CHAPTER 3

第3章 数字签名与身份认证

计算机安全有四大安全原则：机密性、完整性、可认证性和不可抵赖。机密性是指保护消息内容不泄漏给非授权拥有此消息的人，即使是攻击者观测到了消息的格式，也无法从中提取消息的内容或得到任何有用的信息。实现机密性的最重要手段就是采用加密算法对消息进行加密，这是本书上一章的内容。完整性是指保证消息的内容没有受到任何非法修改、删除或替代。最常用的方法完整性保护是采用封装和签名，即用加密的方法或 Hash 函数产生一个明文的摘要附在传送消息上，作为验证消息完整性的依据，也称为完整性校验值。可认证性是最重要的安全性质之一，所有其他安全性质都依赖于此性质的实现，认证是对主体进行身份识别的过程。不可抵赖也称不可否认性，即用户不可否认敏感的消息或文件。不可否认性包含的内容很多，如接收方不可否认、发送方不可否认等。

3.1 安全协议

互联网已给经济、生活、军事等领域带来了巨大变革。互联网的出现和发展与 TCP/IP 协议族密切相关。与互联网迅速发展相随的是逐年增加的网络入侵事件，网络安全问题日益成为我们需要关注的焦点。影响网络安全的因素是多方面的，这里先从网络协议的设计引入进行探讨。

3.1.1 安全协议的定义

安全协议，有时也称为密码协议，是以密码学为基础的消息交换协议，此定义包含以下两层含义：①安全协议以密码学为基础，体现了安全协议与普通协议之间的差异；②安全协议也是通信协议，其目的是在网络环境中提供各种安全服务。密码学是网络安全的基础，但网络安全不能单纯依靠安全的密码算法。安全协议是网络安全的一个重要组成部分，需要通过安全协议进行实体之间的认证、在实体之间安全地分配密钥或其他各种秘密、确认发送和接收的消息的不可否认性等。

总之，安全协议是建立在密码体制基础上的一种交互通信协议，它运用密码算法和协议逻辑来实现认证和密钥分配等目标。

全面掌握安全协议定义，还需了解安全协议在运行环境中的角色。①协议参与者，即协议执行过程中的双方或多方，也就是发送方与接收方。协议的参与者可能是完全信任的人，也可能是攻击者，例如，认证协议：发起者/响应者；签名协议：签名申请者/签署人/验证人；零知识证明：证明者/验证者；电子商务协议：商家/银行/用户。②攻击者（或称敌手），即协议过程中企图破坏协议安全性的人。如被动攻击者主要是试图获取信息，主动攻击者则是为了达到欺骗、获取敏感信息、破坏协议完整性的目的。攻击者可能是合法的参与者，或是外部实体，也可能是协议的参与者。③可信第三方，即在完成协议的过程中，能帮助可信任的双方完成协议的值得信任的第三方。如仲裁者（用于解决协议过程中出现的纠纷）和密钥分发中心等。

3.1.2　安全协议的分类

按照协议完成的功能进行划分，最常用的安全协议主要有以下四类。

1. 密钥生成协议

密钥生成协议的目的是在通信的实体中建立共享的会话密钥，会话密钥通常使用对称密码算法对每一次单独的会话加密。密钥生成协议可采用对称密码体制或非对称密码体制建立会话密钥，可借助于一个可信的服务器为用户分发密钥，即密钥分发协议，可通过两个用户协商建立会话密钥。

2. 认证协议

认证是对数据、实体标识的认证。数据完整性可由数据来源认证保证。实体认证是确认某个实体的真实性的过程。认证协议主要用于防止假冒攻击。

3. 电子商务协议

电子商务就是利用电子信息技术进行各种商务活动。电子商务协议中主体往往代表交易的双方目标利益不太一致。因此电子商务协议最关注公平性，即协议应保证交易双方都不能通过损害对方利益得到不应该得到的利益。常见的电子商务协议有拍卖协议、SET 协议等。

4. 安全多方计算协议

安全多方计算协议的目的是保证分布式环境中各参与者以安全的方式来共同执行分布式的计算任务。分布式计算环境中不妨假定在执行过程中总会受到一个实体的攻击。安全多方计算协议的两个最基本的安全要求是保证协议的正确性和参与方私有输入的秘密性，即协议执行完后每个参与方都应得到正确的输出，并且除此之外不能获知其他任何信息（如数据库访问、联合签名等）。

根据参与者以及密码算法的使用情况进行分类，可以分为七类：无可信第三方的对称密钥协议、应用密码校验函数的认证协议、具有可信第三方的对称密钥协议、使用对称密钥的签名协议、使用对称密钥的重复认证协议、无可信第三方的公钥协议和有可信第三方的公钥协议。

3.2 数字签名

什么是数字签名呢？这是初学者首先需要明确的概念。需要声明的是：将手写的签名经过扫描仪扫描后再输入电脑中，这不是数字签名。数字签名基于非对称密码算法。我国于2004年8月28日第十届全国人民代表大会常务委员会第十一次会议通过了《中华人民共和国电子签名法》(简称《电子签名法》)，法律中对电子签名[①]的定义是：数据电文中以电子形式所含、所附用于识别签名人身份并表明签名人认可其中内容的数据。简言之，电子签名就是一串数据，该数据仅能由签名人生成，并且该数据能够表明签名人的身份。《电子签名法》明确规定：数据电文不得仅因为其是以电子、光学、磁或者类似手段生成、发送、接收或者储存的而被拒绝作为证据使用。现在，电子签名和传统文件中的手写签名具有同等的法律效应。

数字签名在网络安全、提供身份认证和不可否认性等方面有着重要意义。《电子签名法》的颁布对我国的电子商务进行规范和整顿具有重要推动作用。

1991年，美国国家标准和技术学会(NIST)发布了数字签名标准(Digital Signature Standard，DSS)并于1993年和1996年作了修订。在数字签名标准中采用的是数字签名算法(Digital Signature Algorithm，DSA)。虽然RSA同样具有数字签名的能力，但RSA的使用不是免费的。所以，NIST决定开发一个免费的数字签名算法(DSA)。DSA提出之后，受到了很多公司的抵制，原因是多方面的。首先，这些公司曾经在RSA上投入了大量人力和资金。其次，他们质疑DSA的安全性。虽然现在很多质疑都已得到回击，但并没有解决所有问题。

为什么要使用数字签名呢？可不可以用消息验证码来代替数字签名呢？答案是否定的。为了说明这个问题，我们考虑下面的情况：假设Alice和Bob正通过网络完成一笔交易，那么可能出现两种欺骗。

(1) Bob伪造一个消息并使用与Alice共享的密钥K产生该消息的认证码，然后声称该消息来自A。但实际上Alice并未发送任何消息。

(2) 既然Bob有可能伪造Alice发来的消息，那么Alice就可以对自己发过的消息予以否认。

这两种欺骗在实际应用中都有可能发生。双方争执不下而对簿公堂，但如果没有更有效的机制避免这种争端，法庭根本无法做出仲裁。

为了能够有效地解决这种冲突，数字签名必须具有以下特点。

(1) 发送方必须用自己独有的信息来签名以防止伪造和否认。

(2) 这种签名很容易产生。

(3) 对于接收方，应该很容易验证签名的真伪。

(4) 对于给定的x，找出$y(y\neq x)$使得签名$S(y)=S(x)$在计算上是不可行的。

(5) 找出任意两个不同的输入x，y，使得$S(y)=S(x)$在计算上是不可行的。

数字签名体制一般由两个部分组成：签名算法和验证算法。签名算法的密钥由签名方秘密保存，验证算法的密钥通常是公开的，以便他人验证签名的有效性。

---

① 电子签名在本书中同数字签名具有相同的概念。

数字签名可以分为两大类：直接数字签名和基于仲裁的数字签名。直接数字签名的实现很简单，仅仅涉及发送方和接收方两个通信实体。接收方需要了解发送方的公开密钥 K_{pub}。发送方使用其私钥 K_{pri} 对整个消息报文进行加密来生成数字签名。接收方收到整个报文之后用 K_{pub} 来解密。基于仲裁的数字签名需要可信第三方的参与。所谓可信第三方，是指所有通信方都可以信赖的一个通信实体。当需要数字签名时，发送方 Alice 发往接收方 Bob 的所有签名报文都首先送给仲裁。仲裁检验该报文及其签名的出处、内容，然后标注报文日期，并附加上一个仲裁的签名，最后由仲裁发给 Bob。

在特殊场合下，对签名算法有特殊的要求。比如 3.2.2 节所介绍的多重数字签名、3.2.3 节介绍的不可抵赖数字签名以及 3.2.4 节描述的盲签名等。

下面对一些常用的数字签名算法加以介绍。

3.2.1　数字签名算法

1. DSA 签名算法

DSA 签名算法利用单向 Hash 函数产生消息的一个 Hash 值，Hash 值连同一个随机数 r 一起作为签名函数的输入，签名函数还需使用发送方的秘密密钥和供所有用户使用的公开密钥。签名函数的两个输出 s 和 t 就构成了消息的签名(s,t)。接收方收到消息后再产生消息的 Hash 值，将 Hash 值与收到的签名一起输入验证函数，验证函数还需输入发送方的公开密钥。如果验证函数的输出与收到的签名成分 t 相等，则验证了签名是有效的。

下面简单介绍 DSA 签名算法的过程。在介绍 DSA 签名算法过程之前，必须交代 DSA 签名算法将会使用的参数。

全局公开密钥 p：p 是满足 $2^{L-1}<p<2^{L}$ 的大素数，其中 $512\leqslant L\leqslant 1024$ 且 L 是 64 的倍数。

q：q 是 $p-1$ 的素因子，满足 $2^{159}<p<2^{160}$，即 q 长为 160 比特。

g：$g=h^{(p-1)/q}\bmod p$，其中 h 是满足 $1<h<p-1$ 且使得 $g>1$ 的任一整数。

用户秘密密钥 x：x 是满足 $0<x<q$ 的随机数。

用户的公开密钥 y：$y\equiv g^{x}\bmod p$。

秘密随机数 r：r 满足 $0<r<q$。

DSA 签名算法过程如图 3-1 所示，具体计算如下。

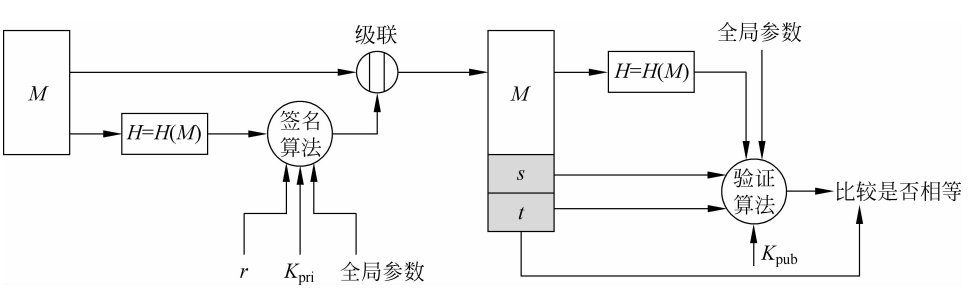

图 3-1　DSA 签名算法过程

(1) 签名方首先计算 $t\equiv(g^{r}\bmod p)\bmod q$。

(2) 利用 SHA 算法计算消息 M 的 Hash 值 $H(M)$。

(3) 计算 $s \equiv r^{-1}(H(M)+xt) \bmod q$

签名方完成以上过程后将三元组(M,s,t)作为自己的签名发送给接收方。接收方收到了签名(M_1,s,t),为了验证这个签名,所做计算如下。

(1) 计算 $w=s^{-1} \bmod q$。

(2) 计算 $u_1=[H(M_1)\cdot w] \bmod q$,$u_2=tw \bmod q$。

(3) 计算 $v=[(g^{u_1}y^{u_2}) \bmod p] \bmod q$,验证 v 是否和 t 相等。若相等,验证成功,若不等,则接收方有理由认为消息 M_1 是伪造的,拒绝接收该签名。

证明如下。

当且仅当 $M=M_1$ 时,下式成立:

$$v \equiv [(g^{H(M_1)w}g^{-xtw}) \bmod p] \bmod q \equiv [g^{(H(M_1)+xt)s^{-1}} \bmod p] \bmod q$$
$$\equiv (g^{r's's^{-1}} \bmod p) \bmod q \equiv (g^r \bmod p) \bmod q = t$$

DSA 签名算法的安全性是基于求解离散对数的困难性。

2. RSA 签名算法

在理解了上一章非对称密码算法(RSA)的基础上,来学习 RSA 签名算法就非常简单了。签名方产生了一对公开密钥(n,e)和私有密钥(n,d)之后,就可以对消息进行签名。RSA 算法的签名过程如图 3-2 所示。

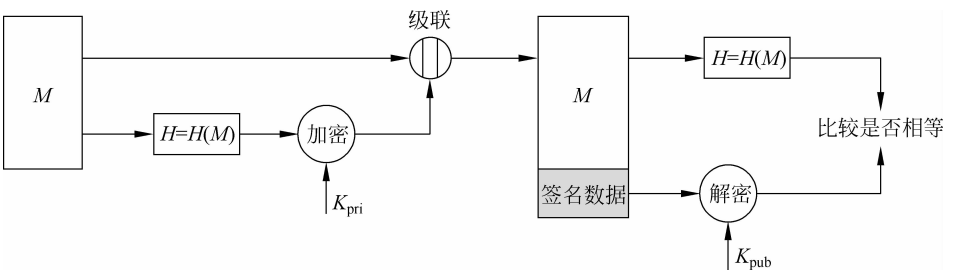

图 3-2　RSA 签名算法过程

签名过程如下。

(1) 利用摘要算法计算消息的摘要 $H(M)$。

(2) 用私有密钥(n,d)加密消息摘要得到 $s=H(M)^d \bmod n$。

签名完成之后,签名方将(M,s)发送给接收方。接收方收到签名(M_1,s)之后通过以下步骤进行验证。

(1) 利用摘要算法计算消息的摘要 $H(M_1)$。

(2) 用公开密钥(n,e)加密消息摘要得到 $s_1=s^e \bmod n$。如果 $s_1=H(M_1)$,那么验证通过,否则拒绝接收该签名。因为当且仅当 $M=M_1$ 时,$s_1=H(M_1)$。

RSA 签名算法的安全性是建立在大数分解问题的困难性之上。

3. Elgamal 签名算法

在上一章曾经介绍 Elgamal 非对称密码算法,这里将介绍 Elgamal 签名算法。Elgamal 签名算法由 T. Elgamal 于 1985 年提出。它的安全性基于求解离散对数的困难性。Elgamal 签名算法的修正版已被美国 NIST 作为数字签名的标准,即前面我们所介绍的

DSA 算法。

设 p 是一个大素数；g 是 z_p^* 的一个生成元；x 是签名方的秘密密钥，$x \in z_p^*$；y 是用户 A 的公开密钥，$y=g^x \bmod p$。则 Elgamal 签名算法的过程如下。

对于待签名的消息 M，签名方执行以下步骤。

(1) 计算 M 的 Hash 值 $H(M)$。

(2) 选择随机数 $r \in z_p^*$，计算 $t=g^r \bmod p$。

(3) 计算 $s=[H(M)-xt]r^{-1} \bmod (p-1)$。

然后签名方将以 (M,s,t) 作为产生的数字签名发送给接收方。接收方在收到消息 M_1 和数字签名 (s,t) 后，先计算 $H(M_1)$，并按下式验证：$y^t t^s \equiv g^{H(M_1)} \pmod p$。

如果 $y^t t^s$ 与 $g^{H(M_1)} \pmod p$ 相等，则验证通过，否则拒绝接收该签名。验证的正确性可以通过下式证明。当且仅当 $M=M_1$ 时，

$$y^t t^s \equiv g^{tx} g^{rs} \equiv g^{tx+H(M)-xt} \equiv g^{H(M)} \pmod p \equiv g^{H(M_1)} \pmod p$$

4. 其他数字签名方案

除了以上介绍的三种最常见的数字签名方案以外，还有很多其他的数字签名算法，比如 Rabin 签名算法、Schnorr 签名算法、GOST 签名算法、ESIGN 签名算法、Okamoto 签名算法、OSS 签名算法等。其中，Rabin 签名算法的安全性是建立在大数分解这一难题之上；而 Elgamal、Schnorr、DSA、GOST、ESIGN、Okamoto 等签名算法的安全性是基于有限域上求解离散对数的困难性。它们统称为离散对数签名体制。GOST 签名算法是俄罗斯采用的数字签名标准，ESIGN 签名算法、Okamoto 签名算法是由日本 NTT 的 T. Okamoto 等设计的。OSS 签名算法于 1984 年由 Ong、Schnorr、Shamir 等联合发表。

关于这些数字签名方案，读者可以阅读其他参考书。

3.2.2　多重数字签名

在很多情况下，需要两个或多个人同时对一份文件进行签名，例如共同签署协议的 Alice、Bob、Carlos 甚至更多人。为描述方便起见，不妨假设 Alice 和 Bob 需要同时对文件进行签名。那么，签名方案有三种。

(1) Alice 和 Bob 分别对文件的副本签名，如图 3-3(a)所示。

(2) Alice 首先对文件进行签名，然后 Bob 对 Alice 的签名再进行签名，如图 3-3(b)所示。

(3) 利用单向 Hash 函数实现多重签名，如图 3-3(c)所示。

前两种多重签名方案有明显的缺陷：如果 Alice 和 Bob 分别对文件的副本进行签名，那么签名的消息是原文的两倍；第二种方案使得验证签名出现困难。在不验证外层 Bob 签名的情况下验证内层 Alice 的签名是不可能的。

但如果采用单向散列函数，实现多重签名则很简单。具体步骤如下。

(1) Alice 对文件的 Hash 值签名。

(2) Bob 对文件的 Hash 值签名。

(3) Bob 将他的签名交给 Alice。

(4) Alice 把文件、她的签名和 Bob 的签名一起发送给 Carlos。

(5) Carlos 分别验证 Alice 和 Bob 的签名。

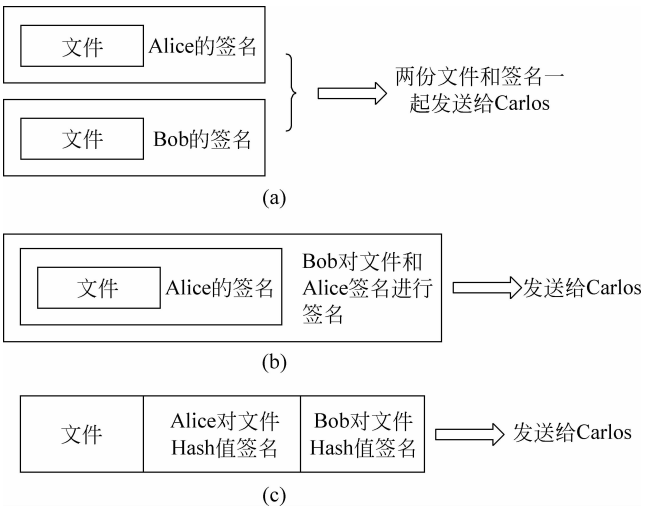

图 3-3　多重签名方案

在采用单向 Hash 函数的多重签名方案中，Alice 和 Bob 能同时或顺序地完成第(1)、(2)步的签名。而 Carlos 可以只验证其中一人的签名而不用验证另一人的签名。

以上例子说明的是两重签名的情况，很容易将之推广得到多重签名方案。

3.2.3　不可抵赖数字签名

到目前为止，我们所介绍的数字签名都有这样一个共同点：签名方用自己的私钥进行数字签名，验证方用签名方的公钥进行验证，而签名方的公钥是公开的，因此，任何人都可以对这份文件的真实性进行验证。这在一些场合下是非常有用的，比如发布信息公告。但有些时候人们又想尽量避免这种情况，比如在私人信件中：不怀好意的朋友总是可以把你的私人信件公布于众，并且任何人都可以验证。于是签名方希望有这样一种数字签名方案：在没有签名方的同意下，接收方不能把签名给第三者看。不可抵赖数字签名正是这样一种方案。

不可抵赖数字签名的思想非常简单，大致过程如下所述。

(1) Alice 向 Bob 出示一个签名。

(2) Bob 产生一个随机数并送给 Alice。

(3) Alice 利用随机数和其私人密钥进行计算，并将计算结果发送给 Bob。Alice 只能计算该签名是否有效。

(4) Bob 确认这个结果。

Bob 不能让第三者 Carlos 确信 Alice 的签名是有效的，因为 Carlos 不知道 Bob 的随机数。Carlos 只有在他与 Alice 本人完成这个协议后才能确信 Alice 的签名是有效的。

最先由 David Chaum 提出了不可抵赖数字签名的具体算法。算法原理如下：首先公布一个大素数 p 和它的原根 g。Alice 拥有私钥 x 并公布公钥 y，其中 $y=g^x \bmod p$。当需要签名时，Alice 计算 $z=m^x \bmod p$。

Bob 对数字签名的验证过程如下。

(1) Bob 选择两个小于 p 的随机数 a 和 b，计算 $c=z^a(g^x)^b \bmod p$，并发送给 Alice。

(2) Alice 计算 $t=x^{-1} \bmod q$，$d=c^t \bmod p$，并把 d 发送给 Bob。

(3) Bob 进一步确认：$d \equiv m^a g^b \pmod p$。如果该式成立，则 Bob 就认为签名是真实的，否则拒绝该签名。

正确性证明：

$$d = c^t \bmod p = (z^a g^{xb} \bmod p)^t \bmod p = (z^{at} g^{xbt}) \bmod p = (m^{axt} g^{btx}) \bmod p$$

由于 $t=x^{-1} \bmod (p-1)$，所以 $tx=1 \bmod (p-1)$，即 $tx=k\varphi(p)+1$，其中，$\varphi(p)$ 为欧拉函数，因为 p 是素数，所以 $\varphi(p)=p-1$。

根据欧拉定理得到

$$\begin{aligned} d &= (m^{axt} g^{btx}) \bmod p = (m^{a[k\varphi(p)+1]} g^{b[k\varphi(p)+1]}) \bmod p \\ &= [(m^a g^b) \bmod p][m^{ak\varphi(p)} g^{bk\varphi(p)} \bmod p] \\ &= [(m^a g^b) \bmod p][(m^{ak} g^{bk})^{\varphi(p)} \bmod p] \\ &= [(m^a g^b) \bmod p] \cdot 1 \\ &= (m^a g^b) \bmod p \end{aligned}$$

如果 Bob 将 Alice 的签名给 Carlos 看，希望 Carlos 相信这是 Alice 的签名。但同时，如果 David 打算使 Carlos 相信这是 Eva 的签名，那么他可以伪造该协议的副本，首先伪造第(1)步的消息，然后作第(3)步的计算产生 d，再伪造第(2)步的消息，称这个 d 来自 Eva。对于 Carlos 来说，Bob 和 David 给他的副本是相同的，他不能相信签名的有效性，除非他亲自和 Alice 完成这个协议。

3.2.4　盲签名

通常情况下，签名方需要了解他所要签署的文件的内容，否则他不会轻易签名。但在一些特殊的场合，Alice 可能需要这样一种机制：希望 Bob 对一份文件签名，但又不想他知道文件的内容。这就是将要讨论的盲签名，盲签名的应用并不广泛，只是在特定的领域有着重要的用途，比如在新型电子商务中。

完全盲签名并不真正实用，因为 Alice 可以让 Bob 签署任意一份对她有利的文件，比如“Bob 欠 Alice 100 万美元”……

实际应用中都采用“分割-选择”技术，可使 Bob 知道他所签署文件的大体内容，但是并不准确。

利用 RSA 算法，D. Chaum 最初于 1985 年提出了第一个盲签名算法。该算法原理如下。

假设签名方 Bob 的公钥为(n,e)，私钥为(n,d)：

(1) 若 Alice 需要 Bob 对消息 m 进行盲签名，她首先选取 $1<k<m$，并计算 $t \equiv mk^e \bmod n$，将计算结果发送给 Bob。

(2) Bob 对 t 进行盲签名，计算 $t^d \equiv (mk^e)^d \bmod n$，并将计算结果发送给 Alice。

(3) Alice 计算 $S \equiv t^d/k \bmod n$，得到 $S \equiv m^d \bmod n$。这就是 Bob 对 m 的盲签名。

正确性证明：

因为

$$t^d \equiv (mk^e)^d \bmod n \equiv m^d k \bmod n$$

所以

$$\frac{t^d}{k} \bmod n \equiv \frac{m^d k}{k} \bmod n = m^d \bmod n$$

有关盲签名的更深入探讨，请查阅相关书籍。

3.2.5 群签名

群签名，即群数字签名。群签名是在 1991 年由 Chaum 和 van Heyst 提出的一个签名概念。Camenish、Stadler、Tsudik 等对这个概念进行了修改和完善。群签名在管理、军事、政治及经济等多个方面有着广泛的应用。

所谓群签名就是满足这样要求的签名：在一个群签名方案中，一个群体中的任意一个成员可以以匿名的方式代表整个群体对消息进行签名。与其他数字签名一样，群签名是可以公开验证的，而且可以只用单个群公钥来验证。

一个群签名是一个包含下面过程的数字签名方案。

(1) 设置：一个用以产生群公钥和群管理者私钥的概率多项式时间算法。

(2) 加入：一个用户和群管理员之间的使用户成为群管理员的交互式协议。执行该协议可以产生群成员的私钥和成员证书，并使群管理员得到群成员的私有密钥。

(3) 签名：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：输入为(消息，对消息的签名，群公钥)一个算法，在群公钥的作用下，确定签名的有效性。

(5) 打开：一个在给定一个签名及群私钥的条件下确认签名人的合法身份的算法。

一个好的群签名方案一般具有以下特点。

(1) 正确性：合法成员发出的签名能通过验证。

(2) 匿名性：给定一个群签名后，除了群管理员之外，确定签名人的身份在计算上是不可行的。

(3) 不可伪造性：只有获得群成员证书和签名密钥的群成员才能够生成合法的群签名。

(4) 不可关联性：在不打开群签名的情况下，确定两个不同的签名是否为同一个成员签署在计算上不可行。

(5) 可跟踪性：群管理员在必要的时候可以打开一个签名，确认签名者的身份。

(6) 抗合谋攻击型：即使一些群成员串通在一起也不能产生其他人的合法签名或不可跟踪的合法签名。

(7) 防陷害攻击：包括群管理员在内的任何人都无法以其他成员身份产生合法的群签名。

如群管理者计算通常的签名方案的密钥对$(\mathrm{sig_m}, \mathrm{ver_m})$，计算概率公开密钥密钥加密方案的密钥对$(\mathrm{encr_m}, \mathrm{decr_m})$，公布两个公开密钥作为群公开密钥。Alice 以如下方式加入该群：选择一个秘密的随机密钥 x 并计算成员密钥 $z=f(x)$，f 是单向函数。Alice 发送 z 给管理者，管理者返回一个成员证书 $v=\mathrm{sig_m}(z)$。Alice 的群私钥是元组(x,z,v)。

为了代表群对消息 m 签名，Alice 使用群管理者的加密密钥 $\mathrm{encr_m}$ 对(m,z)加密，即 $d=\mathrm{encr_m}(r,(m,z))$，$r$ 是充分大的随机串。Alice 计算一个非交互最小泄露证明 p，证明她知道 x_1，v_1 和 r_1 满足如下方程：$d=\mathrm{encr_m}(r_1,(m,f(x_1)))$和 $\mathrm{ver_m}(v_1,f(x_1))=\mathrm{correct}$。

Alice 对消息 m 的签名是(d,p)，签名可通过检查证明 p 进行验证。若要打开此签名，群管理者解密密文 d 得到成员密钥 z，从而揭示 Alice 的身份。

3.3　消息认证和身份认证

认证分为两种，一是对消息的认证，二是对身份的认证。在本节中，将集中阐述如何对消息进行认证。在下一章将集中说明如何对身份进行认证。

所谓消息认证，是指接收方对收到的消息进行检验，检验内容包括消息的源地址、目的地址，消息的内容是否受到篡改以及消息的有效生存时间等。消息认证可以是实时的，也可以是非实时的。例如，网上银行对消息的认证属于实时认证，而电子邮件中对消息的认证属于非实时认证。

所谓身份认证，是指系统对网络主体进行验证的过程，用户必须向系统证明其身份。系统验证通信实体的一个或多个参数的真实性和有效性，以判断他的身份是否和他所声称的身份一致。相当于现实生活中通过检验身份证来识别身份。

3.3.1　消息认证的概念

消息认证离不开 Hash 函数，因此本部分首先介绍 Hash 函数。

在密码学书籍中经常可以见到单向函数、Hash 函数、单向 Hash 函数以及单向陷门函数等概念。这些概念是消息认证、身份认证、非对称密码算法以及数字签名的基石。

1. 单向函数

顾名思义，单向函数的计算是不可逆的。即已知 x 要计算 y 使得 $y=f(x)$ 很容易；但若已知 y 要计算 x，使得 $x=f^{-1}(y)$ 就很困难。在这里，“困难”的程度定义为：即便使用世界上所有的计算机一起计算 x 值，都需要花数百万年甚至更长的时间。

生活中类似单向函数的例子很多，比如拼图。要把一个完整的拼图拆开并打乱是一件很简单的事情，但是如果要把这些混乱的小图案拼成一张完整的图却需要很长的时间。

2. Hash 函数

在计算机科学领域中，Hash 函数是使用得最广泛的概念之一。它把可变长度的输入字符串映射为固定长度的输出字符串。输出字符串被称为“Hash 值”。如果对于不同的输入经过 Hash 映射后却得到了相同的 Hash 值，我们就称该 Hash 函数产生了冲突。一个好的 Hash 函数应该是无冲突的。Hash 函数是公开的，任何人都可以看到它的处理过程。Hash 函数不一定是单向函数，对于那些既是 Hash 函数，也是单向函数的映射，我们称为单向 Hash 函数，用 $H(\cdot)$ 表示。

单向 Hash 函数既有单向函数的特点，也有 Hash 函数的特点。对于信息 M，要计算 $H(M)=m$ 很简单，但若给定 Hash 值 m，要找出其相对应的原文 M 是相当困难的。

3. 单向陷门函数

单向陷门函数是一类特殊的单向函数，它包含一个秘密陷门。在不知道该秘密陷门的情况下，计算函数的逆是非常困难的。但若知道该秘密陷门，那么计算函数的逆就非常简单。数学上对单向陷门函数的严格定义是：把数论函数 $f(n)$ 称为单向陷门函数，如果它满足下面三个条件：

(1) 对 $f(n)$ 的定义域中的每一个 n，均存在函数 $f^{-1}(n)$，使得

$$f^{-1}(f(n)) = f(f^{-1}(n)) = n$$

(2) $f(n)$与 $f^{-1}(n)$都很容易计算；

(3) 仅根据已知的计算 $f(n)$的算法，去找出计算 $f^{-1}(n)$的容易算法是非常困难的。

仍以拼图为例，如果不知道拼图各个碎片的编号，要将混乱的碎片拼成完整的图是困难的；但如果知道碎片的编号，要完成整个工作就相当简单。

单向陷门函数是非对称密码学的基础。非对称密码算法如 RSA、ECC 等利用了单向陷门函数的性质。而单向函数和单向 Hash 函数却不能用做加密，为什么呢？因为计算单向函数的逆是非常困难的，如果利用单向函数加密，任何人都不可能进行解密。单向 Hash 函数的作用在于消息认证。

接下来看一下消息认证系统的概念。消息认证的基本方法有两种，一是采用 Hash 函数，二是采用消息认证码(Message Authentication Code，MAC)。这两种方法的区别在于是否需要密钥的参与。消息认证码是指消息被一密钥控制的公开函数作用后产生的固定长度的、用做认证符的数值。为了能够对消息进行认证，通信双方 Alice 和 Bob 需要共享一密钥 K。设 Alice 欲发送给 Bob 的消息是 M，Alice 首先计算 MAC$=f(K,M)$，其中函数 $f(\cdot)$是有密钥控制的公开函数，然后将原消息和 MAC 值级联在一起，再向 Bob 发送出去。Bob 收到消息后做与 Alice 相同的计算，求得一新 MAC，并与收到的 MAC 做比较。由于除了 Alice 和 Bob 之外，没有其他人知道密钥 K。因此，如果比较结果相等，则 Bob 有理由认为该消息确由 Alice 所发，而且传输过程中没有经过任何修改；如果比较结果不相等，那么 Bob 拒绝接收该消息。Bob 可以认为要么该消息是伪造的，要么该消息在传输过程中出了差错。MAC 的认证方式如图 3-4 所示。

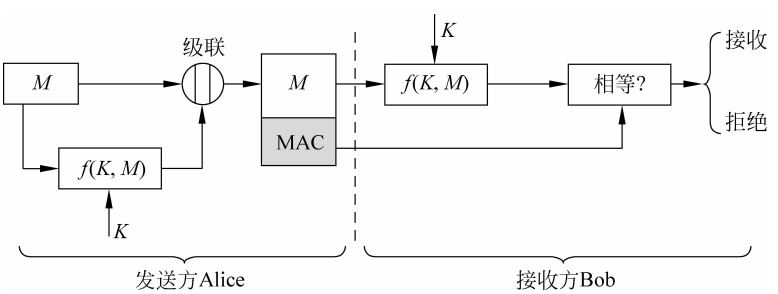

图 3-4　MAC 认证方式示意

细心的读者不难发现，Alice 和 Bob 的通信内容在网上是以明文形式传输的。因此攻击者也能够看到完整的消息内容。在一些情况下，Alice 和 Bob 不希望第三者能够解读消息，因此 Alice 需要对消息 M 进行加密。加密的方式有两种：其一，Alice 首先将 M 进行加密得到密文 C，然后再计算密文 C 的 MAC，最后将 C 和 MAC 级联在一起发送给 Bob，如图 3-5 所示；其二，Alice 计算出 M 的 MAC 值之后，将 M 和 MAC 级联，对整个消息进行加密，然后发送给 Bob，如图 3-6 所示。一般情况下，Bob 更希望对接收到的整个消息进行认证，因此，第二种方式更为常用。

用 Hash 函数进行消息认证则不需要密钥的参与。Hash 函数是一个公开的函数，它可以将任意长的消息 M 映射为一个较小固定长度值 $H(M)$，我们称函数值 $H(M)$为 Hash 值(散列值)、Hash 码(散列码)或 Hash 摘要。Hash 码是消息中所有比特的函数，任意一个比

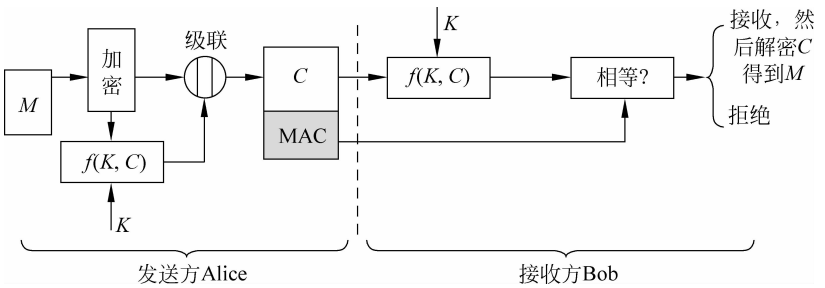

图 3-5　MAC 认证方式示意——只加密 M

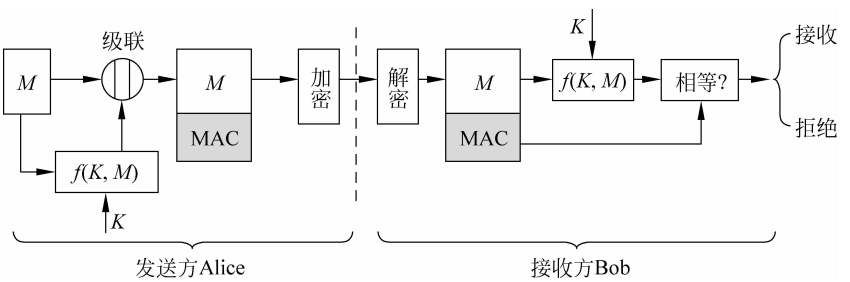

图 3-6　MAC 认证方式示意——加密整个消息

特的改变都将会使 Hash 码发生巨大改变。因此，利用 Hash 函数可以检测消息传播过程中是否遭受篡改。因此，单从这一点上看，Hash 函数的功能类似于循环冗余校验码(CRC)和奇偶校验码。

跟消息认证码一样，Hash 函数也有多种使用方式：可以只进行 Hash 摘要，也可以配合加密算法一起使用。参照图 3-4、图 3-5 和图 3-6，可以得到图 3-7～图 3-9 所示的 Hash 函数使用方式。除此之外，Hash 函数如果配合非对称密钥加密算法一起使用，就提供了数字签名的功能。

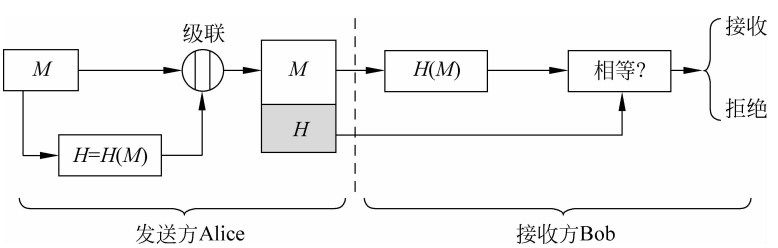

图 3-7　Hash 函数使用方式示意

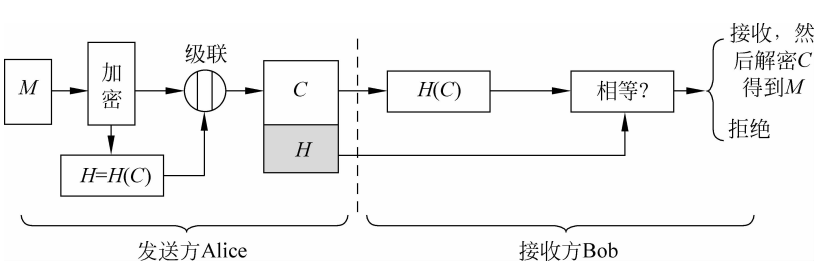

图 3-8　Hash 函数使用方式示意——只加密 M

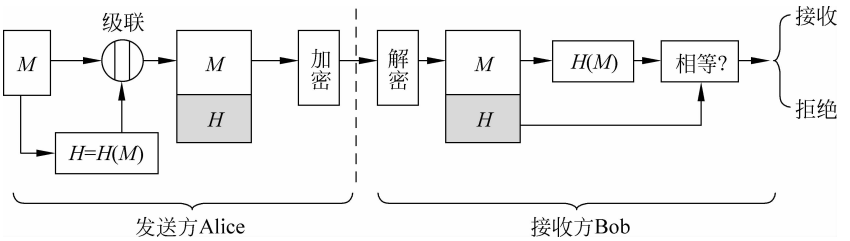

图 3-9　Hash 函数使用方式示意——加密整个消息

Hash 函数相当于为需要认证的数据产生一个“数字指纹”。为了能够实现对数据的认证，Hash 函数应满足以下条件。

(1) 函数的输入可以是任意长。

(2) 函数的输出是固定长。

(3) 已知 x，求 $H(x)$较为容易，可用硬件或软件实现。

(4) 已知 h，求使得 $H(x)=h$ 的 x 在计算上是不可行的，这一性质称为函数的单向性，称 $H(x)$为单向 Hash 函数。

(5) 对于给定的 x，找出 $y(y\neq x)$使得 $H(y)=H(x)$在计算上是不可行的。如果单向 Hash 函数满足这一性质，则称其为弱无碰撞。

(6) 找出任意两个不同的输入 x，y，使得 $H(y)=H(x)$在计算上是不可行的。如果单向 Hash 函数满足这一性质，则称其为强无碰撞。

了解了消息认证的两种基本方法，下面介绍具体的消息认证算法。

3.3.2　消息认证算法

1. MD5 算法

MD5 是密码学家 Ron Rivest 提出来的算法，MD5 根源于一系列消息摘要算法，从最初很脆弱的 MD 到现在广泛使用的 MD5（包括未发表的 MD3），Ron Rivest 做出了卓越的贡献。

MD5 是一种速度非常快的消息摘要算法。MD5 的输入可以是任意长，以 512 位为单位分成块，输出是 128 位的消息摘要。

大体上说，MD5 算法总共分为以下五步。

1) 填充字节

在原消息中增加填充位，增加的长度在 1～512 之间。从而使填充后的消息长度等于一个值，该值比 512 的倍数少 64。例如，原消息长度为 100 位，那么应该填充 348 位，因为 100＋348＝512－64；如果原消息长度为 448，那么应该填充 512 位，因为 448＋512＝1024－64。

为什么填充后的消息长度要比 512 的倍数少 64 呢？这是因为这 64 位是用来记录原消息长度的。如果原消息长度大于 2^{64}，那么就取其低 64 位值。填充后的消息长度加上 64 位长度值，使得整个消息的长度变成了 512 的倍数。

2) 分块

由于整个消息的长度为 512 的倍数，所以可以将消息分成很多块，每块长 512 位，每一块由 16 个 32 比特的字构成。

3）初始化寄存器

MD5 缓冲区初始化算法要使用 128 比特长的缓冲区以存储中间结果和最终 Hash 值，缓冲区用 4 个 32 比特长的寄存器 A，B，C，D 构成。每个寄存器的初始十六进制值分别为 A＝0x01234567，B＝0x89ABCDEF，C＝0xFEDCBA98，D＝0x76543210。

4）处理每一个分块

每一个分块都由压缩函数 $H(\cdot)$处理，压缩函数是算法的核心，它又有 4 轮处理，每一轮 16 步，总共 64 步。

5）输出结果

每一个分块都被处理之后，最后压缩函数的输出即为产生的消息摘要。

MD5 算法的整个过程如图 3-10 所示。

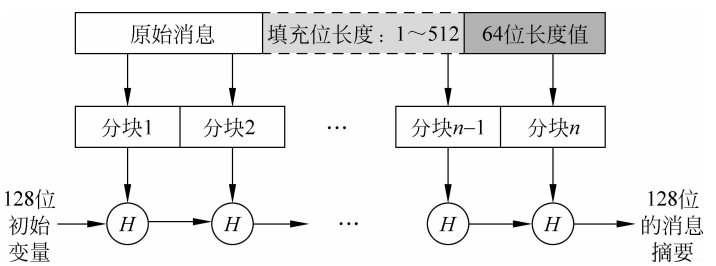

图 3-10　MD5 算法过程示意

2. SHA 算法

SHA（Secure Hash Algorithm，安全 Hash 算法）算法最初是 1993 年由美国国家标准与技术学会 NIST 和 NSA 联合发布的。1995 年作了一些修订，后来正式更名为 SHA-1。

SHA 的输入长度没有下限，但上限是 2^{64}。这和 MD5 算法不一样，MD5 算法没有规定输入消息长度上限。SHA 算法的输出为 160 位的消息摘要，比 MD5 的输出长 32 位。

同 MD5 一样，SHA 是在 MD4 的基础上修改而成的。因此，SHA 和 MD5 有很多的相同点。SHA 大体也是由五步构成。

1）填充

填充的方法和 MD5 一样，使得填充后的消息长度比 512 的倍数少 64。

2）分块

将整个消息分成 512 位的块，这些块就是消息摘要处理的逻辑输入。

3）初始化寄存器

SHA 缓冲区初始化算法要使用 160 比特长的缓冲区以存储中间结果和最终 Hash 值，缓冲区用 5 个 32 比特长的寄存器 A，B，C，D，E 构成。每个寄存器的初始十六进制值分别为 A＝0x01234567，B＝0x89ABCDEF，C＝0xFEDCBA98，D＝0x76543210，E＝0xF0E1D2C3。

4）处理每一个分块

处理的过程和 MD5 算法类似，不同的是，每轮的压缩函数有 20 步迭代，因此总共 80 步迭代。

5）输出结果

每一个分块都被处理之后，最后压缩函数的输出即为产生的消息摘要。

SHA-1 算法的整个过程如图 3-11 所示。

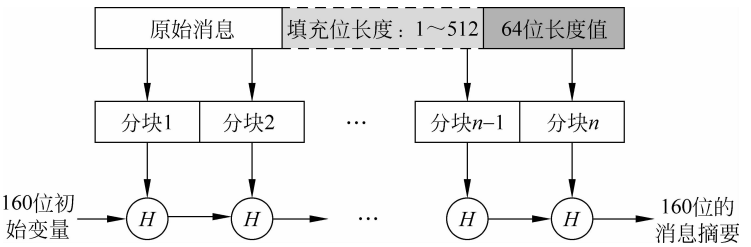

图 3-11　SHA-1 算法过程示意

3. HMAC 算法

HMAC 是指基于散列的消息认证码（Hash-based Message Authentication Code）。在 Internet 上广泛使用的安全套接层协议（Secure Socket Layer，SSL）就使用了 HMAC 算法。

HMAC 的思想是继续利用现有的消息摘要算法而没有必要重新开发一个新算法。HMAC 用共享的密钥加密消息摘要，从而输出 MAC。同 MD5、SHA-1 一样，HMAC 也是一个分块算法，每块长通常为 512 位。

下面详细说明 HMAC 的 7 步操作过程。

1）密钥变换

密钥变换的目的是使密钥 k 的长度和消息块的长度 l 相等。如果 $k<l$，那么在 k 的左端添加 0，使得 k 和 l 的总长度相等；若 $k>l$，则通过相应的消息摘要算法得到密钥 k'，然后使得 k' 和 l 的长度相等。

2）异或

将第一步的输出 k 作为输入，与 ipad 异或得到输出 S_1。其中 ipad 是十六进制字符串 36 的重复，即 ipad＝0x363636…，其长度与 l 的长度相等。

3）消息级联

将 S_1 与原消息 M 级联，S_1 在前，M 在后。

4）计算消息摘要

采用一定的消息摘要算法，如 MD5 或 SHA-1，计算上一步输出的消息摘要 H。

5）异或

将第一步的输出 k 作为输入，与 opad 异或得到输出 S_2。其中 opad 是十六进制数 5A 的重复，即 opad＝0x5A5A5A…，其长度与 l 的长度相等。

6）消息级联

将 S_2 与第四步计算出来的消息摘要 H 级联，S_2 在前，H 在后。

7）输出最终结果

对步骤 6）的输出采用相应的消息摘要算法，得到最终的输出结果。

HMAC 的完整过程如图 3-12 所示。

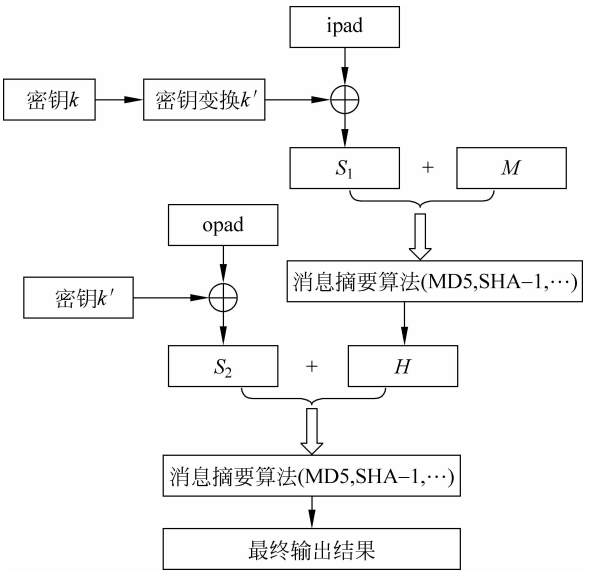

图 3-12　HMAC 完整过程

3.3.3　身份认证的概念

身份认证的目的是在不可信的网络上建立通信实体之间的信任关系。在网络安全中，身份认证的地位非常重要，它是最基本的安全服务之一，也是信息安全的第一道防线。在具有安全机制的系统中，任何一个想要访问系统资源的人都必须首先向系统证实自己的合法身份，然后才能得到相应的权限。这好比生活中我们要进入工作大楼，必须首先向大楼保安人员出示工作证，否则将会被拦截在外。

实际网络攻击中，黑客往往首先瞄准的就是身份认证：试图通过冒名顶替的手段来获取一些非法财物或信息。这类似于生活中的不法分子经常伪造身份证来获得检查人员的信任。

目前，计算机系统采取的身份认证方法有很多，比如口令认证、智能卡认证、基于生物特征的认证、双因素认证、基于源地址的认证、数字证书和安全协议等。

一个成熟的身份认证系统应该具有以下特征。

(1) 验证者正确识别对方的概率极大。

(2) 攻击者伪装以骗取信任的成功率极小。

(3) 通过重放攻击进行欺骗和伪装的成功率极小。

(4) 实现身份认证的算法计算量足够小。

(5) 实现身份认证所需的通信量足够小。

(6) 秘密参数能够安全存储。

(7) 对可信第三方的无条件信任。

(8) 可证明安全性。

身份认证的方法有以下几种。

1. 口令的认证

基于口令的认证是指系统通过用户输入的用户名和密码来确立用户身份的一种机制。基于口令的身份认证是最常见的也是最简单的一种身份认证机制，例如，电子邮箱、论坛账号等都是通过口令来确认对方的身份。这种机制曾经发挥过巨大的作用，但是随着时间的推移，在企业环境中基于口令的认证机制并不是最方便最有效的。主要原因有以下几点。

首先，口令要多复杂？通常，用户创建口令时总是选择便于记忆的简单口令，比如以电话号码、生日、门牌号等作口令。但这样一来别人也容易猜到；若选择复杂的口令，则用户自己也可能忘记。这个矛盾因素严重影响到口令的强度和验证效率。

其次，在许多场合，输入的口令很容易泄漏，只需通过键盘上的手势就大致能猜出来。甚至一些恶意程序诸如特洛伊木马程序可以记录用户输入的口令，然后秘密地通过网络发送出去。

再次，口令传输不安全。一旦用户输入口令后，如何传送给系统或验证服务器？很多口令在网络上都是以明文形式传输，比如电子邮箱的口令。一旦攻击者截获该口令，他就可以肆无忌惮地对用户的邮箱做任何事情。

最后，口令存储也不安全。口令在系统中是怎样被存储的？在过去许多软件工具采用简单的加密存储，但它们一般采用强度不高的加密或允许从系统外获得文件。一些简单的强行破解程序很容易解密。许多口令破解程序已经被开发出来。在这些软件的帮助下，可以很方便地破解 UNIX、Windows NT、Windows XP 的用户口令和缓存口令。其他一些程序也可以很轻松地从浏览器或应用中获取口令。

为了提高口令的安全性，人们提出了口令生命周期的概念，用以强迫用户经常更换口令。但是强度不高的口令仍然强度不高，而且某些用户经常使用以前用过的口令。也有些用户经常加一些数字到同一个口令之前或之后。

如何提高口令的安全性呢？专家给出了以下几点建议：提高口令长度；禁止用户名和其他可能的选择；强迫用户经常更换口令；对安全系统中的用户进行培训；审计口令更换情况和用户的登录情况；建立定期检查审计日志的习惯；在用户 N（N 值是系统预先设定的）次登录不成功后自动锁定账户；不允许对口令文件的随便访问；限制用户更改验证系统的方法。

值得注意的是，即使满足了以上所有的要求，这种努力也只能改进口令使用的安全期长短而已。

既然基于口令机制的身份认证很不安全，那么它还有没有存在的必要呢？答案是肯定的。主要有以下两方面原因。首先，口令是经济有效的安全机制之一，同指纹认证、虹膜认证、人脸认证等基于生物特征的认证机制相比，口令认证机制具有数据量小。速度快等优点；其次，口令是非常简单和易于使用的，大多用户都能接受，实施起来简单方便。

2. 智能卡认证

比口令认证方式稍微安全的认证方式是智能卡认证。智能卡（smart card）是当今信用卡领域的新产品。所谓智能卡，实际上就是在信用卡上安装一个微型电脑芯片，这个芯片包含了持卡人的各种信息。这种芯片与传统的磁条卡相比，不易伪造，因而具有更高的防伪能力和安全性。自 20 世纪 70 年代末，智能卡在法国诞生以来，各国都在着手研制智能卡。目

前，智能卡已经广泛地应用于银行、电信、交通等社会的各个方面，发展非常迅速。

由于要借助物理介质，智能卡认证技术是较安全可靠的认证手段之一。智能卡一般分为存储卡和芯片卡。存储卡只用于储存用户的秘密信息，比如用户的密码、密钥、个人化数据等，存储卡本身没有计算功能。芯片卡一般都有一个内置的微处理器，并有相应的 RAM 和可擦写的 EPROM，具有防篡改和防止非法读取的功能。芯片卡不仅可以存储秘密信息，还可以在上面利用秘密信息计算动态口令。智能卡具有广泛的应用，常用的手机 SIM 卡和新一代的身份证都属于智能卡。

类似于智能卡，还有一些其他的物理设备可以用来实现身份认证，比如射频卡等。

口令认证实际上要让用户证明他所知道的内容；而智能卡认证则要让用户证明他所拥有的设备。

3. 基于生物特征的认证

近年来，基于生物特征的认证发展非常迅速。在很多企业、学校已经在使用基于生物特征的设备来负责考勤和安全。

人体有很多特征可以用来唯一标识一个个体，比如指纹、人脸、声音、虹膜等。基于这些生物特征的认证机制各有优缺点：指纹是相对稳定的，但采集指纹不是非侵犯性（非接触性）的。人脸识别具有很多优点，如主动性、非侵犯性等，但面部特征会随年龄变化，而且容易被伪装。语音特征具有与面相特征相似的优点，但也会随年龄、健康状况及环境等因素而变化，而且语音识别系统也比较容易被伪造或被录音所欺骗。最近人们提出的虹膜识别基本上避免了所有上述问题，但虹膜识别技术不太成熟，而且造价较高。

虹膜是盘状的薄膜，位于眼球的前方。同指纹一样，世界上不存在虹膜完全一样的两个人。即便是同一个人，他的左眼和右眼的虹膜也是不一样的。

虹膜认证机制具有很多优点，比如可靠性高，错误接受率和错误拒绝率是最低的；虹膜在眼睛的内部，用外科手术很难改变其结构；由于瞳孔随光线的强弱变化，想用伪造的虹膜代替活的虹膜是不可能的。此外，虹膜的采集也具有非侵犯性，很容易被公众接受。

虽然虹膜认证具有很多优点，但虹膜认证行业的发展面临着两个巨大障碍：技术的不成熟和市场占有率不高。虽然很多公司都在开发虹膜识别产品，但虹膜识别产品的核心技术——虹膜识别算法和图像采集设备，还未被国内厂商所掌握。另外，虹膜认证适合于高安全性应用，并不适合低端用户。

4. 双因素认证

双因素认证是对传统的静态口令机制的改进，并得到了专家和用户的认可，而且已有许多成功案例。

身份认证有三个要素。

（1）所知道的内容：需要使用者记忆身份认证内容，例如密码和身份证号等。

（2）所拥有的物品：使用者拥有的特殊认证加强机制，例如智能卡、射频卡、磁卡等物理设备。

（3）所具备的特征：使用者本身拥有的唯一特征，例如指纹、人脸、声音、虹膜等。

单独来看，这三个要素都有被攻击或破坏的可能：用户所知道的内容可能被别人猜出或者被用户自己忘记；用户所拥有的物品可能被丢失或被偷盗；用户所具备的特征是最为

安全的因素，但是实施起来代价昂贵，一般用在顶级安全需求中。把前两种要素结合起来的身份认证机制就称为双因素认证。

自动提款机采取的认证方式就是双因素认证：使用者必须利用银行卡，再输入个人密码才能对该账户资金进行操作。

相比静态的口令认证机制，双因素认证提高了认证的可靠性，降低了电子商务的两大风险：来自外部非法访问者的身份欺诈和来自内部的网络侵犯。双因素认证比静态口令认证增加了一个认证要素。攻击者仅仅获取了用户口令或者仅仅拿到了用户的令牌访问设备，都无法通过系统的认证。因此，这种方法比基于口令的认证方法具有更好的安全性，在一定程度上解决了静态口令认证机制所面临的威胁。

双因素动态身份认证的解决方案由三个主要部件组成：简单易用的令牌、代理软件以及功能强大的管理服务器。

(1) 令牌：令牌可以使用户在证明自己的身份后获得受保护资源的访问权。令牌会产生一个随机但专用于某个用户的“种子值”，每过一段很小的时间，该“种子值”就会自动更新一次，其数字只有对指定用户在特定的时刻有效(即动态口令)。综合利用用户的密码和令牌的随机“种子值”，使得用户的电子身份很难被模仿、盗用或破坏。

(2) 代理软件：代理软件在终端用户和需要受到保护的网络资源中发挥作用。当一个用户想要访问某个资源时，代理软件会将请求发送到管理服务器端的用户认证引擎。

(3) 管理服务器：管理服务器具有集中式管理能力。当管理服务器收到一个请求时，使用与用户令牌一样的算法和种子值来验证正确的令牌码。如果用户输入正确，就赋予用户一定的权限，否则将提醒用户再次输入。

双因素认证采用了动态口令技术。动态口令技术有两种解决方案：同步方式和异步方式。同步方式中，在服务器端初始化客户端令牌时，就对令牌和服务器端软件进行密钥、时钟和事件计数器同步，然后客户端令牌和服务器端软件基于上述同步数据分别进行运算得到运算结果；用户欲登录系统时，就将运算结果传送给认证服务器。由服务器端进行比较，若两个运算值一致，则表示用户身份合法。整个过程中，认证服务器和客户端令牌没有交互。而在异步过程中，认证服务器需要和客户端令牌进行交互。在服务器端对客户端令牌和服务器端软件进行了密钥、时钟和事件计数器同步之后，一旦用户要登录系统，认证服务器首先向用户发送一个随机数，用户将这个随机数输入到客户端令牌中，令牌返回一个结果，然后用户将这个结果返送给认证服务器，认证服务器将这个值与自己计算得出的值进行比较，如果两者匹配，则证明用户为合法用户；否则拒绝接收该用户的操作。这种机制虽然能够为系统提供比静态口令更高强度的安全保护，但也存在如下缺点。

(1) 只能进行单向认证，即系统可以认证用户，而用户无法对系统进行认证。这就使得攻击者有可能伪装成系统骗取用户的信任。

(2) 不能对要传输的信息进行加密，敏感的信息可能会泄密出去。

(3) 不能保证信息的完整性，即不能保证信息在传输过程中没有被修改。

(4) 不支持用户方和服务器方的双方抗抵赖。

(5) 代价比较大，通常需要在客户端和服务器端增加相应的硬件设备。

(6) 存在单点失效，一旦认证服务器出问题，整个系统就不可用。

由此可见，无论是静态的口令认证机制还是动态的双因素认证机制，都不能提供足够的

安全性。

5. 源地址认证

基于源地址的认证实现最简单，但安全性也最差。它通过鉴别对方的地址来判定对方的身份。目前很多安全产品都有源地址认证功能，比如防火墙。安全管理人员可以通过配置文件来限制一定的访问。但对于黑客来说，只需要简单地通过伪造数据包的方式就可攻破源地址认证机制。

6. 基于 PAP 的认证

用于点对点(Point-Point Protocol，PPP)的身份认证协议，例如通过 ADSL(Asymmetric Digital Subscriber Line，非对称数字用户环路)拨号上网时进行的身份认证就是基于 PAP(Password Authentication Protocol)的。由于 PAP 采取明文口令传输，因此，黑客可以简单地用 Sniffer、Ethereal 等嗅探工具获得用户的用户名和密码。

7. 基于 CHAP 的认证

大多数身份认证协议都属于 CHAP(Challenge Handshake Authentication Protocol，PPP 询问握手认证协议)。这种协议不在网络上传送任何口令信息，因此，相比 PAP，这种身份认证协议要安全得多。

3.3.4　身份认证协议

1. 双向身份认证协议

双向身份认证协议需要消息的发送方和接收方同时在线，而单向认证则没有这样的要求。在重要的商务活动中，通信双方在通信之前要相互确认对方的真实身份，并且希望他们之间的通信不会被第三者阅读。

无论是单向认证还是双向认证，都可以分为基于对称密钥加密和非对称密钥加密两种情况。下面就这两种认证方式分别介绍相应的协议。在介绍协议之前，先弄清几个概念和一些常用记号，对理解协议过程是很有帮助的。

时间戳(Timestamp)：在电子商务中，时间是十分重要的信息。和普通文件一样，在网络上传输的电子商务信息的日期是十分重要的，它是防止文件被伪造、篡改、防抵赖的关键性内容。时间戳是经加密后形成的凭证文档。它包括三个部分：需加时间戳的文件的摘要(Digest)、数字时间服务(Digital Timestamp Service，DTS)、收到文件的日期和时间、DTS 的数字签名。使用时间戳的各方的系统时钟应该是同步的。

随机数：计算机利用一定的算法产生的数值。严格地说，计算机产生的随机数应该称为“伪随机数”，因为这个数不是真正随机的。为了叙述方便，一般都将“伪”字去掉。

序列号：序列号是一个随机数值，通信双方协商各自的序列号初始值。一个新消息当且仅当它有正确的序列号时才被接收。但序列号的方法不适合在身份认证协议中使用。因为它要求每个用户要单独记录与其他每一用户交换的消息的序列号，这样增加了用户的负担。

挑战-应答(Challenge-Answer)：假设 Alice 向 Bob 发送了一个一次性随机数询问，Bob 的回答中应该包含正确的随机数。该机制称为挑战-应答机制。

在以下章节中，我们将采用以下统一的符号。

K_{ab}：通信实体 A 和通信实体 B 共享的对称密钥。

K_{prt}：可信第三方(服务器)的私钥。

K_{pub}：可信第三方(服务器)的公钥。

A→B：通信实体 A 向通信实体 B 传送消息。

$\{M\}_k$：用密钥 k 加密信息 M。

本章介绍的认证协议都是经典的认证协议。这些认证协议曾经在网络安全中发挥过巨大作用,但都相继被发现存在着一定的缺陷。现在这些协议都已经不再使用,但它们仍然是检验新的协议设计和分析方法的有效工具。实际网络中使用的安全协议很多都是下述协议的改进版。为了便于读者理解安全协议的设计,我们首先从这些最简单的协议开始介绍。

1) 基于对称密码的双向认证协议

基于对称密码的双向认证协议有如下几种。

(1) NS 协议。NS 协议最初由 Needham 和 Schroeder 于 1978 年提出。NS 协议在安全协议的发展中起到了非常重要的作用。NS 协议分为基于对称密码(NSSK)和非对称密码(NSPK)两种版本。在这里将要论述的是基于对称密码的 NS 协议。

① A→T：A,B,N_a

② T→A：$\{N_a,B,K_{ab},\{K_{ab},A\}_{K_{bt}}\}_{K_{at}}$

③ A→B：$\{K_{ab},A\}_{K_{bt}}$

④ B→A：$\{N_b\}_{K_{ab}}$

⑤ A→B：$\{N_b+1\}_{K_{ab}}$

NS 协议的目的是在通信双方之间分配会话密钥。参加 NS 协议的主体有三个：通信双方 A 和 B,以及可信第三方 T。

在协议的第一步,A 向 T 发送消息,指明通信双方的身份 A、B 和一个随机数 N_a。

第二步,T 生成 A 和 B 之间的会话密钥 K_{ab},并向 A 发送 N_a、B 和 K_{ab}。用 B 和 T 之间的共享密钥 K_{bt}加密证书$\{K_{ab},A\}_{K_{bt}}$,该证书只能由 B 解密。最后用 T 和 A 之间的共享密钥 K_{at}加密整个消息。

第三步,A 收到 T 发送的消息之后,解密整个消息得到 A、B 之间的共享密钥 K_{ab}和证书$\{K_{ab},A\}_{K_{bt}}$。A 原封不动地向 B 转发这个证书。

第四步,B 解密这个证书得到共享密钥 K_{ab}并用它加密随机数 N_b,然后发送给 A。

第五步,A 收到消息后解密得到 N_b,向 B 做出应答$\{N_b+1\}_{K_{ab}}$。由于只有 A、B、T 三者知道 N_b 和 K_{ab},所以其他任何人都很难冒充其中的一方。

NS 协议整个过程如图 3-13 所示。

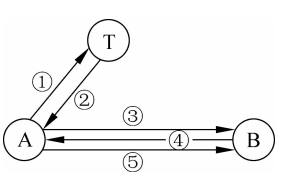

图 3-13　NS 协议过程示意

可以看到,在 NS 协议中使用了随机数。随机数在认证协议设计与分析中具有重要作用：它可以用来保证消息的新鲜性,防止消息重放攻击[①]。时间戳也可以用于抵御重放攻击,但二者之间还是有区别的。使用时间戳时,一般要求各主体的时钟同步,但时间戳并不和某个主体直接关联,任何一个主体产生的时间

① 重放攻击：攻击者复制或伪造诚实用户所期望的消息格式并重放,达到破坏协议安全的目的。重放攻击是很严重的一种攻击,很多攻击形式都是基于此类攻击而实施的。

戳都能被其他主体用来检验消息的新鲜性。随机数则是由某个主体产生的随机数值，一个主体只能根据它自己所产生的临时值来检验消息的新鲜性。此外，时间戳不具有唯一性，它通常有一个有效范围，只要它位于这个有效范围内，主体都接受它的新鲜性。而临时值则具有唯一性，任何一个主体在两次会话中产生相同的临时值的概率是非常小的。

(2) Otway-Rees 协议。Otway-Rees(OR)协议是在 1987 年提出的一种认证协议。同 NS 协议一样，OR 协议其目的也是在通信双方之间分配会话密钥，它需要三个协议参与者：通信双方和认证服务器。

OR 协议的步骤如下。

① A→B：$M,A,B,\{N_a,M,A,B\}_{K_{at}}$

② B→T：$M,A,B,\{N_a,M,A,B\}_{K_{at}},\{N_b,M,A,B\}_{K_{bt}}$

③ T→B：$M,\{N_a,K_{ab}\}_{K_{at}},\{N_b,K_{ab}\}_{K_{bt}}$

④ B→A：$M,\{N_a,K_{ab}\}_{K_{at}}$

协议的第一步，A 生成 M 和随机数 N_a，并向 B 发送通信主体代号 A、B、M 以及服务器需要的信息。

第二步，B 生成随机数 N_b，用 B 和 T 之间的共享密钥加密之后连同 A 发送过来的消息，原封不动地传送给 T。

第三步，T 解密消息，验证这两条加密消息中的 M、A、B 是否一致。如果一致，则 T 生成 A 和 B 之间的会话密钥 K_{ab}，并分别用 A 和 T 以及 B 和 T 的共享密钥加密，将其传送给 B。

第四步，B 将对 A 有用的部分发送给 A。A 和 B 分别解密相应的消息，检查随机数的正确性。如果无误，则 A 和 B 可以应用密钥 K_{ab} 进行会话。

OR 协议中也采用了随机数来防止重放攻击。

OR 协议的步骤如图 3-14 所示。

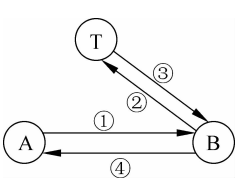

图 3-14　OR 协议过程示意

(3) Yahalom 协议。Yahalom 协议是在 1988 年提出的另外一个经典的认证协议。它同 NS 协议和 OR 协议一样，需要三个协议参与者：通信双方和认证服务器。Yahalom 协议的设计目的也是在不安全的信道上为通信双方分配会话密钥。

Yahalom 协议的步骤如下。

① A→B：A,N_a

② B→T：$B,\{A,N_a,N_b\}_{K_{bt}}$

③ T→A：$\{B,K_{ab},N_a,N_b\}_{K_{at}},\{A,K_{ab}\}_{K_{bt}}$

④ A→B：$\{A,K_{ab}\}_{K_{bt}},\{N_b\}_{K_{ab}}$

协议的第一步，A 向 B 发送 A 和随机数 N_a。

第二步，B 产生随机数 N_b，并用它与 T 的共享密钥 K_{bt} 加密随机数 N_a、N_b。

第三步，T 解密后得到 N_a 和 N_b，然后生成会话密钥 K_{ab}，然后用 A 与 T 的共享密钥加密 K_{ab}、N_a、N_b；用 B 与 T 的共享密钥加密会话密钥 K_{ab} 并发送给 A。

第四步，A 解密收到的消息可以得到会话密钥。然后原封不动向 B 转发 $\{A,K_{ab}\}_{K_{bt}}$ 和用共享密钥加密的随机数 N_b。

比较以上三个协议，可以发现 Yahalom 具有一个特点：A、B 都需要和 T 通信，而 NS

协议和 OR 协议只需要其中一方和 T 进行通信。

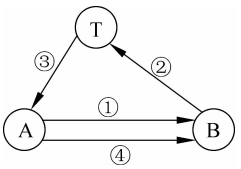

图 3-15　Yahalom 协议过程示意

Yahalom 协议的步骤如图 3-15 所示。

(4) 安全 RPC 协议。另外一种具有代表性的协议是安全 RPC(Remote Procedure Call,远程过程调用)协议,它是早期的认证协议,与上述三个协议不同的是:安全 PRC 协议只包含两个通信主体 A 和 B。A 与 B 之间存在着一个旧的共享会话密钥 K_{ab}。该协议的设计目的在于 A、B 之间协商一个新的会话密钥 K'_{ab}。

安全 RPC 协议的步骤如下。

① A→B: $A,\{N_a\}_{K_{ab}}$

② B→A: $\{N_a+1,N_b\}_{K_{ab}}$

③ A→B: $\{N_b+1\}_{K_{ab}}$

④ B→A: $\{K'_{ab},N'_b\}_{K_{ab}}$

协议的第一步,A 生成随机数 N_a 并用旧的共享密钥 K_{ab} 加密,向 B 发送自己的身份 A 和已加密的消息。

第二步,B 收到 A 的消息之后解密得到 N_a,然后生成随机数 N_b,并用旧的会话密钥 K_{ab} 加密两个随机数并发送给 A。

第三步,A 收到 B 的消息之后,解密得到 N_b,用旧的会话密钥加密 N_b+1 并发送给 B。

第四步,B 用旧会话密钥加密新会话密钥 K'_{ab} 和序列号 N'_b。

(5) 大嘴青蛙协议。大嘴青蛙(Big Mouth Frog)协议是早期最简单的认证协议。该协议仅有两步。

① A→T: $A,\{T_a,B,K_{ab}\}_{K_{at}}$

② T→B: $\{T_t,A,K_{ab}\}_{K_b}$

协议的第一步,A 首先生成 A、B 之间的会话密钥 K_{ab} 和时间戳,并用它和 T 之间的共享密钥加密后发送给服务器。

第二步,服务器解密后得到 A、B 之间的会话密钥 K_{ab},然后用它和 B 之间的共享密钥加密 K_{ab} 和时间戳 T_t 并发送给 B。B 收到后解密就可以得到会话密钥 K_{ab}。

大嘴青蛙协议的整个步骤如图 3-16 所示。

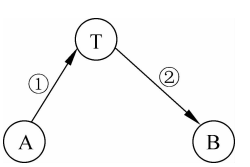

图 3-16　大嘴青蛙协议过程示意

2) 基于非对称密码的双向认证协议

基于非对称密码的 NS 认证协议(NSPK)是 NSSK 的孪生兄弟。它们都是在 1978 年发表的著名认证协议。

多数基于非对称密码的认证协议的设计目的都是使通信双方安全地交换共享会话密钥。但 NSPK 认证协议的目的却是使通信双方安全地交换两个彼此独立的秘密。同 NSSK 认证协议一样,NSPK 协议的参与者也是三个:通信主体 A、B 以及充当可信第三方的服务器 T。

NSPK 协议的大体步骤如下。

① A→T: A,B

② T→A: $\{K_b,B\}_{K_{pri}}$

③ A→B：$\{N_a, A\}_{K_b}$

④ B→T：B, A

⑤ T→B：$\{K_a, A\}_{K_{pri}}$

⑥ B→A：$\{N_a, N_b\}_{K_a}$

⑦ A→B：$\{N_b\}_{K_b}$

协议的第一步，A 向 T 发送通信主体名称 A 和 B。

第二步，T 向 A 发送用它的秘密密钥 K_{pri} 加密的 B 及其公开密钥 K_b。

第三步，A 用 T 的公开密钥解密他收到的消息 2，得到 B 的公开密钥 K_b，并向 B 发送用 K_b 加密的随机数 N_a 与 A。

第四步，B 向 T 发送通信主体名称 A 和 B。

第五步，T 向 B 发送用它的秘密密钥 K_{pri} 加密的 A 及其公开密钥 K_a。

第六步，B 用 T 的公开密钥解密他收到的消息 5，得到 A 的公开密钥 K_a 并向 A 发送用 K_a 加密的随机数 N_a 与 N_b。

第七步，A 向 B 发送用 K_b 加密的临时值 N_b。这里，临时值 N_a 与 N_b 就是 A 与 B 希望交换的秘密。

NSPK 协议有两个重要作用：其一，通过消息 1，2，4 和 5 的交换，达到 A 与 B 相互获得对方公开密钥的目的；其二，通过消息 3，6，7 的交换，A 与 B 分别得到对方的秘密，达到交换秘密 N_a 与 N_b 的目的。

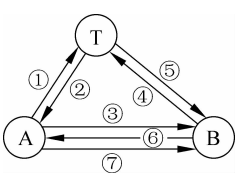

图 3-17　NSPK 协议过程示意

NSPK 协议的步骤如图 3-17 所示。

2. 单向身份认证协议

单向认证是指仅由通信的一方对另一方进行身份认证。

单向身份认证有两种情况：发送方对接收方进行认证和接收方对发送方进行认证。单向身份认证不需要消息的发送方和接收方同时在线。很多应用程序都需要采用单向身份认证协议，比如电子邮件等。发送方先将消息发往接收方的邮箱，接收方阅读时直接从邮箱获取消息。在商务活动中，接收方在阅读前需要对发送方进行身份认证。例如，甲方收到乙方的文件后总要先确认文件是否真正由乙方发出。好比生活中我们收到信件后，总要先检验信件的署名和笔迹。

1）基于对称密码的单向认证协议

由于在单向认证协议中不要求发送方和接收方同时在线，因此，发送方和接收方之间就不应该有交互过程。对基于对称密码的双向 NS 协议稍作改动，就可以形成基于对称密码的单向认证协议 NS^+。

① A→T：A, B, N_a

② T→A：$\{K_{ab}, B, N_a, \{K_{ab}, A\}_{K_{bt}}\}_{K_{at}}$

③ A→B：$\{K_{ab}, A\}_{K_{bt}}, \{M\}_{K_{ab}}$

协议的第一步，A 向 T 发送通信实体的代号 A、B 和一个随机数 N_a。

第二步，T 首先产生 A、B 之间的会话密钥 K_{ab}，然后用 T 和 B 之间的共享密钥产生一个证书 $\{K_{ab}, A\}_{K_{bt}}$。该证书只能由 B 解密。然后 T 将该证书和会话密钥 K_{ab} 以及其他信息

一起用 T 与 A 之间的共享密钥 K_{at} 加密后发送给 A。

第三步，A 解密从 T 发送过来的信息，得到密钥 K_{ab}。用 K_{ab} 加密待发信息 M，连同证书 $\{K_{ab},A\}_{K_{bt}}$ 一起发送给 B。

当 B 收到这两条消息之后，首先解密证书得到会话密钥 K_{ab}，然后再用 K_{ab} 解密消息。

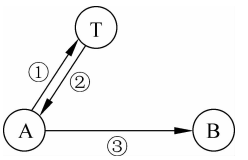

图 3-18　NS⁺ 协议步骤示意

该协议的步骤如图 3-18 所示。

2）基于非对称密码的单向认证协议

非对称密码算法具有很多优点，它既能用于加密数据，又能提供认证性。

基于非对称密码的单向认证协议实现非常简单，只有一个回合。

A→B：$\{M,\{H(M)\}_{K_{a\_pri}},\{T,A,K_{a\_pub}\}_{K_{pri}}\}_{K_{b\_pub}}$

A 采用一定的消息摘要算法，计算消息的摘要 $H(M)$，然后用自己的私钥对该消息摘要进行签名。A 再将原消息、消息签名以及自己的证书一起用 B 的公钥加密之后发送给 B。

该协议是接收方对发送方的认证，同样，有时发送方需要对接收方进行认证。

3. 零知识身份认证

关于零知识协议，Bruce Schneier 曾经给出了一个很生动的故事。

Alice：“我知道联邦储备系统计算机的口令、汉堡包秘密调味汁的成分以及 Knuth 第四卷的内容。”

Bob：“不，你不知道。”

Alice：“我知道。”

Bob：“你不知道！”

Alice：“我确实知道！”

Bob：“请你证实这一点！”

Alice：“好吧，我告诉你。”（Alice 悄悄说出了口令。）

Bob：“太有趣了！现在我也知道了。我要告诉《华盛顿邮报》！”

Alice：“啊呀！”

这个故事说明了一个道理，通常 Alice 要使 Bob 确信她知道某个秘密，那么她需要把该秘密告诉 Bob 以得到证实。但这样一来，Bob 也知道了这个秘密。Bob 可以将这个秘密告诉给任何人，而 Alice 却无力阻止 Bob 的胡作非为。

但零知识协议改变了这种状况。在不告诉 Bob 秘密的情况下，Alice 可以向 Bob 证明她确实知道某个秘密。

Jean Jacques Quisquater 和 Louis Guillou 用一个关于洞穴的故事来解释零知识。如图 3-19 所示，洞穴中有一道门，只有知道咒语的人才可以打开这道门。如果不知道咒语，门两边的通道都是死胡同。

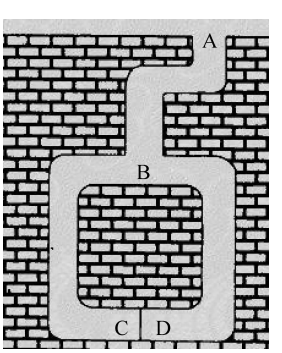

图 3-19　零知识认证

Alice 要向 Bob 证明她知道这道门的咒语，而她又不愿意将咒语泄漏给 Bob。那么 Alice 可以按照以下步骤来使 Bob 相信她确实知道这道门的咒语。

（1）Bob 站在 A 点。

（2）Alice 一直走进洞穴，到达 C 点或者 D 点。

(3) 在 Alice 消失在洞穴中之后，Bob 走到 B 点。

(4) Bob 向 Alice 喊叫，要她：从左通道出来，或者从右通道出来。

(5) Alice 答应了。如果有必要，她就用咒语打开门。

(6) Alice 和 Bob 重复第(1)步到第(5)步 n 次。

这个协议所使用的技术称为**分割选择**。使用这个协议，Alice 可以向 Bob 证明她确实知道开启门的咒语。原因在于：Alice 不能猜出每一次 Bob 要她从哪边出来。第一次，Alice 猜中的概率为 1/2，连续两次猜中的概率为 1/4……如果重复第(1)步到第(5)步 n 次，Alice 欺骗成功的概率仅仅为 $1/2^n$。因此，经过很多次试验之后，如果 Alice 的每一次试验结果都是正确的，那么 Bob 完全可以肯定 Alice 确实知道开启门的咒语。

基本的零知识协议由下面几部分组成。

(1) Alice 用她的消息和一个随机数将一个难题转变为另一个难题，新的难题和原来的难题同构。然后她用她的信息和这个随机数解这个新的难题。

(2) Alice 提交这个新的难题的解法。

(3) Alice 向 Bob 透露这个新难题。Bob 不能用这个新难题得到关于原难题或其解法的任何信息。

(4) Bob 要求 Alice：向他证明新旧难题是同构的，或者公开她在第(2)步中提到的解法并证明是新难题的解法。

(5) Alice 同意 Bob 的要求。

Alice 和 Bob 重复第(1)步到第(5)步 n 次。

4. 身份认证的实现与应用

1) RADIUS

RADIUS (Remote Authentication Dial In User Service)是一种远程身份验证拨入用户服务协议。RADIUS 协议最初是由 Livingston 公司提出的，设计 RADIUS 的初衷是为拨号用户进行认证和计费。后来经过多次改进，形成了一项通用的认证、计费、授权协议。RADIUS 协议是完全开放的，任何安全系统和厂商都可以免费使用它的公开源代码。RADIUS 协议应用范围很广，包括普通电话、上网业务计费，对虚拟专用网 (Virtual Private Network，VPN)的支持可以使不同的拨入服务器的用户具有不同权限。无线网络的接入认证也采用 RADIUS 协议。RADIUS 协议已经被广泛实施在各种各样的高安全级别的网络环境中。

RFC 2865 和 RFC 2866 对 RADIUS 作了详细的描述。RADIUS 能够提供身份验证(Authentication)、授权(Authorization)和记账(Account)服务，即 AAA 服务。

RFC 2865 和 RFC 2866 定义了以下 RADIUS 消息类型。

(1) 访问-请求(Access-Request)：由 RADIUS 客户端发送，请求对连接尝试进行身份验证和授权。

(2) 访问-接收(Access-Accept)：由 RADIUS 服务器发送，以响应“访问-请求”消息。此消息表明 RADIUS 服务器已接受客户端的连接请求。

(3) 访问-拒绝(Access-Reject)：由 RADIUS 服务器发送，以响应“访问-请求”消息。此消息通知 RADIUS 客户端连接尝试被拒绝。如果凭据未被验证或连接尝试未被授权，RADIUS 服务器将发送此消息。

(4) 访问-质询(Access-Challenge)：由 RADIUS 服务器发送，以响应“访问-请求”消息。此消息是对需要响应的 RADIUS 客户端的质询。

(5) 记账-请求(Accounting-Request)：由 RADIUS 客户端发送，为接受的连接指定记账信息。

(6) 记账-响应(Accounting-Response)：由 RADIUS 服务器发送，以响应“记账-请求”消息。此消息确认对记账请求消息的成功接受和处理。

RADIUS 采用了客户/服务器模式(C/S)，它的客户端最初就是网络接入服务器(Net Access Server，NAS)，现在任何运行 RADIUS 客户端软件的计算机都可以成为 RADIUS 的客户端。RADIUS 协议认证机制很灵活，可以采用 PAP、CHAP 或者 UNIX 登录认证等多种方式。RADIUS 的认证步骤如图 3-20 所示。

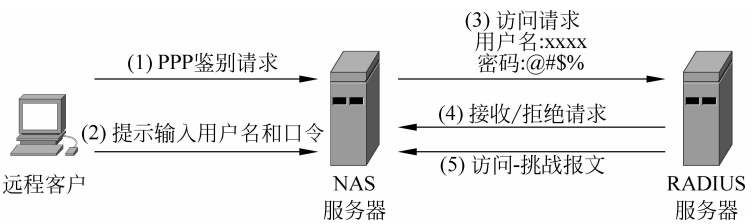

图 3-20　RADIUS 认证步骤

(1) 远程客户和 NAS 服务器建立一个 PPP 会话，并发送 PPP 鉴别请求。

(2) 网络接入服务器提醒用户输入用户名和密码。用户输入用户名和密码后发送给 NAS。

(3) 网络接入服务器从客户端接收用户名和密码，并把此信息和其他属性放在一个 Access-Request 数据包中发送到 RADIUS 安全服务端。在发送数据包前计算用户密码的 Hash 值，用户名以明文形式发送，密码以 Hash 值的方式传送。

(4) RADIUS 安全服务器验证客户机的合法性，并对用户进行鉴别。根据用户提供的用户名和密码，与数据库中的用户名和密码匹配。根据匹配结果，RADIUS 服务器返回应答报文。应答报文是下列报文中的一个。

Access-Accept：如果结果不匹配，远程用户就不会通过认证。网络接入服务器提示用户重新输入用户名和口令，如果连续匹配失败，该用户将被拒绝访问。

Access-Accept：如果结果匹配，远程用户会通过认证。

Access-Challenge：RADIUS 服务器发出此响应以核实用户的身份。网络接入服务器提示用户输入其他数据并把数据发送给 RADIUS 安全服务器。此数据包可定期地发送到网络接入服务器，从而让用户再次提交用户名和口令。以避免用户一次登录成功后，忘却了关闭会话，从而导致安全隐患。

RADIUS 还支持代理和漫游功能。所谓代理，就是一台服务器，它负责转发开启 RADIUS 协议的计算机之间的 RADIUS 数据包。所谓漫游功能，是代理的一个具体实现，它可以让用户通过和其无关的 RADIUS 服务器接入网络，类似于手机漫游，在上海注册的手机用户在北京同样可以接听或拨打电话。

以上讨论的是 RADIUS 协议的认证功能。关于 RADIUS 的记账和授权功能请参见相关的论述。

2) Kerberos

Kerberos 这个名称来源于希腊神话，意思是“多头狗”，它守卫着地狱的大门。这用于比喻网络安全中的认证协议简直再恰当不过了。Kerberos 认证协议是由 MIT(美国麻省理工学院)设计的，其设计目标是允许客户以一种安全的方式来访问网络资源。其基础是 NS 协议。Kerberos 与 NS 协议不同之处在于：Kerberos 认为所有的时钟已经同步好了。Kerberos 经历了很多改版，其当前版本是 Kerberos v5，但是大多数应用仍然是基于版本 4 的。

Kerberos 协议不同于先前介绍的所有协议，它有四个参与者：通信主体客户 A，服务器 B 以及认证服务器(Authentication Server，AS)，票据服务器(Ticker Granting Server，TGS)。认证服务器的作用是对登录的每个主体进行认证；票据服务器的作用在于向网络上的服务器证明客户的真实身份。

Kerberos 提供了一种主体验证与标识的方法。假设客户 Alice 希望同服务器 Bob 进行通信，利用 Kerberos 进行认证的过程有以下几步。

(1) 客户 A 请求 Kerberos 认证服务器(AS)发给接入 Kerberos TGS 的票据。

(2) 认证服务器(AS)在其数据库中查找与客户 A 相对应的信息，比如实体 A 的详细地址、A 与 AS 的共享密钥 K_{as}；然后 AS 产生一个会话密钥 KS，用客户 A 的密钥 K_{as} 所导出的密钥 K'_{as} 对此会话密钥 KS 进行加密；再生成一个票据分配许可证(Ticket-Granting Ticket，TGT)，并用 AS 与 TGS 的共享密钥 K_{st} 对 TGT 进行加密。许可证包含 KS、时间戳等信息。认证服务器(AS)把加密信息发送给客户 A。

(3) 客户 A 利用自己的密钥 K_{as} 导出密钥 K'_{as}，用它解密就可以得到会话密钥 KS。然后，客户 A 向 TGS 发出请求，申请接入某一目标服务器 B 的票据。此请求包括目标服务器名称 B、TGT 以及用 KS 加密的时间戳等信息。

(4) 票据服务器(TGS)用它与认证服务器(AS)共享的密钥 K_{st} 对 TGT 进行解密得到 KS。然后 TGS 产生新的会话密钥 K_{ab} 供客户实体 A 与目标服务器 B 使用。TGS 向客户 A 发送两条消息：第一条消息用会话密钥 KS 加密，加密的内容是新的会话密钥 K_{ab} 和目标服务器 B 的名称；第二条消息是 A 访问服务器 B 的票据，该票据用 TGS 与目标服务器 B 的共享密钥 K_{bt} 加密，票据内容包含通信主体 A 的名称和会话密钥 K_{ab}。最后将这两个加密信息都发送给客户 A。

(5) 客户 A 将接收到的第一个报文用 KS 解密后，获得与目标服务器 B 的会话密钥 K_{ab}。这时，客户 A 制作一个新的认证单，包括时间戳、地址等信息，并用获得的会话密钥 K_{ab} 对该认证单进行加密。当 A 需要访问目标服务器 B 时，将加密的认证单和从 TGS 收到的票据 Ticket 一并发给目标服务器 B。

(6) 目标服务器 B 对票据和认证单进行解密检查，如果一切检查均无错误，服务器 B 做出响应，将时间戳加 1，然后用会话密钥 K_{ab} 加密后发送给 A，以便让 A 确认服务器 B 已经收到会话密钥 K_{ab}。

为了便于理解，我们去掉了协议中的一些非关键信息，并且将以上过程用形式化表示如下。

① A→AS：A

② AS→A：$\{KS\}_{K'_{as}}$，$\{KS, T_s\}_{K_{st}}$ (TGT=$\{KS, T_s\}$)

③ A→TGS：B，TGT，$\{T_a\}_{KS}$

④ TGS→A：$\{B,K_{ab}\}_{KS}$，Ticket=$\{A,K_{ab}\}_{K_{bt}}$

⑤ A→B：Ticket，$\{T'_a\}_{K_{ab}}$

⑥ B→A：$\{T'_a+1\}_{K_{ab}}$

Kerberos 协议认证过程如图 3-21 所示。

Kerberos v5 比 v4 复杂一些，也需要更多的开销。在 Kerberos v5 中，票据有更长的生存期，允许票据被重复使用，并且也允许发送未来才有效的票据。关于 Kerberos v5 的详细介绍，读者可以参阅相关书籍。

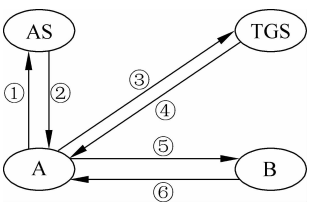

图 3-21　Kerberos 协议步骤示意

3.3.5　匿名认证

先介绍一个概念，个人信息（私人信息），指能够对个人进行甄别的相关信息或者个人所有与其自身权利相关的信息，如姓名、性别、出生年月日、住址、电话号码、职业、收入情况、银行账号、信用卡号码等。这些信息中，有些是和个人机密信息相关，如银行账号、信用卡号码等，有些则是和个人隐私相关，如电话通信记录、购物记录、就诊病历等。

传统的实名认证，就是根据某些个人信息来对个人的身份进行甄别，以达到认证目的，也就是所谓的对号入座。被认证对象的个人信息在认证过程中被认证机构锁定。

个人信息的泄漏以及针对个人保密信息的攻击已经成了值得重视的话题。信用卡账号被盗，个人银行密码被攻破，这类严重的信息泄露自然不必说，就是连 E-mail 地址泄漏所导致的垃圾邮件干扰也很让人头疼。对于某些游戏玩家来说，游戏账号被盗恐怕也属于极具杀伤力的网络安全事件。这类事件的发生，很大程度上是因为传统的认证过程要求绑定个人信息，使个人信息不得不在网络上多次、多点流动，从而增加了被攻击的机会。另外传统模式下多种个人信息呈聚集状分布（如通过银行账号不难检索出交易记录），一旦单点被击破就很容易导致多种个人信息泄露。

由于个人信息的保护日益受到重视，匿名认证技术也逐步成为信息安全技术的一个热点。匿名认证，从原理上可以简单地归结为将认证过程和个人信息相分离。按照分离的手段不同，匿名认证可以有多种实现方式。

匿名认证技术使用零知识证明作为指导思想，以盲签名作为具体的签名手段，达到认证过程和被认证对象的个人信息相分离的目的。目前主要的匿名认证体系有团签名（group signature）方案和匿名令牌（anonymous token）方案。

1. 团签名

在传统的认证方-被认证者系统中，增加一名团管理者（group manager），用于管理被认证者的个人信息。被认证者需要加入团，获取用于签名的成员私钥，认证方获取成员公钥。认证双方根据口令消息（challenge message）和钥对匹配来完成认证过程。

团签名是最简单的一种匿名认证体系，它只是在传统的认证体系中增加了一名团管理者角色，以达到认证和个人信息相分离的目的。但是这种认证体系有一个缺点——成员私钥的管理问题。按照协议，每一个成员都拥有一个成员私钥，而这些私钥要么是等同的，要么是具有等效签名能力的，否则认证方无法根据同一公钥进行认证，而如果认证方使用不同的公钥来认证不同的对象，也就基本失去了匿名认证的意义。无区别的成员私钥的使用也

就导致了无区别的权力分配，这样就很难对各成员进行细化的资源配置管理。

2．匿名令牌

匿名令牌方案是在团签名方案的基础上做了一些改进。令牌管理者配发给被认证者的不是上述的成员私钥，而是结构更为复杂的匿名令牌。匿名令牌，简单地说，就是包含成员个人信息的部分片断的数据结构。成员的个人信息的另一部分片断被令牌管理者所持有，而如果没有这部分片断信息，匿名令牌中的成员个人信息是极难被解读的。

除此之外，匿名令牌本身是一次一乱的，也就是理论上即使是同一个成员，每次认证时使用的令牌都是不相同的。匿名令牌能够做到这一点，是因为每次认证完毕后认证方都会对匿名令牌进行刷新，而刷新的过程通过密钥交换(key exchange)协议来完成，以保证密钥本身不会在网络上被窃取。

习题 3

1．计算机安全四大原则是什么？

2．什么是机密性？什么是认证性？什么是完整性？什么是不可抵赖？请分别予以阐述。

3．什么是单向函数？请利用你所学过的数学知识，寻找一个函数 $y=f(x)$，使其具有以下要求：

(1) x，y 之间是一一对应的；

(2) 已知 x，计算 y 很容易；

(3) 已知 y，计算 x 相对比较复杂。

4．什么是消息认证？认证的内容包括哪些？

5．消息认证的方法有哪两种？请阐述它们之间的区别。

6．什么是数字签名？有哪些比较著名的数字签名的算法？

7．请用图表示 RSA 签名算法和 DSA 签名算法的过程。

8．什么是多重数字签名？什么是不可抵赖数字签名？什么是盲签名？请举例说明它们分别用于什么场合。

9．什么是身份认证？为什么需要身份认证？一个成熟的身份认证系统包含哪些特征？

10．身份认证的方法有哪些？

11．如何保证口令安全性？

12．什么是基于智能卡的身份认证？你周围有采用智能卡认证的例子吗？

13．请阐述身份认证和数字签名的区别。

14．什么是零知识认证？为什么需要零知识认证？

15．什么是单向身份认证协议？什么是双向身份认证协议？参照 3.3.4 节的内容，请设计一个单向身份认证协议和双向身份认证协议。

16．什么是 RADIUS？请描述 RADIUS 认证的步骤。

17．什么是 Kerberos？请描述 Kerberos 的协议步骤。