目的是为了处理某个颁发者的名字随时间的流逝而重用的可能。
- **主体唯一标识符**——可选域。它含有一个主体的唯一标识符。
- **扩充域**——扩展项提供了一种将用户或公钥与附加属性关联在一起的方法。
- **签字值**——该域中含有颁发证书的 CA 的数字签名。

证书作为各用户公钥的证明文件，必须由一个可信赖的 CA 用其密钥对各个用户的

公钥分别签署，并存放在X.500目录中供索取。所有CA都以层次结构存在。每个CA都有自己的公钥。这个公钥用该CA的证书签名后存放于更高一级CA所在服务器。但是，由于Root CA位于最顶端，没有上一级结点，故不受此限。若A想获得B的公钥，A可先在目录中查找ID_B，利用CA的公钥和Hash算法验证B的公钥证书的完整性，从而判断公钥是否正确。同样，公钥证书也存在一些缺陷，如公钥证书的签名都存放在其上一级机构所在的服务器中。在使用一个公钥证书前，用户不得不一级一级地核对有关的数字签名。但由于用户不能检查Root CA的公钥，因而不能确认Root CA是否被冒名顶替。

另外，用户在使用一个公钥之前，必须核对证书撤销列表CRL表，以确认该公钥是否作废。这样，即使CRL非常安全且高度可用，也难以满足数以百万计的用户的频繁访问。CRL很容易成为瓶颈，导致用户冒险使用一个未经核对的CRL表的公钥，给系统的安全性带来威胁。同时，用户私钥的管理也会带来问题，就是每个用户必须把私钥存放在计算机中，这也是一个不安全的因素。

5.4.2　基于X.509证书的双向认证过程

基于X.509证书的认证实际上是将个体之间的信任转化为个体对组织机构的信任，因此，这种认证系统需要有CA的支持。利用X.509数字证书可以实现相互实体身份的强认证功能，这里的“强”是指不是简单地使用口令，而是使用时间戳和基于随机数的挑战与应答。认证过程如图5-3所示。

图5-3　双向身份认证实现示意

(1) 用户A选取时间戳t_A，表示消息生成时间和期满时间，被用来防止信息传递的延迟以抗重放；生成一个非重复的随机数r_A(r_A用来抗重放攻击)及密钥k_A，A用自己的密钥k_A加密$\{t_A, r_A, B\}$即$\{t_A, r_A, B\}k_A$，并将它发送给B。

(2) B收到消息后执行以下动作：获取A的X.509证书，并验证证书的有效性，从A证书中提取A的公开密钥信息，验证A的身份是否属实，同时检验消息的完整性；检查B自己是否是消息的接收者；验证时间戳t_A是否为当前时间，检查r_A是否被重放；最后B生成一个非重复的随机数r_B(作用与r_A相同)，并向A发送消息$\{r_B, t_B, A, r_A\}k_B$。

(3) A收到消息后执行以下动作：获取B的X.509证书，并验证证书的有效性，接着从B的证书中提取B的公开密钥，验证B的公开密钥，验证B的身份，同时检验消息的完整性；检查A自己是否是消息的接收者；验证时间戳t_B是否为当前时间，并检查r_B是否被重放。

5.5 安全认证协议

下面将分别介绍几个有代表性的网络安全认证协议。

5.5.1 NSSK 认证协议

该协议是由 Roger Needham 和 Michael Schroeder 在 1978 年提出的著名的 Needham-Schroeder 认证协议。采用对称密钥算法的 NS 协议被称为 NSSK，采用非对称密钥算法的 NS 协议被称为 NSPK。NSSK 认证协议需要有一个称为鉴别服务器的可信权威机构参与密钥分发中心 KDC(key distribution center)，KDC 拥有每个用户的秘密密钥。若用户 A 欲与用户 B 通信，则用户 A 向鉴别服务器申请会话密钥。在会话密钥的分配过程中，双方身份得以鉴别。

首先来看一下 NSSK 的协议流程，如图 5-4 所示。

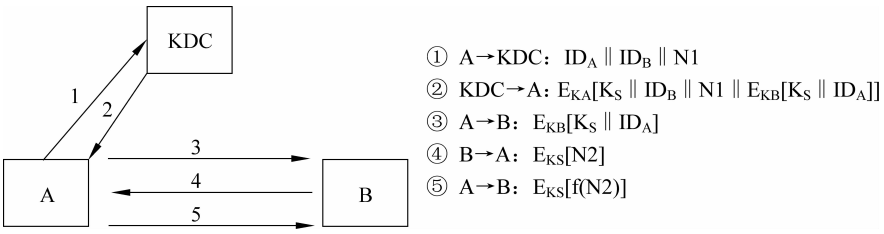

图 5-4 NSSK 的协议流程

其中 KDC 是密钥分发中心，Ra、Rb 是一次性随机数，保密密钥 Ka 和 Kb 分别是 A 和 KDC、B 和 KDC 之间共享的密钥，Ks 是由 KDC 分发的 A 与 B 的会话密钥，EX 表示使用密钥 X 加密。式中 K_A、K_B 分别是 A、B 与 KDC 共享的主密钥。协议的目的是由 KDC 为 A、B 安全地分配会话密钥 K_S，A 在第②步安全地获得了 K_S，而第③步的消息仅能被 B 解读，因此 B 在第③步安全地获得了 K_S，第④步中 B 向 A 示意自己已掌握 K_S，N2 用于向 A 询问自己在第③步收到的 K_S 是否为一新会话密钥，第⑤步 A 对 B 的询问做出应答，一方面表示自己已掌握 K_S，另一方面由 f(N2)回答了 K_S 的新鲜性。可见第④、⑤两步用于防止一种类型的重放攻击，比如敌手在前一次执行协议时截获第③步的消息，然后在这次执行协议时重放，如果双方没有第④、⑤两步的握手过程的话，B 就无法检查出自己得到的 K_S 是重放的旧密钥。

5.5.2 Kerberos 认证协议

1. Kerberos 认证系统简介及符号表示

Kerberos(注：Kerberos 是古希腊神话里的一条三头看门狗)是一种用于公共网络上的安全认证系统。Kerberos 身份认证系统是 MIT Athena(雅典娜)项目中的一部分，被 Windows 2000、UNIX 系统广泛采用。其 V1～V3 版是开发版本，V4 为 Kerberos 原型，获得了广泛的应用，V5 自 1989 年开始设计，于 1994 年成为 Internet 的标准(RFC