第 3 章

身 份 认 证

身份认证是信息系统的第一道安全防线，其目的是确保用户的合法性，阻止非法用户访问系统。身份认证对确保信息系统和数据的安全是极其重要的，可以通过验证用户知道什么、用户拥有什么、用户的生物特征等方法来进行用户的身份认证。

3.1 节给出了身份认证的定义和主要的认证方式；3.2 节介绍了常用的口令认证机制并分析了该机制存在的安全隐患和相应的增强机制；3.3 节介绍了一次性口令认证的原理和实现机制；3.4 节介绍了智能卡认证方式，详细介绍了 USB Key 认证原理；3.5 节简要介绍了基于生物特征的认证方式；3.6 节介绍了三种身份认证协议；3.7 节简要介绍了零知识证明。

视频讲解

3.1 概　　述

身份认证是验证主体的真实身份与其所声称的身份是否相符的过程，它是信息系统的第一道安全防线，如果身份认证系统被攻破，那么信息系统所有其他安全措施将形同虚设，因此，身份认证是信息系统其他安全机制的基础。

身份认证包括标识与鉴别两个过程。标识(Identification)是系统为区分用户身份而建立的用户标识符，一般是在用户注册到系统时建立，用户标识符必须是唯一的且不能伪造。将用户标识符与用户物理身份联系的过程称为鉴别(Authentication)，鉴别要求用户出示能够证明其身份的特殊信息，并且这个信息是秘密的或独一无二的，任何其他用户都不能拥有它。

【例 3-1】 用户 Alice 在某信息系统中的标识符为 abtoklas，跟用户标识符相关的认证信息是用户口令，该口令存储在信息系统中，只有 Alice 和系统知道，如果没有人能获取或猜测 Alice 的口令，那么标识符和密码的组合可以认证用户的身份。

常见的身份认证方式有以下几种。

(1) 利用用户所知道的东西，如口令、PIN(Personal Identification Number)码或者对预先设置问题的答案。

(2) 利用用户所拥有的东西，如电子钥匙卡、磁卡、智能卡等物理识别设备。

(3) 利用用户所具有的生理特征，也称为静态生物特征，如指纹、声音、视网膜、DNA 等。

(4) 利用用户的行为特征，也称为动态生物特征，如通过语音模式、笔迹特征、打字节奏等进行的识别。

这几类身份认证方式各有利弊。第一类方法最简单,系统开销小,但是最不安全;第二类安全性比第一类高,但是认证系统相对复杂;第三、四类的安全性最高,但是涉及更复杂的算法和实现技术,且成本高。前两类身份认证技术起步较早,目前相对成熟,应用也比较广泛,第三、四类技术由于在安全性上的优势正在迅速发展中。

尽管上述每种方法都可以给用户提供一定的安全认证服务,但每种方法都存在一些问题,如口令可能被猜到,智能卡可能会丢失,采用生物特征进行认证存在误报、漏报等问题。在现实中,可以通过多个因素共同鉴别用户身份的真伪,例如,在银行 ATM 机上取款需要插入银行卡,同时需要输入银行卡密码,这就采用了双因素认证,鉴别的因子越多,鉴别真伪的可靠性就越大。当然,在设计鉴别机制时需要考虑认证的方便性和性能等综合因素。

下面逐一讨论以上三种形式的身份认证技术。

3.2 基于口令的认证

3.2.1 口令认证过程

口令是通信双方预先约定的秘密数据,基于口令的认证方式是最常用的一种身份认证技术,这种认证方法简单易行,不需要投入太多就可以实现,广泛应用于操作系统、网络、数据库和应用程序中。

在网络环境下的口令认证中,用户事先在服务器上进行注册,建立自己的用户账号,包括用户标识符和口令,这些信息存储在服务器口令表中,只有用户自己和服务器知道,通常情况下,只要用户保持口令的保密性,非授权用户就无法使用该用户的账户。图 3.1 描述了口令认证的过程,用户在登录界面输入用户标识符和口令,服务器在接收到客户端认证请求后,跟数据库口令表中存储的信息进行比对,如果找到匹配项则认证通过,否则返回认证失败信息。

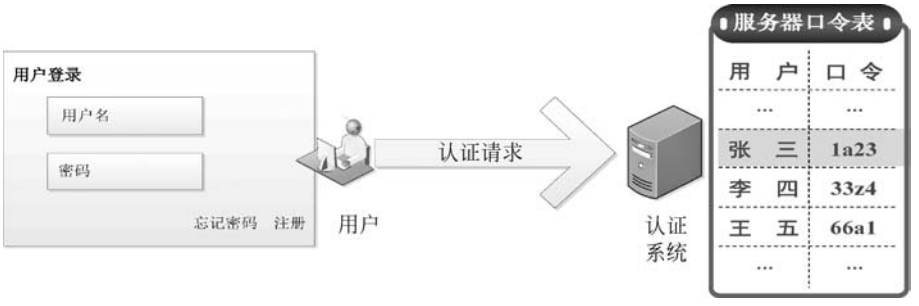


图 3.1 基于口令的身份认证过程

如图 3.1 所示身份认证方式的优点是简单易用,在安全性要求不高的情况下易于实现,但是该机制存在严重的安全问题。

1. 口令质量不高

通常用户为了记忆、使用方便,会采用与自己周围事物相关的单词或数字作为口令,

这些类型的口令容易破解，常被称为弱口令，常见的弱口令类型有以下几种。

（1）用户名或用户名的变形。

（2）电话号码、执照号码等。

（3）一些常见的单词。

（4）生日。

（5）长度小于5的口令。

（6）空口令或默认口令。

（7）上述词后加上数字。

针对弱口令的攻击主要有字典攻击和穷举攻击（暴力破解）。

1）字典攻击

由于大部分人选用的密码都是与自己周围事物有关的单词或数字，攻击者利用各种手段收集用户可能使用的口令构成口令字典，借助于口令破解工具逐一尝试字典中的口令，实现口令的破解。

【例3-2】 口令字典是一个字符串表，表中的每一项都是可能被选用的口令，图3.2就是典型的口令字典。

2）穷举攻击（暴力破解）

如果把字符串的全集作为字典进行攻击称为穷举攻击，通俗地说，就是尝试所有可能的口令，直到破解成功，这是一种最原始、最粗犷的方式，也称为暴力破解。当用户的密码比较短时，就很容易被穷举。

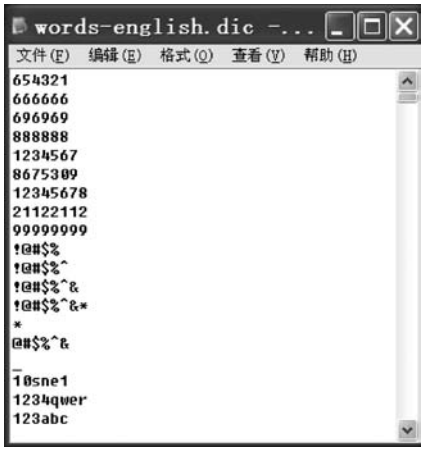


图3.2　口令字典

【例3-3】 在Windows（Windows NT和Windows 2000/XP）平台上，口令文件是%systemroot%\system32\config中一个名为“SAM”的文件；在UNIX/Linux平台上，口令文件是/etc/passwd或者/etc/shadow。一旦攻击者获得了以上信息，就会尝试各种口令攻击工具来进行破解。常用的两个口令破解工具如John the Ripper和Lophtcrack，分别用于破解UNIX/Linux系统和Windows NT/2000/XP下的口令文件，这两个口令破解工具的原理就是字典攻击和穷举攻击。

2. 口令明文存储

认证服务器端如果采用明文方式存储口令，一旦攻击者成功访问数据库，则可以得到所有用户的用户名、口令信息。

【例3-4】 2011年12月21日，中国最大的软件开发者技术社区CSDN后台数据库被盗，由于明文存储，642万多个用户的账号、口令等信息被泄露。

3. 口令嗅探和重放攻击

一些信息系统对传输的口令没有进行加密处理，攻击者通过网络嗅探可以轻易得到口令的明文。即使口令经过加密也难以抵抗重放攻击，因为攻击者可以将截获的加密信

息直接发送给认证服务器,这些加密信息是合法有效的,服务器同样也能认证通过。

4. 攻击者运用社会工程学、键盘记录器等获取口令

此处不展开详细介绍。

3.2.2 口令认证安全增强机制

为此,需要采取措施对口令认证机制进行安全增强。

1. 提高口令质量

口令认证的安全性很大程度上取决于口令质量,涉及以下几点。

(1) 增加口令的复杂度和长度。

增加口令复杂度需要增大口令的字符空间,口令的字符空间不要仅限于常见的26个小写字母、0～9这10个数字,要扩大到所有可被系统接收的字符,如26个大写字母,特殊字符@、#、%、*等,选用的口令最好是字母、数字、特殊字符的组合。表3.1给出了可通过键盘输入的字符分类。同时还需注意口令的长度,选择长口令可以增加破解的时间,假定口令长度为4,口令字符空间为95,则组成的所有可能口令数目为95^4,对于一台处理性能为每秒100万条指令的高性能计算机来说,采用穷举攻击破译口令大约需要2min,而当口令长度增长到6,采用同样的计算机,破译时间增长到8天,由此可见,口令破解的难度随口令长度呈指数级增长。满足复杂度和长度要求的口令为强口令,为了增强口令认证的安全性,需要采用不容易被破解的强口令。

表3.1 可通过键盘输入的字符分类

分　类	个　数	解　释
小写字母	26	abcdefghijklmnopqrstuvwxyz
大写字母	26	ABCDEFGHIJKLMNOPQRSTUVWXYZ
数字	10	0123456789
其他	33	! @ # $ %^& * ()－＝_＋[]{}|；'：",.＜＞/？`～
总计	95	

(2) 在用户使用口令登录时,还可以采取更加严格的控制措施。

① 限制登录时间。例如,用户只能在某段时间内才能登录到系统中。

② 限制登录次数。例如,如果有人连续多次登录失败,则认证系统拒绝再次认证请求,这样可以防止字典攻击和暴力破解。

③ 尽量减少会话透露的信息。例如,登录失败时不提示是用户名错还是口令错,使外漏的信息最少。

2. 口令加密存储

采用明文方式存储口令有很大风险,任何人只要得到存储口令的数据库,就可以得到全体合法用户的口令。

目前常用的方法是服务器口令文件中存储用户名(标识符)和口令的散列值,认证过程如图3.3所示,Alice是示证者,Bob是验证者,Alice向Bob发送身份标识符Alice和口令Pass,Bob接收到口令后,计算Hash(Pass),并与口令文件中相应的散列值进行比对,匹配则允许登录,否则拒绝登录。

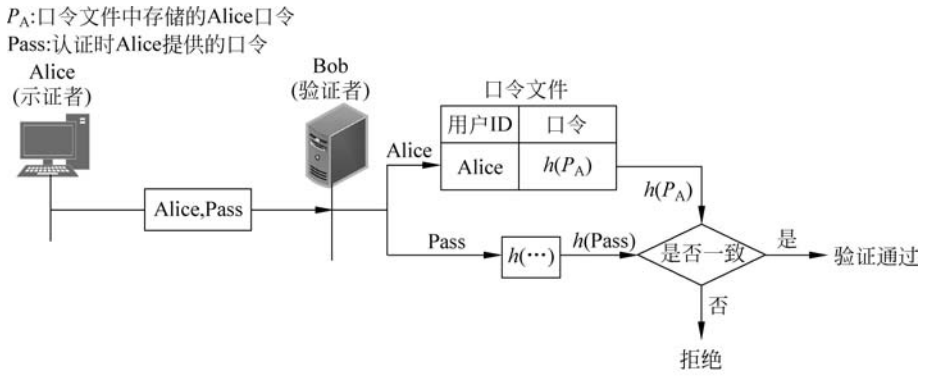


图 3.3　口令直接散列的认证过程

由于在口令文件中存储的是口令的散列值，通过散列值想要计算出原始口令在计算上是不可能的，这就相对增加了安全性。但是如果攻击者获取了口令文件，也可以采用字典攻击方法，用跟目标系统相同的散列函数计算出各口令的散列值，与窃取的口令散列值进行比对，实施口令的破解。为了防范这种攻击，在实际使用的系统中，在计算口令散列时会使用“盐值”来增加口令加密的复杂性。例如，在 UNIX 类操作系统中，“盐值”是一个随机数，口令和“盐值”作为散列函数的输入，生成一个定长的散列码，存储在口令文件中，用户身份认证过程如图 3.4 所示。当 Alice 试图登录 UNIX 时，Alice 向 UNIX 发送身份标识符 Alice 和口令 Pass，操作系统用标识符 Alice 检索口令文件，获得对应的“盐值”和口令的密文，“盐值”和用户提供的口令 Pass 被作为输入数据进行加密，如果加密结果跟口令文件中存储的加密口令相匹配，那么用户身份认证通过，否则拒绝登录。

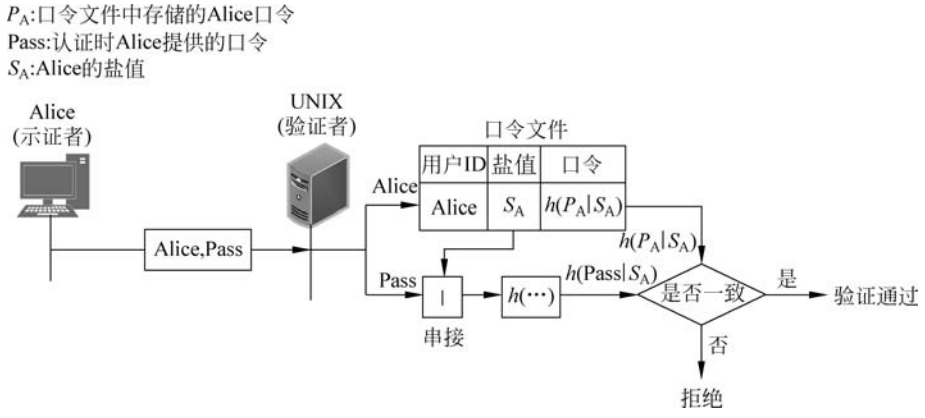


图 3.4　加了盐值的口令散列认证过程

使用“盐值”具有如下好处。

(1) 可以防止相同口令在口令文件中可见。即使两个不同的用户选择了相同的口令，这些口令也会被分配不同的“盐值”，因此，这两个用户加密后的口令是不相同的。

(2) 显著增加了离线口令字典攻击的难度。对于一个 b 位长度的“盐值”，可能产生

的口令数量将会增长 2^b 倍,这将大大增加通过字典攻击猜测口令的难度。

(3) 使得攻击者几乎不可能发现一个用户是否在两个或更多的系统中使用了相同的口令。

3. 口令加密传输

在通信链路上,如果口令以明文传输,黑客可以采用网络监听工具对通信内容进行网络嗅探,盗取传输的用户名和口令信息,为了应对这种网络嗅探行为,可以对传输的口令信息进行加密,例如,利用散列函数对传输的口令进行加密处理。

但是即使对传输的口令进行加密,攻击者仍可以冒充用户身份,利用截获的密文信息,攻击者可以构造新的登录请求并将其提交到同一服务器,服务器不能区分这个登录请求是来自合法用户还是来自攻击者,这种攻击称为重放攻击。传统的口令认证之所以无法抵御重放攻击,主要原因是通常使用的计算机口令是静态的,也就是说,在一定时间内是不变的,而且可重复使用,这就给攻击者可乘之机,利用截获的信息可以重复登录,冒充用户身份执行非法操作。为了解决这个问题,需要采用一次性口令技术(又称为动态口令),同一口令不能重复认证,即使攻击者截获了传输的认证数据包,采用重放攻击再次向服务器发送,由于数据包中的口令已经被使用过,无法验证通过,采用动态口令可以有效抵抗重放攻击。

视频讲解

3.3 一次性口令的认证

20世纪80年代初,针对静态口令认证的缺陷,美国科学家 Leslie Lamport 首次提出了利用散列函数产生一次性口令(One Time Password,OPT)的思想,即在用户的每次登录过程中,加入不确定因素以生成动态变化的示证信息,从而提高登录过程的安全性。由贝尔通信研究中心于1991年开发的 S/KEY 是一次性口令的首次实现。

目前,一次性口令技术已经得到了广泛应用,比较简易的实现有短信密码、验证码、口令卡等。在短信密码中,身份认证系统以短信形式发送随机的6/8位密码到客户的手机上,客户在登录或交易认证时输入此动态密码,从而确保系统身份认证的安全。验证码通常称为全自动区分计算机和人类的图灵测试(Completely Automated Public Turing Test to Tell Computer and Humans Apart,CAPTCHA),是一种主要区分用户是计算机和人的自动程序。这类验证码的随机性不仅可以防止口令猜测攻击,还可以有效防止攻击者对某一个特定注册用户用特定程序进行不断的登录尝试,例如刷票、恶意注册、论坛灌水等。

【例 3-5】 使用口令卡实现一次性口令认证。

如图3.5所示是应用于网上银行的口令卡,卡的一面以矩阵的形式印有若干字符串,初始时有覆膜。不同的账号对应的口令卡不同,用户在网上银行进行对外转账或缴费等支付交易时,网上银行系统随机给出一组口令卡坐标,客户根据坐标从卡片中找到对应的口令组合并输入到网上银行系统中,网上银行系统据此来对用户进行身份鉴别。

基于口令卡的鉴别过程如下。

(1) 认证端系统的数据库中存放用户的账号和对应的口令卡内容。

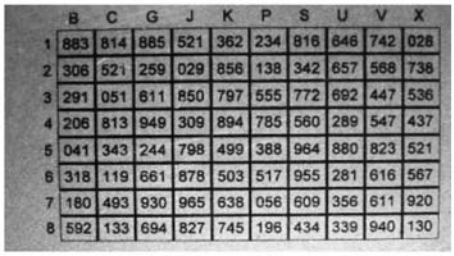
(2) 用户向认证端发送认证请求,该请求中包含用户标识符信息。

(3) 认证端检查用户标识符是否有效,如果无效,则向用户返回错误信息,结束鉴别过程; 如果有效,则进入下一步。

(4) 认证端生成两个随机坐标(也称挑战),并通过网络传输给用户端。

(5) 用户根据坐标利用口令卡查出对应的6位口令,并将查找的结果发送给认证端。

(6) 认证端利用相同的口令卡验证用户发送过来的口令,如果一致则鉴别通过,否则返回鉴别失败信息。



	B	C	G	J	K	P	S	U	V	X
1	883	814	885	521	362	234	816	646	742	028
2	306	521	259	029	856	138	342	657	568	738
3	291	051	611	850	797	555	772	692	447	536
4	206	813	949	309	894	785	560	289	547	437
5	041	343	244	798	499	388	964	880	823	521
6	318	119	661	878	503	517	955	281	616	567
7	180	493	930	965	638	056	609	356	611	920
8	592	133	694	827	745	196	434	339	940	130

图3.5　口令卡

采用口令卡的认证方式,用户每次输入不同的动态口令,防止了重放攻击。用户只要保管好手中的口令卡,就能较好地确保资金交易的安全,但是口令卡所提供的一次性口令是有限的,一旦口令卡中的口令使用完后,需要重新换卡。该方法对用户端要求比较低,不要求用户端进行数据加密,攻击者经过网络嗅探,可重构口令卡坐标数据,这种认证方式安全系数较低,因此,一般口令卡对网上交易有金额限制,如果要进行大额交易,建议使用安全性更强的U盾之类的口令令牌。

上述一次性口令产生机制比较简单,动态口令数目有限,并且传输过程中没有进行加密处理,更为安全的一次性口令是以密码学为基础产生的。根据动态因素的不同,主要分为两种实现技术: 同步认证技术和异步认证技术。其中,同步认证技术又分为基于时间同步的认证技术和基于事件同步的认证技术; 异步认证技术即为挑战/应答认证技术。

1. 基于时间同步的认证技术

在一次性口令生成机制中,时间同步方案原理较为简单,该方案产生一次性口令的动态因素是登录时间,为了使服务端能产生跟用户相同的口令以验证用户的身份,该方案要求用户和认证服务器的时钟必须严格一致,用户持有称为动态口令牌的专用硬件,令牌内置电源、同步时钟、密钥和机密算法,如图3.6所示。时间令牌根据同步时钟和密钥每隔一个时间单位(如1min)产生一个动态口令,将用户登录时输入动态令牌显示的当前口令发送给认证服务器,认证服务器根据当前时间和密钥副本采用相同的算法计算出口令,最后将认证服务器计算出的口令和用户发送的口令相比较,得出授权用户的结论。

图3.6　动态口令牌

除了采用动态令牌硬件产生一次性口令之外,随着手机的普及,出现了手机令牌方式,手机令牌是一种手机客户端软件,基于时间同步方式,每隔30s

产生一个随机6位动态密码,口令生成过程不产生通信及费用,具有使用简单、安全性高、低成本、无须携带额外设备等优势。

时间同步机制具有操作简单、携带方便等优点,使用该方案的难点在于需要解决好网络延迟等不确定因素带来的干扰,使口令在生命周期内顺利到达认证系统。

2. 挑战/应答方式

应用最广泛的一次性口令实现方式是挑战/应答(Chanllenge/Response)方式,该方式的基本原理为每次认证时,服务器随机产生一个挑战发送给客户端,客户端基于挑战给出应答,服务器通过验证应答的有效性来判断客户端身份的合法性。最为经典的挑战/应答方案是S/Key协议,S/Key口令认证方案分为两个过程:注册过程和认证过程。注册过程只执行一次,而认证过程则在用户每次登录时都要执行。

1) S/Key注册过程

步骤1:新用户选择将在服务器上注册的用户名ID、口令PW,并提交给服务器,服务器为该ID选择随机种子Seed和最大迭代数n,并将Seed和n发送给用户。

步骤2:用户收到Seed和n后,对P_0=Seed||PW进行n次Hash运算$H^n(P_0)$,并将计算结果通过安全的信道发送给认证服务器。

步骤3:认证服务器收到$H^n(P_0)$后,将$H^n(P_0)$、最大迭代次数n与对应的用户ID保存在认证数据库。

在完成注册工作后,用户只需记住其登录ID和口令PW即可。

2) S/Key认证过程

S/Key认证过程如图3.7所示。

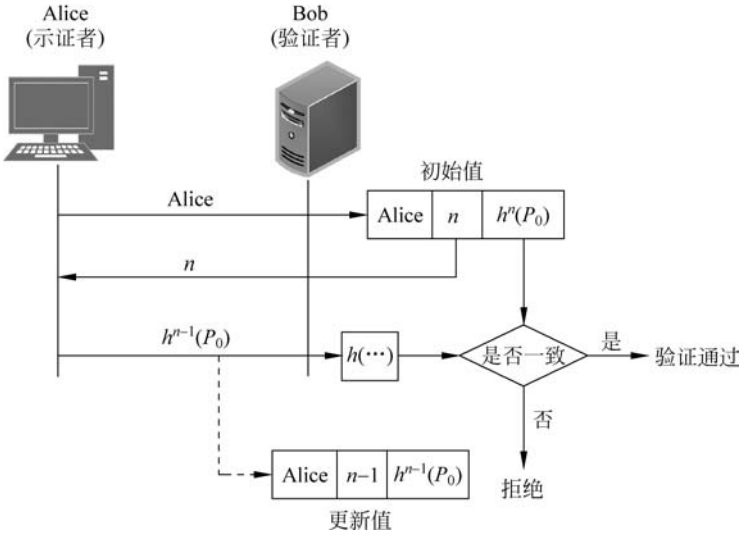


图3.7 S/Key认证过程

当用户第1次登录时,在认证服务器上当前保存的认证数据有用户ID、$H^n(P_0)$和迭代次数n,第1次登录认证过程如下。

步骤1：用户发送登录请求，并将ID发送给服务器，服务器从数据库中取出迭代次数 n 传送给用户。

步骤2：用户收到 n 后，计算 $H^{n-1}(P_0)$，并将结果发送给认证服务器。

步骤3：认证服务器收到认证数据后，计算 $H(H^{n-1}(P_0))$，并与数据库中保存的 $H^n(P_0)$ 相比较。若两者相同，则认证通过，用户成功登录，服务器将收到的 $H^{n-1}(P_0)$ 替换数据库中的 $H^n(P_0)$，迭代数修改为 $n-1$，以便下一次认证时使用；否则，认证失败，服务器拒绝用户的登录请求。

S/Key口令序列认证方案在认证过程中用户传送的口令每次都不同，所以该方案能满足最初的抵御重放的目标。但是S/Key口令认证方案存在“小数攻击”“协议破坏攻击”“内部人员攻击”等形式的网络攻击。

小数攻击是当用户向服务器请求认证时，黑客截取服务器传来的种子和迭代值，修改迭代值为较小值，并假冒服务器，将得到的种子和较小的迭代值发送给用户。用户利用种子和迭代值计算一次性口令，黑客在此截取用户传来的一次性口令，并利用已知的单向散列函数依次计算较大迭代值的一次性口令，就可以获得该用户后继的一系列口令，进而在一段时间内冒充合法用户而不被察觉。此外，S/Key口令序列认证方案在执行性能方面还存在运算量大、需要多次散列计算、当迭代次数较小时需要重新进行初始化等不足。

当前挑战应答机制的实现主要是基于对称密码或非对称密码实现，例如，采用对称密码实现挑战应答一次性口令机制的基本工作过程如图3.8所示。

图3.8　挑战/应答的基本工作过程

(1) 认证请求。用户端首先向认证端发出认证请求。

(2) 挑战(或质询)。认证端产生一个随机数 X(称为挑战)发送给客户端。同时，认证端根据用户ID取出对应的密钥 K，在认证端用加密引擎对发送给客户机的随机数进行运算，得到运算结果Es。

(3) 应答。客户端程序利用用户和认证端共享的密钥 K 对发送过来的随机数进行加密得到运算结果Ev，并将此结果作为认证数据发送给认证端。

(4) 鉴别结果。认证端比较Ev、Es是否相同，如果相同则该用户为合法用户。

3.4　基于智能卡的认证方式

利用用户所拥有的东西进行身份认证相对复杂，成本高，但这种方法的安全性也比较高，它需要借助于一些物理介质，如磁卡、智能卡等。

磁卡是曾经得到广泛应用的一种用以证实个人身份的手段，由于磁卡仅有数据存储能力，而无数据处理能力，没有对其记录的数据进行保护的机制，因此伪造和复制磁卡比较容易。随着微处理器的发展，出现了智能卡。

智能卡又称为CPU卡，是一种嵌有单片机芯片的IC卡。在外形上，它将一个集成电

路芯片镶嵌于塑料基片中,封装成卡的形式,与覆盖磁条的磁卡相似。智能卡上的单片机芯片包含CPU、EPROM、RAM、ROM和芯片操作系统(Chip Operating System,COS),不仅具有读写和存储数据的功能,而且能对数据进行处理,因此,智能卡被称为最小的个人计算机。目前,智能卡在许多应用领域取代了磁卡。

单用智能卡作为用户的身份凭证仍有不足之处,如果智能卡丢失,那么捡到卡的人就可以假冒真正的用户。因此需要额外的且在智能卡上不具有的信息进行辅助认证,这种信息通常采用个人识别号(Personal Identification Number,PIN)。在验证过程中,验证者不但要验证持卡人的卡是真实的卡,同时还要通过PIN码来验证持卡人的确是他本人。采用这种双因素认证机制进一步提升了身份认证的可靠性。

智能卡体积小,方便携带,可以在任何地点进行电子交易,智能卡的读卡器也越来越普遍,有USB型的,也有PC型的,在Windows终端上也可以设置智能卡插槽。

目前,在网上银行中采用的高级别安全工具USBKey是当前比较流行的智能卡身份认证方式。USBKey结合了现代密码学技术、智能卡技术和USB技术,是新一代身份认证产品,具有以下特点。

(1) 双因素认证。每一个USBKey都具有硬件PIN码保护,用户只有同时取得USBKey和用户PIN码,才能登录系统。

(2) 带有安全存储空间。USBKey具有8~128KB的安全数据存储空间,可以存储数字证书、用户密钥等秘密数据,对该空间的读写操作必须通过程序实现,用户无法直接读取,且用户私钥是无法导出的,杜绝了复制用户数字证书或身份信息的可能性。

(3) 硬件实现加密算法。USBKey内置CPU或者智能卡芯片,可以实现PKI体系中使用的数据签名、加解密等各种算法,加解密运算在USBKey内进行,保证了用户密钥不会出现在计算机内存中,从而杜绝了用户密钥被黑客截取的可能性。

USBKey身份认证系统有以下两种模式。

(1) 基于挑战-应答的双因素认证方式。

先由客户端向服务器发出一个验证请求,服务器接收到此请求后生成一个随机数(挑战)并通过网络传输给客户端,客户端将收到的随机数通过USB接口提供给智能卡的计算单元,由计算单元使用该随机数与存储在安全存储空间中的密钥进行运算得到一个结果(应答)作为认证证据传给服务器。与此同时,服务器也使用该随机数与存储在服务器数据库中的该客户密钥进行相同运算,如果服务器的运算结果与客户端回传的响应结果相同,则认为客户端是一个合法用户。这种模式的挑战/应答认证方式只能对客户端的身份进行认证,无法实现对服务端的身份认证。

(2) 基于数字证书的认证方式。

随着PKI技术的成熟,许多应用开始使用数字证书进行身份认证与数字加密。数字证书是由权威公正的第三方机构即CA中心签发的,以数字证书为核心的加密技术,可以对网络上传输的信息进行加密和解密、数字签名,确保网上传递信息的机密性、完整性,以及交易实体身份的真实性,签名信息的不可否认性,从而保障了网络应用的安全性。USBKey作为数字证书的存储介质,可以保证数据证书不被复制,并可以实现所有数字证书的功能。

USBKey 中预置了加密算法、摘要算法、密钥生成算法等，可利用密钥生成算法首先为用户生成一对公/私钥，私钥保存在 USBKey 中，公钥可以导出向 CA 申请生成数字证书，数字证书也保存在 USBKey 中，在进行客户端身份认证时，客户端向服务器发送数字证书，服务端利用 CA 的公钥验证数字证书的真实性，完成对客户端身份的认证。客户端可也要求服务端发送数字证书以验证服务端的真实身份。

【例 3-6】 U 盾，即中国工商银行 2003 年推出的客户证书 USBKey，是中国工商银行为客户提供的网上银行业务的高级别安全工具。该产品采用的信息安全技术，核心硬件模块由 CPU 及加密逻辑、RAM、ROM、EEPROM 和 I/O 五部分组成，是一个具有安全体系的小型计算机。除了硬件，安全实现完全取决于技术含量极高的智能卡芯片操作系统(COS)，该操作系统就像 DOS、Windows 等操作系统一样，管理着与信息安全密切相关的各种数据、密钥和文件，并控制各种安全服务。从技术角度看，U 盾是用于网上银行电子签名和数字认证的工具，基于 PKI 技术，采用 1024 位非对称密钥算法对网上数据进行加密、解密和数字签名，确保网上交易的保密性、真实性、完整性和不可否认性。USBKey 具有硬件真随机数发生器，密钥完全在硬件内生成，并存储在硬件中，能够保证密钥不出硬件，硬件提供的加解密算法完全在加密硬件内运行。

网上银行用户发起业务指令(例如支付 5000 元)时，被要求插入 USBKey，同时输入与之相对应的 PIN 码进行身份验证，验证成功后，客户端计算机将敏感的网上银行业务指令传输到 USBKey 中，USBKey 中的芯片操作系统先将指令进行 Hash 运算以形成数字摘要，同时用 USBKey 中的私钥对数字摘要加密以产生数字签名。另外，客户端随机产生 DES 密钥，用 DES 密钥对刚才生成的数字签名、网上银行业务指令原文进行加密，同时用服务端的公钥对 DES 密钥加密，将这两部分加密信息通过 USB 接口提交到客户端系统，进而提交给网上银行服务器系统。网上银行服务器收到客户端信息后，使用自己对应的私钥解密得到 DES 密钥，然后用 DES 密钥解密得到数字签名及指令，之后用客户端公钥验证数字签名，可有效防止指令中途被篡改，并且能够保证用户身份不可抵赖性。客户端 USBKey 工作流程如图 3.9 所示。

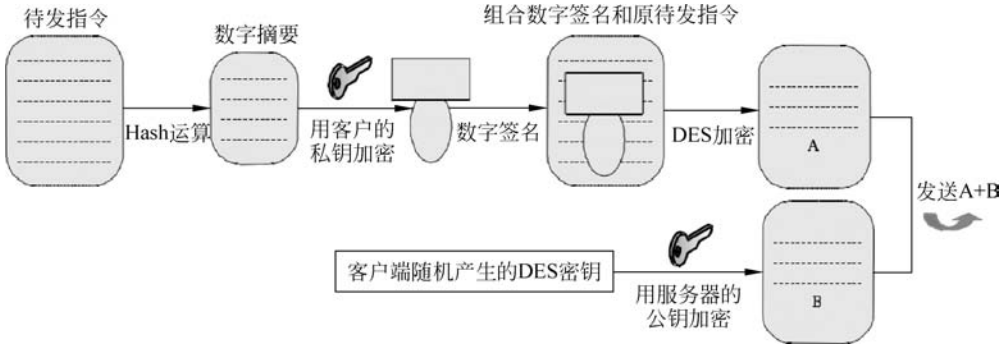


图 3.9　USBKey 工作流程

从技术上来讲，USBKey 算得上是目前最为安全的网上银行认证工具。当前几乎所有的网上银行都有采取这种基于硬件的数字证书身份认证的方法来保护用户网上银行交

易安全。不过由于内置了芯片,USBKey 的制作成本也相对较高,一个功能完备的 USBKey 的市场价格大约是 50～60 元。

利用 USBKey 方式进行身份认证尽管看起来很完美,但实际上还存在一些安全问题。USBKey 认证系统主要存在以下两个安全漏洞:一是黑客完全有能力截获从计算机上输入的静态 PIN 码,用户没有及时取走 USBKey 的时候,黑客便可利用 PIN 码来取得虚假认证,进行转账操作;二是在用户发出交易指令后,通过 USB 接口传送到 USBKey 之前存在一段安全真空期,传输过程中的信息没有受到任何保护,黑客此时可悄无声息地篡改用户指令,而 USBKey 还会坚定地保护篡改后的指令。交易完成后,用户才会发现刚才转账的过程出现了差错,这时损失已经不可挽回。

3.5 基于生物特征的认证方式

传统的用户身份认证机制有许多缺点,目前,虽然从最早的"用户名+口令"方式过渡到了在网银中广泛使用的 USBKey 方式,但它仍然有许多缺点,首先需要随时携带智能卡,其次它也容易丢失或失窃,补办手续烦琐,并且仍然需要用户出具能够证明身份的其他文件,使用很不方便。直到生物识别技术得到成功应用,身份认证机制才真正回归到了对人类最原始的特性上。基于生物特征的认证技术具有传统的身份认证手段无法比拟的优点。采用生物鉴别技术,可不必再记忆和设置密码,使用更加方便。生物特征鉴别技术已经成为一种公认的、最安全的和最有效的身份认证技术,将成为 IT 产业最为重要的技术革命。

基于生物特征的身份认证方式是利用人体固有的生理特征或行为特征来进行身份识别或验证。生理特征与生俱来,多为先天性的,如指纹、眼睛虹膜、声音、人脸等,行为特征则是习惯使然,多为后天性的,如笔迹、步态等。这些生物特征具有唯一性、稳定性和难以复制性,采用生物认证技术具有很好的安全性、可靠性和有效性,与传统的身份确认手段相比,具有无法比拟的优点。近几年来,全球的生物识别技术已从研究阶段转向应用阶段,对该技术的研究和应用如火如荼,前景十分广阔。

生物识别的核心在于如何获取这些生物特征,并将之转换为数字信息,存储于计算机中,利用可靠的匹配算法来完成验证和识别个人身份。基于生物特征的认证机制一般过程如下。

(1) 认证系统先对用户的生物特征进行多次采样,然后对这些采样进行特征提取,并将平均值存放在认证系统的用户数据库中。

(2) 鉴别时,对用户的生物特征进行采样,并对这些采样进行特征提取。通过数据的保密性和完整性保护措施将提取的特征发送到认证系统,并在认证系统上解密用户的特征。

(3) 比较步骤(1)和步骤(2)的特征,如果特征匹配达到近似要求,认证系统向用户返回鉴别成功的信息,否则返回鉴别失败的信息。

与传统身份鉴别相比,生物识别技术具有以下特点。

(1) 随身性：生物特征是人体固有的特征，与人体是唯一绑定的，具有随身性。

(2) 安全性：人体特征本身就是个人身份的最好证明，满足更高的安全需求。

(3) 唯一性：每个人拥有的生物特征各不相同。

(4) 稳定性：生物特征如指纹、虹膜等，不会随时间等变化。

(5) 广泛性：每个人都具有这种特征。

(6) 方便性：生物识别技术不需要记忆密码与携带使用特殊工具，不会遗失。

(7) 可采集性：选择的生物特征易于测量。

基于生物特征的认证系统安全性高，但成本也高。另外，技术上的发展也证明，这些生物特征在安全上并不是无懈可击的。例如，有研究者在2002年发现可以用凝胶铸成的指纹模子来瞒骗指纹识别器。同时生物认证系统并不具有普适性，例如，人的视网膜图案是独一无二的，用视网膜做认证的确十分可靠，但是由于需要使用聚光灯来获取独特的眼球后面的血管图，并不是每个登录系统的人都愿意用一个仪器来扫描自己的眼睛，因为人的肉眼暴露在这种视网膜扫描装置下会觉得很不舒服，所以这样的系统只在一些高安全环境中实施。

3.6 身份认证协议

在网络中，通常是服务器要验证用户的身份，称为单向认证，但是为了防止网络钓鱼等假冒服务器的现象发生，客户端有时也需要验证服务器的身份，这就需要进行双向认证，下面依次介绍采用挑战/应答方式的认证过程。

3.6.1 单向认证

单向认证是通信的一方认证另一方的身份，例如，服务器在向用户提供服务之前，先要认证用户是否是这项服务的合法用户，但是不需要向用户证明自己的身份，单向认证可采取普通的口令认证，也可用对称密码或非对称密码体制实现。

1. 采用对称密码体制实现单向认证

假设Bob要验证Alice的身份，Alice首先向Bob提出认证请求，Bob产生一个随机数R_B发送给Alice，Alice用双方共享的密码$K_{A\text{-}B}$加密该随机数后发送给Bob，Bob如果能用相同的密码解密得到R_B，则Alice的身份得到Bob的验证，因为密码$K_{A\text{-}B}$是Alice和Bob共享的秘密，其他任何人没有该密钥就无法构造出R_B的密文，如图3.10所示。

采用对称密码体制进行双单向身份认证的方式有很多，上例只是其中的一种。

2. 采用非对称密码体制实现单向认证

这里Bob要验证Alice的身份，Alice首先向Bob提出认证请求，Bob产生一个随机数R_B并用Alice的公钥加密后发送给Bob，Alice用私钥解密得到随机数后发送给Bob，Bob验证发送过来的随机数，则Alice的身份得到Bob的验证，如图3.11所示。

采用非对称密码体制进行双单向身份认证的方式有很多，上例只是其中的一种。

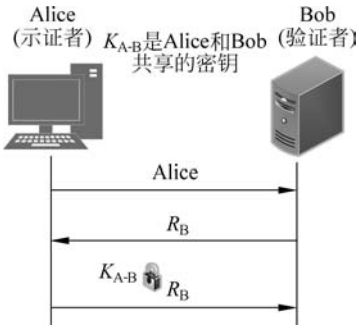


图 3.10 采用对称密码体制实现单向认证

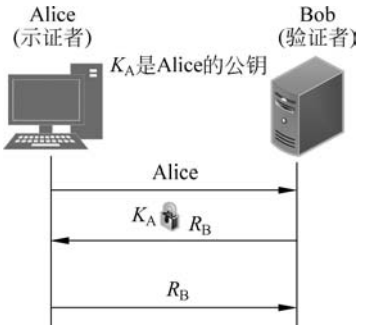


图 3.11 采用非对称密码体制实现单向认证

3.6.2 双向认证

双向认证需要通信双方互相认证对方的身份,同样可以采用普通的口令认证方式,这里主要讨论采用对称密码体制和非对称密码体制实现双向身份认证。

1. 用对称密码体制实现双向认证

Alice 首先向 Bob 提出认证请求,Bob 产生一个随机数 R_B 发送给 Alice,Alice 为了同时验证 Bob 的身份,也产生一个随机数 R_A,串接在 R_B 后并用双方共享的密钥 K_{A-B} 加密,密文发送给 Bob,Bob 由于拥有密钥 K_{A-B},可以解密,如能分解出 R_B,则验证通过 Alice 的身份,同时为了验证自己的身份(也即拥有密钥 K_{A-B}),他将 R_A、R_B 调换顺序串接后用共享密钥加密后发送给 Alice,Alice 解密后如能分离出 R_A,则验证通过 Bob 的身份,如图 3.12 所示。

采用对称密码体制进行双向身份认证的方式有很多,上例只是其中的一种。

2. 用非对称密码体制实现双向认证

Alice 首先产生一个随机数 R_A,用 Bob 的公钥加密后发送给 Bob,Bob 接收到后用自己的私钥解密得到 R_A,产生一个随机数 R_B 串接在 R_A 后用 Alice 的公钥加密后发送给 Alice,Alice 用自己的私钥解密,如果能分离出 R_A 则验证通过 Bob 的身份,同时为了验证自己的身份将解密分离出的 R_B 发送给 Bob,Bob 接收到之后可以验证 Alice 的身份,如图 3.13 所示。

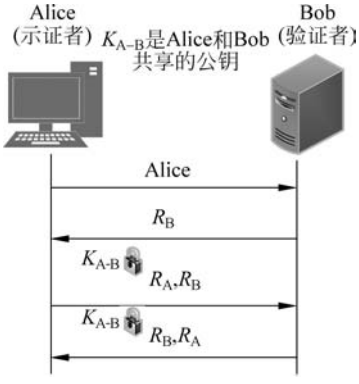


图 3.12 采用对称密码体制实现双向认证

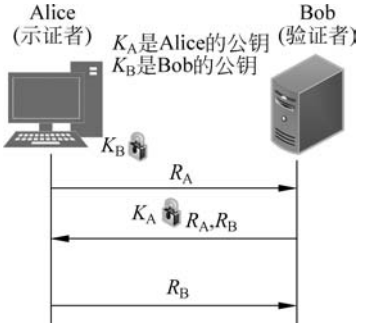


图 3.13 采用非对称密码体制实现双向认证

采用公钥密码体制进行双向身份认证的方式有很多，上例只是其中的一种。

3.6.3　可信的第三方认证

可信的第三方认证也是一种通信双方相互认证的方式，但是认证过程必须借助于一个双方都能信任的第三方，一般而言可以是政府机构或其他可信赖的机构。当两端欲进行通信时，彼此必须先通过信任的第三方认证，然后才能互相交换密钥，而后进行通信。借助于信任第三方的认证协议相当多，各有各的特色与优缺点，其中一个最著名的例子就是由美国麻省理工学院提出的 Kerberos 协议。

Kerberos 是在 20 世纪 80 年代中期作为美国麻省理工学院“雅典娜计划”(Project Athena)的一部分开发的，其前三个版本仅用于内部使用，第四个版本得到了广泛应用，第五版于 1989 年设计，目前已经作为 Kerberos 的标准协议。Microsoft 的 Windows 2000 之后的操作系统、Red Hat Linux 等都支持 Kerberos 认证协议，此外，该协议也应用于电子商务、无线局域网等领域。下面来看一下 Kerberos 协议的演进过程。

最简单的第三方认证方案如图 3.14 所示。在这个方案中，在客户 C 和应用服务器 V 之间增加了一个认证服务器(Authentication Server，AS)，用于对用户进行统一的认证和授权。在认证前，首先要求系统中所有的安全实体包括所有的客户 C 和服务器 V 都需要在 AS 上注册，每个客户都有一个口令 P_c，每个服务器与认证服务器共享一个密钥 E_{kv}。整个协议过程分为 3 个步骤。

(1) C→AS：$ID_c \| P_c \| ID_v$

(2) AS→C：Ticket

$Ticket = E_{kv}(ID_c \| AD_c \| ID_v)$

(3) C→V：$ID_c \| Ticket$

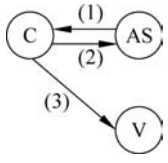


图 3.14　一种简单的第三方认证方案

首先看协议的第一步交换过程，客户向认证服务器发送认证请求，包括客户标识 ID_c、口令 P_c 及想要访问的服务器标识 ID_v，符号 ‖ 表示级联。认证服务器验证用户的口令、检查该用户是否有权访问服务器 ID_v，如果认证通过，则发送给客户一张票据，票据内容包括客户标识 ID_c、客户的网络地址 AD_c 以及要访问的服务器标识 ID_v，客户向服务器出示这张票据，服务器要验证这张票据是否是 AS 签发的，为了防止票据的伪造，该票据必须用 AS 与服务器共享密钥 E_{kv} 加密。通过这张票据，AS 要向 V 说明 AS 已经认证过 C 的身份，并且 C 确实具有访问 V 的权限。

这种认证方案存在以下安全隐患。

(1) 口令是明文传递的，容易被窃听。

(2) 客户访问不同服务器需要多次身份认证。

(3) 认证服务器 AS 同时要进行认证和授权服务。

(4) 票据 Ticket 不能抵御重放攻击。

在刚才的认证方案中认证服务器 AS 兼任认证和授权两项功能，可以在认证系统中增加一个票据许可服务器(Ticket-Granting Server，TGS)，这样，AS 服务器专门进行身份认证，票据许可服务器专门进行服务授权，认证服务器并不直接向客户发放访问应用服

务器的票据,而是由 TGS 进行发放。因此在认证系统中存在两种票据:“服务许可票据”和“票据许可票据”。下面看一下改进方案,如图 3.15 所示,整个协议过程分为五步。

(1) C→AS : $ID_c \| ID_{tgs}$

(2) AS→C : $E_{kc}[Ticket_{tgs}]$

$Ticket_{tgs}=E_{ktgs}[ID_c \| AD_c \| ID_{tgs} \| TS_1 \| Lifetime1]$

(3) C→TGS : $ID_c \| ID_v \| Ticket_{tgs}$

(4) TGS→C: $Ticket_v$

$Ticket_v=E_{kv}[ID_c \| AD_c \| ID_v \| TS_2 \| Lifetime2]$

(5) C→V : $ID_c \| Ticket_v$

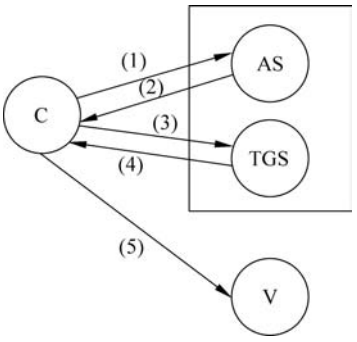


图 3.15 一种改进的第三方认证方案

客户 C 首先向 AS 申请一个票据许可票据(用来证明用户身份的票据),请求报文中包含用户标识 ID_c 及 TGS 的标识 ID_{tgs}。AS 向客户 C 发回一个使用密钥 K_c 加密后的票据,其中的密钥 K_c 是根据用户口令计算出来的,由于只有合法用户才拥有正确的口令,所以只有合法用户才能恢复出票据,这种方法既可认证用户身份又可避免用户口令在网络中明文传输带来的危险。这张票据是交给 TGS 以确认用户的身份已经被 AS 认证过,所以票据许可票据需要用 AS 和 TGS 预先共享的密钥 E_{ktgs} 进行加密。票据许可票据是可重用的,这样用户访问不同的服务器不需要多次输入口令,在票据许可票据中包含客户标识 ID_c、客户网络地址 AD_c、TGS 的标识 ID_{tgs}、时间戳 TS1 和使用时限 Lifetime1。

接下来,客户 C 向 TGS 发送票据许可票据、客户标识 ID_c 以及要访问的服务器标示 ID_v,TGS 在验证票据后查看该用户是否有权访问服务器,如果有访问权限则发送一张服务许可票据,服务许可票据最终是要由服务器验证的,因而该票据用 TGS 和服务器 V 预先共享的密钥 E_{kv} 进行加密,并且该票据在一段时间内可以重用,因而票据包含时间戳和使用时限,说明票据的有效性。

最后,客户 C 向服务器发送服务许可票据,服务器验证票据后即可向客户 C 提供服务。

上述认证方案存在以下安全隐患。

(1) 票据有使用时限,但客户端和服务器系统缺乏时钟同步。

(2) 没有为 C 和 TGS、C 和 V 之间的会话提供会话密钥。

(3) 服务器可能是假冒的。

(4) 票据许可票据和服务许可票据仍可能被重放。

Kerberos 版本 4 较好地解决了上述问题,得到了广泛应用,下面重点介绍该版本的协议过程,如图 3.16

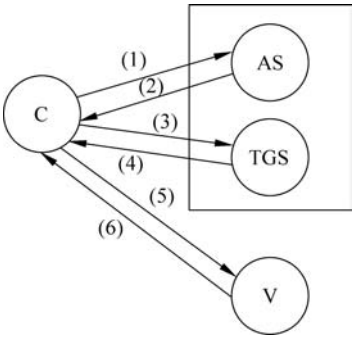


图 3.16 Kerberos v4 认证方案

所示。Kerberos协议建立在对称密码体制基础之上，在协议开始之前，AS和TGS之间、TGS和V之间必须首先完成信任关系的建立，在对称密码体制下，这种信任关系是通过AS和TGS之间、TGS和V之间共享密钥建立的，这些密钥采用人工或其他安全方式分发。整个协议过程分为三个阶段：身份认证、获得服务授权、获取服务。

(1) C→AS：$ID_c \| ID_{tgs} \| TS1$

(2) AS→C：$E_{kc}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| LT_2 \| Ticket_{tgs}]$

$Ticket_{tgs}=E_{ktgs}[K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| LT_2]$

(3) C→TGS：$ID_v \| Ticket_{tgs} \| AU_c$

$AU_c=E_{kc,tgs}[ID_c \| AD_c \| TS_3]$

$Ticket_{tgs}=E_{ktgs}[K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| LT_2]$

(4) TGS→C：$E_{kc,tgs}[K_{c,v} \| ID_v \| TS_4 \| Ticket_v]$

$Ticket_v=E_{kv}[K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| LT_4]$

(5) C→V：$Ticket_v \| AU_c$

$AU_c=E_{kc,v}[ID_c \| AD_c \| TS_5]$

(6) V→C：$E_{kc,v}[TS_5+1]$

第一个阶段为身份认证，包括步骤(1)和(2)，在客户和认证服务器之间运行。客户C首先向AS申请一张票据许可票据(用来证明用户身份的票据)，请求报文中包含用户标识ID_c、TGS标识ID_{tgs}，以及时间戳TS_1，ID_{tgs}说明申请的票据是用来跟某个TGS沟通的，TS_1是用来进行时钟同步的。AS向客户C发回的信息使用密钥K_c加密，其中的密钥K_c是根据用户口令计算出来的，确保只有合法用户才能解密，加密的消息包括C和TGS通信的会话密钥$K_{c,tgs}$、时间戳和有效期、票据许可票据，通过这种方式可以安全地把C和TGS通信的会话密钥分发给C，票据许可票据需要用TGS和AS共享密钥加密，以使得TGS可以验证该票据是AS颁发的，同时在票据许可票据中还包含C和TGS通信的会话密钥$K_{c,tgs}$，由于票据最终会发送给TGS，因而采用这种方式可以安全地把会话密钥分发给TGS。

第二个阶段为获得服务授权，包括步骤(3)和(4)，在客户和TGS之间运行。客户C向TGS发送票据许可票据、要访问的服务器标识ID_v、认证符AU_c，其中，AU_c是用会话密钥加密用户ID、网络地址AD以及时间戳形成的用户认证符，通过这种方式可以防范票据许可票据被重放，因为即使攻击者获取了票据许可票据，由于没有会话密钥，因而无法基于新的时间戳加密生成正确的认证符AU_c。

TGS在解密票据后可以获取会话密钥$K_{c,tgs}$，利用会话密钥解密AU_c的内容，从而对用户的身份进行验证，验证通过后查看用户是否有权限访问服务器ID_v，如果有访问权限则生成一张服务许可票据，同时生成一个客户C与服务器V的会话密钥$K_{c,v}$，利用C和TGS的会话密钥$K_{c,tgs}$加密后发送给C，将客户C和服务器V之间的会话密钥安全分发给客户C。服务许可票据最终是发送给服务器进行验证的，因而用TGS和服务器之间预先共享的密钥E_{kv}进行加密，这样服务器可以验证票据是否是TGS颁发的，同时在票据中还包客户C和服务器V的会话密钥$K_{c,v}$，将会话密钥安全分发给客户C。

第三个阶段为获取服务，包括步骤(5)和(6)，在客户和应用服务器之间运行。客户C

将 TGS 发送过来的消息解密后得到和服务器 V 的会话密钥 $K_{c,v}$ 和服务许可票据,客户 C 向服务器发送服务许可票据和认证符 AU_c,AU_c 的目的是防重放攻击。

最后服务器在接收到客户 C 发送的服务许可票据和认证符后进行验证,验证通过后对时间戳进行简单变换后加密发送给客户 C,用以验证服务器的身份,防止服务器伪造。

Kerberos v4 版本实现了用户仅需一次登录就可凭票据访问各类服务,具有单点登录的效果,不仅实现了服务器对客户端的认证,也实现了客户端对服务器的认证。但是也存在以下问题:一是加密系统依赖性问题,仅能使用 DES 加密算法;二是票据有效期问题,最长有效期为 21h,这对某些应用是不够的;三是使用时间戳应对重放攻击,这要求全网时钟同步,比较难以实现;四是整个协议加密次数比较多,会影响整个协议的效率;五是用户口令如果设置简单,容易遭受穷举攻击。基于这些局限性,Kerberos v5 在第 4 版协议模型的基础上进行了改进,提供了比第 4 版更完善的认证机制。

3.7 零知识证明

通常的身份认证都要求传输口令或身份信息,如果不透露这些信息,身份也能得到证明就好了,这就需要零知识证明技术。

这里假设 P 为示证者,V 为验证者,P 试图向 V 证明自己知道某消息,一种方法是 P 说出这一消息使 V 相信,这样 V 也知道了这一消息,这是基于知识的证明;另一种方法是使用某种有效的数学方法,使得 V 相信他掌握这一信息,却不泄露任何有用的信息,这种方法称为零知识证明。

解释零知识证明概念最经典的例子是 1990 年 Louis C. Guillou 和 Jean-Jacques Quisquater 提出的"洞穴问题",如图 3.17 所示。洞穴里面有一个秘密咒语,只有知道咒语的那些人才能打开 C 和 D 之间的密门。C 与 D 之间的竖线为一扇门,P 知道打开这扇门的咒语,现在 P 要向 V 证明他知道这个咒语,按照传统的做法是 P 把这个咒语告诉 V,然后 V 用这个咒语去打开那扇门,如果能打开说明这个咒语是正确的,那么就可以验证 P 的身份。这样做虽然成功地证明了 P 的身份,但是也泄露了这个咒语,那么 V 可以冒充 P 或者 V 可以把这个咒语透露给第三者,从而使得这个系统存在安全隐患。

以下是用零知识证明理论来证明 P 的身份。

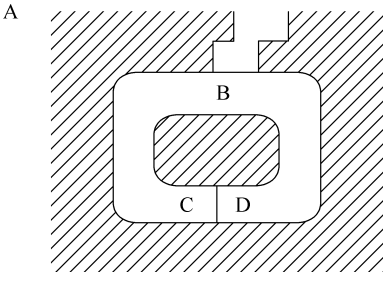


图 3.17 洞穴问题

(1) V 站在 A 点,让 P 进入洞内,这样保证 V 不知道 P 是从哪边进入洞内的。(从左边进入到达 C 点,从右边进入则到达 D 点)。

(2) 当 P 到达 C 点或 D 点以后,V 走到 B 点。

(3) V 向 P 喊叫,要他从左通道或右通道出来。

(4) P 按照 V 的要求从 V 指定的方向出来,如果有必要他就用咒语打开密门。

(5) P 和 V 重复步骤(1)~(4)n 次。

若 P 不知咒语,只有 1/2 的机会猜中 V 的要求,

协议执行 n 次，则只有 2^{-n} 的机会完全猜中，若 $n=16$，则若每次均通过 V 的检验，则 V 受骗的机会仅为 1/65 636。如果 n 足够大，几乎就可以证明 P 是知道这个咒语的并且 V 并没于得到任何关于这个咒语的信息。

最早出现的零知识身份证明协议是 Feige-Fiat-Shamir 协议（FFS 协议），在 Feige-Fiat-Shamir 协议中，可信赖第三方选定 $m=p\times q$，其中，p，q 为两个大素数，m 为 512 b 或者为 1024b。通信双方共享 m，再由可信赖第三方实施公钥私钥的分配，他产生 k 个随机数 $v_1,v_2,\cdots,v_k$（要求 $v_i^{-1} \bmod m$ 存在，$i=1,2,\cdots,k$），且使 v_i 为模 m 的平方剩余，即 $x^2=v_i(\bmod m)$，v_i 为公钥，计算 $s_i=\sqrt{v_i^{-1}} \bmod m$，$s_i$ 作为示证方的私钥。（这里 P 作为示证方，V 作为验证方。）

身份验证协议如下。

（1）用户 P 取随机数 $r(r<m)$，计算 $x=(r^2)\bmod m$，将 x 发送给 V。

（2）V 发送挑战信息串 $b_1,b_2,\cdots,b_k$（b_i 为 0 或者 1）给 P。

（3）P 接受挑战，算出 $y=r\times\prod_{i=1}^{k} vi^{bi} \bmod m$ 作为应答发送给 V。

（4）V 验证 P 的身份，验证 $x=y^2\prod_{i=1}^{k} vi^{bi} \bmod m$，如果相等则受骗概率为 2^{-k}。如果不相等，这个验证过程则终止。

P 和 V 可以重复执行此协议，每次以不同的 r 和 b 串，执行完一次验证过程后 P 能欺骗 V 的概率为 2^{-k}。

Feige-Fiat-Shamir 协议需要经过多次交互才能保证身份认证的安全性，不仅费时而且浪费系统资源，随后又提出了其他的基于零知识证明的身份认证协议，这里不再详细讨论，零知识证明的出现给身份认证带来了一个新方向。

习　　题

一、填空题

1. 身份认证包括_____、_____两个过程。

2. _____是最常见的身份认证方式。

3. 针对弱口令的攻击主要有_____和_____。

4. _____用来抵御重放攻击。

5. 网上银行中采用的身份认证方式有：_____和_____。

6. 不泄露秘密信息进行身份认证的技术称为_____。

7. 身份认证的方式包括利用用户所知道的信息、_____、_____、_____。

8. UNIX 系统在生成口令散列时，需要加入_____。

9. Kerberos 认证中将完成身份认证功能的可信第三方服务器称为_____，将完成服务授权功能的可信第三方服务器称为_____。

10. Kerberos 认证中有两种票据：_____、_____。

二、选择题

1. Windows 系统能设置为在几次无效登录后锁定账号,这可以防止(　　)。
   A. 木马　　B. 暴力攻击
   C. IP 欺骗　　D. 缓存溢出攻击
2. 在以下认证方式中,最常用的认证方式是(　　)。
   A. 基于账户名/口令认证　　B. 基于摘要算法认证
   C. 基于 PKI 认证　　D. 基于数据库认证
3. 以下哪项不属于防止口令猜测的措施?(　　)
   A. 严格限定从一个给定的终端进行非法认证的次数
   B. 确保口令不在终端上再现
   C. 防止用户使用太短的口令
   D. 使用机器产生的口令
4. Kerberos 认证用到以下哪种加密体制?(　　)
   A. 公钥密码体制　　B. 对称密码体制
   C. 散列算法　　D. 异或运算
5. Kerberos 认证中,认证码的作用是(　　)。
   A. 票据防重放　　B. 防止拒绝服务攻击
   C. 对票据颁发者身份认证　　D. 对票据使用者身份认证
6. 有关 Kerberos 认证协议的说法,不正确的是(　　)。
   A. 不支持双向身份认证
   B. 签发的票据都有一个有效期
   C. 与授权机制相结合
   D. 支持分布式网络环境下的认证机制
7. 关于 S/KEY 认证的说法不正确的是(　　)。
   A. 使用了对称密码算法对口令进行加密
   B. 用户登录一定次数后必须重新初始化口令序列
   C. 易遭受小数攻击
   D. 一次性口令认证是一种单向认证
8. 在 Kerberos 认证中,由(　　)完成用户身份认证。
   A. 应用服务器　　B. 认证服务器 AS
   C. 票据许可服务器 TGS　　D. 授权服务器
9. 时间戳可以应对哪种攻击?(　　)
   A. 假冒　　B. 拒绝服务攻击
   C. 篡改　　D. 重放
10. 以下口令设置中符合强口令规则的是(　　)。
   A. ABCABCBC　　B. A,123!b
   C. 1qaz2wsx　　D. 1234321

四、简答题

1. 什么是字典攻击和重放攻击？什么是一次性口令认证？为什么口令加密过程要加入不确定因子？

2. 单机状态下验证用户身份的三种因素是什么？

3. 一次性口令认证实现技术有哪些？

4. 简述基于 PKI 技术体系的 USBKey 认证原理。

5. 自行设计一个利用公钥密码体制实现双向认证的协议。

6. 为防范小数攻击，请阅读相关文献，说明如何改进 S/KEY 一次性口令认证协议。

7. Kerberos v4 版本中认证码的作用是什么？

8. 请阅读相关文档，说一说 Kerberos v5 具有哪些新的安全特性。