目录

第 1 章　概述

计算机的出现，特别是开放式的计算机因特网的普及与发展，使得信息的载体和传播方式发生了根本性的变化，极大地方便了信息的处理与传递，也极大地方便了信息的获取与共享。随着计算机网络广泛应用到社会的各个领域，在信息时代的今天，任何一个国家的政治、军事、经济、外交、商业和金融都离不开信息，科学的发展和技术的进步也离不开信息，信息已成为社会发展的重要战略资源。信息的地位与作用随着信息技术的快速发展，越来越显示出它的重要性。因此对信息的开发、控制和利用，已成为国家利益之间、竞争对手之间争夺的重要内容。相应地，信息安全已成为各国、社会各界关注的焦点，与国家安危、经济发展、社会稳定和战争胜负息息相关。信息安全已成为一个重要的研究领域。

对信息安全的维护需要综合运用管理、法律、技术等多种手段。在技术层面上又需要综合运用身份认证、访问控制、密码、数字签名、防火墙、安全审计、灾难恢复、防病毒和黑客入侵等多种安全技术，并进一步建立信息安全基础设施和构建信息安全的保障体系。

身份认证相当于信息系统的门卫，它识别要求进入系统的每一个用户是否该系统的合法用户，拒绝非法用户进入系统。访问控制则是对进入系统的合法用户对系统资源的访问权进行限制，使用户对资源的使用只能在允许的范围之内。从广义上讲，身份认证也可以看作是一种访问控制。

1.1　信息安全

从人类有信息交流开始，信息安全的问题就存在，但在不同的时期由于信息存储和传输的设备、方式不同，维护信息安全的手段也就不同。早期，计算机诞生之前，人们较多关注的是信息在传递过程中的保密性，使用的也是一些简单的手工加密变换。随着数学、计算机和通信技术的发展，信息的处理和传输能力大大提高，信息安全的含义更为丰富，传统的密码变换已不能满足信息安全的要求。因此必须研究计算机和通信安全的新理论和新技术。

计算机信息系统是由计算机及其相关和配套的设备、设施和网络构成的，是按照一定的应用目标和规则对信息进行采集、加工、存储、传输和检索等处理的

复杂的人机系统。信息系统可能遭受到各种各样的攻击和威胁，而这些攻击和威胁所造成的损失主要体现在系统中信息的安全性和可用性受到了破坏，它往往使得系统中存放的信息被窃取、篡改、破坏、删除和无法传递，甚至整个系统崩溃。20 世纪 80 年代末期，一场计算机病毒危机席卷全球，人们在震惊之余，第一次意识到他们精心构建的计算机系统是如此不堪一击。随着数据库和网络技术的广泛应用，计算机及其网络系统的这种脆弱性暴露得更加充分。计算机犯罪案件迅猛增加，已成为一种社会隐患。

1.1.1 信息系统面临的主要威胁

威胁信息系统安全的因素来自于多个方面，总的来说，可分为人为的恶意攻击和软硬件故障及用户操作失误两类。其中有预谋的人为攻击其威胁程度和防范难度远大于第二类，是系统防范的重点。

据美国联邦调查局的报告，计算机犯罪是商业犯罪中最大的犯罪类型之一，每年计算机犯罪造成的经济损失高达 50 亿美元。加之国际互联网络的广域性和可扩展性，计算机犯罪已成为具有普遍性的国际问题。

对信息系统安全威胁的方式从总体来看主要有以下几种类型：

（1）窃取

合法用户，甚至非法用户（冒充合法用户，进入了系统）未经许可却直接或间接获得了对系统某项资源的访问权，从中窃取了有用的数据或骗取了某种服务，但不对信息作任何修改。这种攻击方式通常被成为被动攻击。

用程序或病毒截获信息是这一类攻击的常见手段。在通信设备或主机中预留程序代码或施放病毒程序，这些程序通过对信息流量进行分析，或通过对信息的破译以获得机密信息，并将有用的信息通过某种方式发送出去。

搭线窃听也是常见的手段，将导线搭到无人监守的网络传输线上进行监听，如果所搭的监听设备不影响网络的负载平衡，网络站点是无法发现的。

对难于搭线监听的可以用无线截获的方式得到信息，通过高灵敏接收装置接收网络站点辐射的电磁波或网络连接设备辐射的电磁波，通过对电磁信号的分析恢复原数据信号从而获得网络信息。

被动攻击不易被发现，原因是它不会导致系统中信息的任何改动，系统的操作和状态也不被改变，留下的痕迹很少，甚至不留痕迹。对付这种攻击的方法主

要是采用加密技术，形成加密通道。

（2）篡改

未经授权的用户成功地获得了对某项资源的访问权后，对信息的全部或部分进行肆意地修改、删除、添加，改变其中内容的次序或形式，改变信息的流向，或者修改程序的功能，改变系统的状态和操作等，破坏信息的完整性、真实性和有效性。某些情形的更改可以用简单的措施检测出来，而一些更精妙的更改却很难发现或检测。

在金融犯罪的案件中，大多是通过修改程序或修改数据达到贪污和欺诈钱财的目的。

（3）伪造

威胁源在未经许可的情形下，在系统中产生出虚伪的数据或服务。例如，电子商务中，不法分子可能希望在网络通信系统中加上假的交易，或者在现有的数据库中增加记录。

伪造信息在网络通信中往往可以使敌方落入陷阱。

（4）拒绝服务

威胁源使系统的资源受到破坏或不能使用，从而使数据的流动或所提供的服务终止。

用户的误操作，软硬件出现故障均可能引起系统内的数据或软件的破坏，因而使得计算机不得不停止工作。隐藏在计算机中具有破坏性的病毒程序被激活后，可能会毁掉系统中某些重要的数据，甚至可能删除系统中的所有数据且使其无法恢复，更严重的可能导致整个系统的瘫痪。又例如，一些不法分子通过断电设置障碍，采用纵火、爆炸、盗窃通信设备等手段导致计算机系统的硬件遭到破坏，使计算机及通信系统无法正常工作。

拒绝服务攻击还可能使网站服务器充斥大量要求回复的信息，消耗网络带宽或系统资源，导致网络或系统不胜负荷以至瘫痪而停止提供正常的网络服务。

（5）重放

在网络通信中重放以前截获到的过时信息，欺骗收方。

（6）冒充

一个实体假冒另一个实体的身份是一种常见的网络攻击手段。

黑客闯入系统的主要途径之一是破译用户的口令，从技术的角度看，防止口令被破译并不困难，但具体执行却较麻烦。经常有些 Web 主管，更不用说用户，同样的口令连续使用好几个月。工作组的成员辞职了，工作组的口令却不改变。甚至，有的用户将自己的口令告诉自己的朋友，这其中就可能有黑客。

（7）抵赖

在网络通信中，用户可能为了自己的利益，在事后否认曾经对信息进行过的生成、签发、接收等行为。

1.1.2 信息系统的脆弱性

以计算机为核心的信息系统面临如此之多的威胁，反映出信息系统本身存在一些固有的弱点和脆弱性。它的脆弱性主要表现在以下方面。

1、硬件设施的脆弱性

除难以抗拒的自然灾害，如雷击、地震、水灾等外，温度、湿度、尘埃、电磁干扰和人为破坏等均可以影响计算机系统各种设备的正常工作。保证计算机信息系统各种设备的物理安全是计算机信息系统安全的前提。

计算机系统通过电磁辐射使信息被截获而失密的案例已经很多，在理论和技术支持下的验证工作也证实了这种截取距离在几百米，甚至可达千米的复原显示给计算机系统信息的保密工作带来了极大的危害。为了防止系统中的信息在空间中的扩散，通常是在物理上采取一定的防护措施，以减少或干扰扩散出去的空间信号。这对重要的政府部门、军事和金融机构在构建信息中心时都将成为首要设置的条件。

在一块小小的磁盘上或光碟上，可以存储大量的数据信息，而它们很容易放在口袋里带出办公室，数据存储的高密度为入侵者窃取信息带来了便利。

另外，这些存储介质也很容易受到损坏（有意或无意），造成大量信息的丢失。

保存在存储介质上的数据可能会将存储介质永久地磁化，因此存储介质上的信息有时擦除不净或不能完全擦除掉，使得介质上留下可读的痕迹，这些信息一旦被利用就可能产生泄密。

另外，大多数计算机操作系统中，删除文件时仅仅只将文件名删除，并将相应的存储空间释放，而文件的内容还原封不动地保留在存储介质上，利用这一现

象入侵者也可以窃取机密信息。

当获得单个、孤立的信息时，它的价值往往不大，但如果将大量相关的信息聚集在一起，经过筛选和分析，则可显出这些信息的重要性。计算机的特点之一就是能收集大量的信息并对其进行自动、高效的处理，这种聚生性可被入侵者利用来窃取他感兴趣的机密信息。

2、软件的脆弱性

计算机信息系统的软件可分为三类：操作平台软件、应用平台软件和应用业务软件。它们以层次结构组成信息系统的软件体系。操作平台软件处于最底层，支持着上层软件的运行。因此操作平台软件的安全是整个信息系统安全的基础，它的任何风险将直接危及到应用平台软件和应用业务软件的安全。应用平台软件在维护自身安全的同时，必须为应用软件提供必要的安全服务。应用业务软件处于顶层，直接与用户打交道。对系统的许多攻击都是通过应用业务层来实施的，它的风险直接反映了系统的安全风险。

在软件的设计与开发过程中往往存在许多错误、缺陷和遗漏，从而形成系统的安全隐患，而且系统越大、越复杂，这种安全隐患就越多。据有关资料估计，微软开发的操作系统 Windows 的各种版本，平均每 100 行代码大约要出现 0.5～1 个错误或缺陷。然而，这些缺陷和漏洞恰恰是黑客进行攻击的首选目标。

导致黑客频频攻入系统内部的主要原因是，相应系统和应用软件本身的脆弱性和安全措施不完善。另外，信息系统中的“后门”是普遍存在的，它可能是生产厂家或程序员在生产过程中为了自便而设置的，也可能是黑客入侵后在其中设置的。利用后门黑客可以在程序中建立隐蔽通道，植入一些隐蔽的恶意程序，进行非法访问，达到窃取、篡改和破坏信息的目的。

目前市场上尚无任何一个大型操作系统或数据库管理系统可以做到完全正确无误没有缺陷，所以这些系统的厂商都要定期地推出新的版本，其中包括数以千计修改过的语句和代码。这些改动大多数是为了纠正系统中的错误或弥补其缺陷。这些系统的设计者永远无法充满自信地宣布已经找到了系统中所有的漏洞。另一方面，入侵者们多数不会公布他们的发现，因此，当你将重要的敏感信息委托给一个大型操作系统或网络中的一个计算机时，你没有理由不为你的信息的安全担忧，尤其是当这些信息对入侵者有足够的价值时。

虽然任何操作系统和数据库管理系统都有缺陷，但绝大多数系统是可用的，可以基本完成其设计功能。这就如一个墙上有洞的房间，虽能居住，却无法将盗贼拒之门外。

3、网络通信的脆弱性

网络系统中，通信线路很容易遭到物理破坏，也可能被搭线窃听，甚至于插入、删除信息。无线信道的安全性更加脆弱，因此通过未受保护的外部线路可以从外部访问到系统内部的数据。

资源共享是建立计算机网络的基本目标之一，但这也为系统安全的攻击者利用共享的资源进行破坏活动提供了机会，也为攻击者利用资源共享的访问路径对其它非共享资源进行攻击提供了机会。

计算机网络是一个复杂的系统，网络的可扩展性使得网络的边界具有不确定性，这使得网络的管理变得十分困难，构成了对网络安全的严重威胁。

信息传输时，一个节点到另一个节点可能存在多条路径，一份报文在从发送节点到达目标节点之间可能要经过若干个中间结点，这种路径的不确定性和中间结点的不确定性，使得仅有起始节点和目标节点的安全保密性还不足以保证信息的安全。

数据库技术和网络特别是 Internet 网技术的兴起、发展和广泛应用极大地促进了社会信息化的进程，它使得信息可以超越时间和空间的界线达到最大程度的共享。现在，无论你在地球上什么地方，也无论什么时候，只要你轻点一下计算机鼠标，你就可以获得许许多多来自不同地方和部门的信息。人们在享受技术进步为我们工作、生活带来的这种方便和效率的同时，也感受到了它所带来的对系统中信息安全的威胁。当前，多平台、充分集成的分布式信息系统成为最流行的处理模式，而集中分布相结合的处理方式也很受欢迎。21 世纪的信息系统将建立在庞大、集成的网络基础上，而在新的信息系统环境中，由于移动计算的普及，存取点将大大增加，从而信息系统的薄弱环节将分布更广。事实上，现代计算机领域中的任何一个大的技术进步都可能对计算机系统自身的安全构成一种新的威胁，所有这些威胁都需要研究出新的方法和技术来予以消除。

1.1.3 信息安全的目标

现代信息技术的飞速发展为社会带来了巨大的效益，高速计算机和高速网络

的逐步政用化、商用化、军用化和民用化反映当今社会对信息及信息技术的巨大依赖性。信息已成为人类宝贵的资源，它关系到国家的机密和企业的发展，甚至关系国家和企业的生死存亡。正因为信息在人类社会活动、经济活动中起着越来越重要的作用，因此信息的安全日益受到社会越来越广泛和深刻的重视。

所谓信息系统的安全，是指对于信息系统中的硬件、软件及数据进行保护，防止它们因偶然或恶意的原因而遭到破坏、更改或泄露。为此，信息系统在获取、存储、处理、传输和控制信息的过程中，要建立和采取一些技术上和管理上的安全保护措施。

信息系统安全从需要保护的对象考虑，可分为外部安全和内部安全。外部安全是指构成信息系统的计算机硬件、外部设备以及网络通信设备的安全，使其不会丢失或受到毁坏，能为系统提供正常的服务。内部安全是指信息系统中程序、数据和服务的安全，使其不被破坏，更改和泄露。无论是内部安全还是外部安全，信息系统安全的最终目标是要使得信息在系统内的任何地方、任何时间和任何状态下保持其安全性。相对来说，维护外部安全比维护内部安全简单一些，它主要是通过物理保护、加强管理和法律的手段来加以防范。但是对于内部安全来说，仅有上述保护措施是不够的，还必须在技术上采取一系列的措施来防范对系统中信息的攻击。

在这里，信息主要是指存放在信息系统中的程序和数据。信息安全则是指这些程序和数据在被存储、处理、执行和传输过程中的安全。

信息安全意味这什么呢？或者说信息安全要达到的目标是什么呢？虽然说法各有所异，但以下几个方面是较为普遍的认识。

1、机密性（Confidentiality）

信息的机密性是指保证信息不泄露给非授权的个人和实体，即使非授权的个人或实体获得了信息也无法知晓信息内容，因而不能利用。机密性是信息系统是否安全的一个重要标志。军事信息系统尤其注重信息的机密性。随着 Internet 网络的联通，信息可以跨地区、跨国界地共享，信息机密性问题变得更为普遍和突出。通常采用访问控制阻止非授权个人或实体获得机密信息，采用加密技术阻止非授权的个人或实体获知信息的内容。

2、完整性（Integrity）

信息的完整性是指维护信息的真实性、准确性和一致性，信息在未被授权的情形下，保持不被篡改、破坏和丢失的特性。并且能够判别信息是否已被改变。

软件的优点是无论在开发还是在实用的过程中都可以随时更改，以使系统具有新的功能 ，但这种灵活易变性给系统的安全带来了威胁。因此，选择值得信任的系统设计者是一个非常重要的问题，一个不忠实的设计者可以在软件中留下陷阱或圈套，以备将来修改软件对系统进行攻击。计算机病毒通过附加一部分病毒程序代码到系统或应用软件上，从而进行病毒传播，如果对软件有一个较好的完整性保护措施，那目前绝大部分病毒就不能蔓延。我们需要采用软件测试工具来检查软件的完整性，并保证这些软件不会被轻易地修改。

数据的完整性是应用计算机系统进行信息处理的用户，特别是商业部门和金融部门的用户十分关心的一个问题，也是怀有恶意的人进行攻击的主要目标之一。

引起信息完整性破坏的原因来自于多方面，如自然灾害、人为的有意和无意行为、因质量或其他因素导致的设备故障、环境和通信的影响以及不可预知的软件错误等方面，保证信息的完整性需要使用多种方法。通常采用访问控制阻止非授权的篡改行为。通过消息摘要算法检验消息是否被篡改。同时，还要运用故障应急方案和多种预防性技术，诸如归档、备份、镜像、崩溃转储和故障前兆分析等手段。

3、可用性(Availability)

信息的可用性是指信息可被授权实体访问并按需求使用的特性。既保障信息资源随时可提供服务。为此，系统必须防止由于人为或非人为因素造成的系统拒绝服务。诸如系统性能降低、系统崩溃而需人工重新启动、数据永久性丢失、网络被破坏以及系统不能正常运行等，均可造成信息资源不能按照用户的需要供用户使用。可用性是信息资源服务功能和可靠性的度量。

系统在运行过程中应具有抗干扰和保持正常工作的能力，即保持工作的连续性和正确性的能力。这涉及物理环境、网络、系统、数据、应用和用户等多方面因素，是对计算机及网络通信系统总体可靠性的要求。

4、抗抵赖性（Non-repudiation）

抗抵赖性是指能保证用户无法在事后否认曾经对信息进行的生产、发送、接

收等行为。这一特性使得信息在通信各方之间的流动保持其真实一致性。如果不能抗抵赖，就可能出现冒充。这一特性对于商业和金融系统显得特别重要。通常采用数字签名方法来提供抗抵赖服务。

5、可控性（Controlability）

信息的可控性是指授权机构可随时控制信息的流向及用户的行为方式，对信息的传播及内容具有控制能力。为保证可控性，首先需要对用户进行身份验证，其次要控制谁能访问系统中的信息以及以什么方式访问，并且要将用户的活动记录下来。当网络中机器的使用时间、敏感操作和违纪操作等，为系统进行事故原因查询、定位、报警以及对事故的实时处理提供详细可靠的依据和支持。

总之，信息安全的宗旨是，不论信息处于动态还是静态，均应该向合法的服务对象提供准确、及时、可靠的信息服务，而对其它任何人员或组织，包括内部、外部以至于敌方，都要保持最大限度的信息的不可接触性，不可获取性，不可干扰性和不可破坏性。

1.1.4 信息安全研究的内容

信息系统是为用户提供信息服务的，信息系统安全的最终目标是为用户提供安全的信息服务。因此信息安全是整个信息系统安全的核心。而信息系统安全是信息安全的保障，信息安全与信息系统的安全密不可分。

由于信息系统是一个复杂的人机系统，因此对信息安全的防护不能仅依靠单一的技术和方法，它必须综合运用多种技术和方法，多层面地进行防护。

信息安全的研究是涉及数学、物理、通信、计算机、管理、法律等多个领域的一门交叉性学科。

1、物理和环境的安全

物理安全是指要保障计算机及网络物理设备不受物理损坏，或者损坏后能及时修复或替换。这需要针对自然灾害，如雷击、地震、水灾等的破坏，设备使用中的自然损坏及人为的破坏，如火灾、盗窃等，采取相应的物理措施，如备份技术，安全设计技术和安全加固技术，采用多层安全防盗门和隔离墙等。此外还要保证信息系统运行的环境安全，如适当的温度、湿度、防电磁辐射和干扰等。物理安全主要保障信息系统的实体安全，为信息在存储、传输和处理的过程中提供安全的物理环境。这是信息安全的基本保障。

2、管理安全

信息安全的建设是一个复杂的系统工程。它需要对信息所处的计算机环境中的各个环节进行统一的综合考虑、规划和构架，并要时时关注组织内不断发生的变化。任何单个环节上的安全缺陷都会对系统和整体安全构成威胁。据权威机构统计表明，网络与信息安全事件中大约有 70%以上的问题都是由管理方面的原因造成的。

管理安全包括行政管理安全和技术管理安全两个层面。

行政管理是指对维护、管理和使用信息系统的人员在工作责任心、职业道德和安全意识等方面的教育和管理，在技术上的培训。要严格对工作人员的审查，制定工作人员必须遵守的规章制度，减少和消除内部工作人员的失误，防止内部工作人员利用职务进行计算机犯罪，防止外部人员勾结内部工作人员进入系统进行破坏或窃取。特别是对信息系统负有资源管理和安全管理责任的管理人员，因为他们的权力过大，在信息系统中他们的权力甚至大过相应组织的领导的权力。因此，对他们的严格管理对整个系统的安全尤为重要。将组织的安全管理条例或安全管理政策写成书面文件，广泛散发到机构内所有员工手中，并对所有员工进行信息安全政策的培训，以使信息安全的意识植根于组织内所有员工的脑海并落实到实际的工作中。

信息系统是人构造的，信息系统也是给人使用的，因此信息安全的问题归根到底是人的问题，仅有对人的行政管理不可能完全解决信息的安全问题。正如虽然有警察、保安和法律在限制人们的行为，维护社会的安全，但我们每一个家庭、每一个办公室的门仍然要上锁，重要的文件柜还要加锁，并且人们在技术上还在不断研究和生产具有强防盗功能的门。与此类似，对于信息系统来说，虽然有上述行政管理的措施甚至法律的手段对人的行为进行限制和规范，但信息系统本身还需要在技术上采取各种措施来进一步规范用户的行为。这种技术上的管理是将使用信息系统的组织的安全需求或它所制定的安全策略，用技术的手段使之在系统中实现。

从微观上看，我们必须通过技术手段识别系统的合法用户和非法用户，将非法用户拦截在系统之外，必须根据使用信息系统的组织的安全需要，采用技术的手段为不同的合法用户分配不同的权限，并控制每一个用户均不能越权进行操

作，还必须为系统指定相应的资源管理员和安全管理员，并对这些用户特权的使用进行监控，以防他们滥用其拥有的特权。系统中还应该有审计机制，监视和记录用户的行为，为日后安全事件的追查和分析提供依据。

从宏观上看，我们需要对信息系统进行安全风险评估，对安全风险来自于哪些方面，应重点保护哪些资源，花费多大代价，采取什么措施，达到什么样的安全强度等进行科学的分析。对一个系统的安全风险研究得越精确，其安全需求也就越明确。

制定计算机系统的安全评估标准可以为研制、开发安全系统的制造商提供技术支持和依据，可推进安全技术和产品标准化、规范化，为安全系统的建设和管理提供技术指导，为用户对于计算机系统内处理机密信息的可信度提供一种评价尺度。著名的安全标准有可信计算机系统评估准则(TCSEC)、通用准则(CC)和安全管理标准 ISO17799 等。

3、立法安全

计算机与通信技术的快速发展，使人们对信息和信息技术的依赖与日俱增。国家的政治、军事、经济、金融、科学技术、文教卫生以至个人生活等绝大部分信息都在计算机信息系统中存储、处理并进行传输，社会的信息化引起计算机犯罪活动的猖獗，而计算机犯罪又是不同于传统犯罪方法的一种高科技和高智能的犯罪。为了惩治这种犯罪，必须制定与信息安全相关的法律、法规，使犯罪分子慑于法律，不敢轻举妄动。

自 1946 年世界上第一台计算机问世后，直到 1973 年，瑞典才率先制定了世界上第一部国家性的《瑞典国家数据保护法》，并成立了国家数据监察局负责这方面的工作。

随后，美国、法国、英国、德国、加拿大和日本等几十个国家，相继制定了有关数据安全和计算机犯罪的法律和法规。

美国是世界上拥有计算机及网络最多，应用最广泛，普及程度最高的国家，也是计算机犯罪最严重的国家，因而拥有较多的针对计算机犯罪的法律。除 31 个州均已制定了这方面的法律外，在联邦法律中，与计算机犯罪有关的就有：1974 年的《个人秘密法》、1978 年的《防止向国外的不正当支付法》、《资金电子调拨法》、《金融秘密法》、1986 年的《计算机欺诈和滥用法》、《电子通信秘密法》、

1987 年的《计算机安全法》、1989 年的《计算机病毒消除法》等等。在这些法律中规定：利用计算机获取非法利益；非法得到计算机系统的服务；用非法手段侵入计算机系统和网络进行盗窃、破坏、篡改数据和文件，或扰乱系统功能；利用计算机或计算机技术知识和技巧窃取金钱、数据和信息等行为，均为计算机犯罪，根据情节将处以罚款、监禁或两罚并用。现在美国国会制定的对付计算机犯罪的联邦法律有 3 种，一是《计算机病毒防治法》、二是《计算机保护法》、三是《计算机网络保护法》。

我国政府也十分重视计算机信息系统的安全，1992 年，我国颁布了《计算机软件保护条例》，又颁布了《中华人民共和国计算机信息系统安全保护条例》，在这之后，又陆续颁布了《中华人民共和国计算机信息网络国际联网管理暂行规定》、《中国公众多媒体通信管理方法》、《国际联网安全保护管理方法》，并在《刑法》中增设了关于计算机犯罪的条款。

4、技术安全

技术安全是指计算机系统本身在运行过程中，采用具有一定安全性质的硬件、软件来实现对信息的安全保护，以使得在整个系统中，在一定程度上甚至完全可以保证系统在无意或恶意的软件或硬件攻击下仍能使得系统内的信息不被破坏、增加、丢失和泄露。

采用技术的手段保护信息安全，通常从以下几个方面着手。

（1）身份鉴别

身份鉴别相当于系统的门卫，它监控进入系统的每一个用户，防止非法用户进入系统。身份鉴别包含两个过程，一是识别，一是验证。所谓识别是指系统要知道访问者是谁？为此每个合法用户必须拥有唯一的识别符（唯一 ID），为了保证识别的有效性，任意两个不同的用户都不能具有相同的识别符。所谓验证是指系统通过访问者所提供的验证信息，对其所声称的身份进行验证，以防假冒。识别信息（用户名或帐号）一般是非秘密的，而验证信息（如口令或通行字）必须是秘密的，只有用户和系统知道。

身份鉴别机制可以用来对抗某一实体谎称是另一实体的假冒攻击。在网络通信中，鉴别机制可用于通信双方相互进行身份验证。

（2）访问控制

经过身份鉴别进入系统的合法用户，对系统资源的访问权还必须受到限制，系统中访问控制机制用以决定每一个用户对哪些资源有访问权和有什么样的访问权。用户对信息资源访问权一般分为读、写、修改、删除、执行、控制等。访问控制的目的是防止合法用户对系统资源的非法访问。传统的访问控制方法分为自主访问控制和强制访问控制。基于角色的访问控制是近几年出现的一种新的控制方法。

（3）信息流控制

仅有访问控制还不足以限制用户的访问权，因为有权访问的用户可以把访问到的数据传递给无权访问的用户。也就是说，访问控制的着眼点是控制用户对系统中数据对象的访问权限，而没有涉及到用户对他们所拥有的信息的管理。因此在系统中还必须使信息沿着能保证信息安全的方向流动，这就需要对信息流进行控制。

访问控制和信息流控制是对系统中数据的机密性、完整性、可用性和可控性实施保护的重要技术。是继身份鉴别之后系统防范安全攻击的第二道屏障。

（4）数据加密

在本书中，数据常用来泛指信息系统中存储和传输着的各种信息（如程序、文件、和各种数据）。数据加密的任务是通过编码技术来改变所要保护的信息的形式，使得除了指定的接收者外，其它人无法读懂，利用编码技术所要隐藏的原始信息称为明文，经过编码技术伪装以后的信息称为密文。

从理论上讲，系统如果有了严格的身份鉴别、访问控制和信息流控制，那么敌手便无法接触到他无权访问的信息。但实际上并非如此，如前所述，信息系统是一个十分复杂的系统，加上信息系统本身所固有的一些脆弱性，使得它随时都可能面临敌手突破或绕过系统已有的安全防线而遭到攻击。因此，在上述安全控制的基础上，对系统中的敏感信息进行加密是信息系统又一道重要的安全屏障。它使得机密信息即使被窃取了，敌手也难以识别和篡改。在网络通信中，加密是阻止传输中的数据被窃取的最好方法，同时，通过数据的加/脱密也可及时发现对数据的篡改。加密还是防止假冒和抵赖的主要方法。

（5）推理控制

推理控制是防止用户根据访问到的数据，用逻辑推理的方法推导出他无权获

得的数据。这一安全技术对于只提供对事物组有关的统计量的访问，而限制对任何特定的个别事物信息访问的统计数据库尤为重要。例如，人口普查部门负责收集有关所有居民的信息，并以不危及个人秘密的方式报道这些信息。问题是统计数据中包含着原有信息的痕迹，通过收集不同的统计量，聪明的用户可能推出有关一些个别事物的秘密信息。推理控制的目标就是要阻止这种推导的成功。

（6）隐通道的分析和限制

隐通道是指两进程以违反系统安全策略的方式传输信息的隐蔽通道，它可以造成严重的信息泄漏。在操作系统和数据库系统中都可能存在隐通道，分析并搜索出系统中存在的隐通道，对其加以清除或限制是计算机系统安全设计中的一个难点。

（7）审计

审计是模拟社会监查机构在计算机系统中用来监视、记录和控制用户活动的一种机制。它对涉及系统安全的操作作完整的记录，使违反系统安全的访问和访问企图留下线索，以便事后分析和追查。

如果在审计中设置报警功能，那么每当有违反系统安全的事件发生或者有涉及系统安全的重要操作进行时，可以向系统安全员发送相应的报警信息，以便及时地对事件进行控制。

审计可以说是信息系统中维护信息安全的最后一道防线。

（8）安全管理

计算机信息系统是一个结构复杂、动态变化的人机系统，仅仅依靠系统提供的安全服务（它可能是多种安全技术的结合）的支持是不够的。要想有效地保护信息系统的安全，最大限度地减小面临的安全风险，还必须有对信息系统严密的安全管理。现代计算机信息系统的安全管理既是一个复杂的技术问题，也是一项要求严格的管理规范。

按照传统的管理模式，仅仅设置系统管理员是不够的。根据安全的需要，系统应设置三类管理员：

① 系统管理员：对系统的资源进行管理和分配，具有创建用户、删除用户等权力；

② 系统安全员：对系统有关安全的事务进行管理，如对主、客体的安全级

进行分配和维护，或对用户的角色进行指派和维护；

③ 系统审计员：对系统的审计事务进行管理，并负责对系统管理员和系统安全员进行审计。

1.2 访问控制

在信息系统中，访问控制是在身份鉴别之后保护系统资源安全的第二道屏障，是信息系统安全的重要功能构件。它的任务是在为用户对系统资源提供最大限度共享的基础上，对用户的访问权限进行管理，防止对信息越权篡改和滥用。它对经过身份鉴别认证后的合法用户提供所需要的且经过授权的服务，拒绝用户越权的服务请求，使信息系统的用户在系统的管理策略下有序地工作。因此，用户在通过了身份鉴别之后，还要通过访问控制才能在信息系统内执行特定的操作。

据有关资料统计，80%以上的信息安全事件为内部人员和内外勾结所致，而且呈上升趋势。因此通过技术手段对用户实施访问控制对信息安全有着重要的作用。

1.2.1 访问控制原理

安全是一个相对的概念，不同的应用有着不同的安全需求，适合于某个系统的安全策略不一定适合于其他的系统。例如，极严格的访问控制策略对于军事系统可能是至关重要的，但对于一些提供公众服务的系统也许就不合适了。因此对访问控制技术的选择取决于被保护环境的独有特性。

对信息系统实施访问控制，首先要对该信息系统的应用背景作安全需求调查，如组织机构，人员结构，用户的数量、类型，特别是系统要为每个用户或各类用户提供什么样的服务等。在对安全需求进行充分考虑分析的基础上，制定出适合其安全需求的安全策略或安全规则。这些规则是对用户行为的规范和约束，只有规则允许的访问才是合法的访问，违反规则的访问请求将被拒绝，违反规则的访问行为将被视为非法。将这些安全规则用计算机能识别的方式描述出来，作为系统进行访问控制的依据。

访问控制控制进入系统后的用户能做什么，也控制代表用户的进程能做什么。

用户对资源的访问可以是信息资源、通信资源或处理资源。访问方式可以是获取信息(称为读)、修改信息(称为写)、添加信息(称为添加)、删除信息(称为删除)或者执行某个程序的功能。访问的方式也可以在应用层进行定义，如借款、

贷款，生成公文、修改公文或签发公文。在应用层上定义的一种访问方式往往包含了一组特定的读、写或添加操作。

访问控制应遵循的一条重要的安全原则是“最小特权”原则，也称“知所必需”，即用户所知道的或他所做的，一定是由他的工作职务或他在该信息系统中的身份、地位决定所必需要的。系统分配给每一个用户的权限应该是该用户完成其职能的最小权限集合，系统不应给予用户超过执行任务所需权限之外的任何权限。

在访问控制策略确定之后，系统必须有一套机制来执行访问控制策略。它主要有以下几项任务。

① 根据系统的安全策略给每一个用户或各类用户进行授权，根据系统的需要，还要具有动态授权或安全地修改权限的功能。

② 在用户提出访问请求时，识别和确认用户。

③ 根据系统对用户的授权和访问控制规则，对用户的请求作出响应：执行或拒绝。

总的来说，访问控制包括两个部分，授权(Authorization)和控制(Control)。而授权和控制均是依据系统事先制定的安全策略来进行的。

1.2.2 访问控制的研究概况

20 世纪七十年代初期，人们开始对计算机系统安全研究的同时，也开始了对访问控制的研究。1971 年，Lampson 提出了访问矩阵的概念，并成功地应用于保护操作系统中的资源。后来 Denning 等人对该概念进行了改进，最后 Harrison 等人将该模型概念完善为一种框架体系结构为系统提供保护。访问矩阵控制模型体现了自主访问控制(Discretionary Access Control)的安全策略，其访问权的管理相对困难，因为仅给每个用户分配对系统文件的访问属性不能有效地控制系统的信息流向，为此，人们开始研究安全性更高的安全策略模型。1973 年，Bell D E 和 La Padula L J 提出了为系统中每个主体和客体分配相应的安全属性来控制主体对客体访问的 BLP 模型，并不断对其进行改进，至 1976 年完成了它的第四版。1976 年 Dennning D E 等人提出了控制信息流向的格模型。该模型所反映的安全需求从本质上来说与 BLP 模型是一致的。但该模型对 BLP 模型进行了扩充，它不仅禁止用户直接访问超过他安全级的客体，而且还禁止他伙同有权访问这些客

体的用户，以某种巧妙的方式间接访问这些信息。

以上工作形成了早期的两种访问控制类型：自主访问控制和强制访问控制。

1983 年美国国防部制定了《可信计算机系统评估准则》(Trusted Computer System Evaluation Criterria，简称 TCSEC)，将计算机系统的安全可信度从低到高分为 D，C，B，A 四类共七个级别：D，C1，C2，B1，B2，B3 和 A1 级。该标准中定义了以上两种访问控制技术，其中自主访问控制被定义作为商用、民用、政府系统和单级军事系统中的安全访问控制标准，强制访问控制被定义作为多级军事系统的安全访问控制标准。这些标准一直被人们认为是安全有效的。这两种访问控制技术也在很多系统中被采用。

1987 年 Clark 和 Wilson 提出了 CW 模型，其中包含合式事务(Well-formed Transaction)和职责分散原则(Seperation of Duty，即 SOD)，它体现了一种应用于商业领域的强制访问控制策略。1989 年 Brewer 等人提出了中国墙(Chinese Wall)策略，它体现了一种应用于金融领域的强制访问策略。

自主访问控制和强制访问控制都存在管理困难的缺点。随着计算机应用系统的规模不断扩大，安全需求不断变化和多样化，需要一种灵活的能适应多种安全需求的访问控制技术。20 世纪 90 年代，基于角色的访问控制(Role Based Access Control，即 RBAC)进入了人们的研究视野。RBAC 的概念最初产生于 20 世纪 70 年代的多用户、多应用联机系统中，90 年代初美国国家标准技术研究所(National Institute of Standards and Technology)的安全专家们提出了基于角色的访问控制技术。1996 年，George Mason 大学教授 Sandhu 等人在此基础上提出了 RBAC96 模型族，1997 年进一步扩展 RBAC96 模型，提出了利用管理员角色来管理角色的思想，提出了 ARBAC97 模型。这些工作获得了广泛的认可。

随着计算机系统网络化，系统面向的应用越来越复杂，安全的讨论逐渐扩展到分布式系统环境，已有的单域环境下集中式的访问控制技术已不能适应分布式系统中出现的安全问题，多个管理域环境下的访问控制成为了人们研究的热点。

习题一

1.1 计算机信息系统可能遭到哪些方式的攻击？你是否遇到过某种攻击？
1.2 计算机信息系统的脆弱性表现在哪些方面？
1.3 信息系统安全的含义是什么？信息安全的含义是什么？它们之间有何关系？
1.4 对计算机信息系统及信息安全的保护应从哪些方面着手？
1.5 对信息安全进行保护的技术手段目前主要有哪些？
1.6 访问控制在信息系统安全中的地位和作用是什么？
1.7 访问控制中，授权的依据是什么？控制的依据又是什么？
1.8 有哪些常用的访问控制技术？

第 2 章 身份认证

随着计算机和互联网技术的发展和普及，信息处理自动化程度越来越高。人们日常生活已经离不开现存的电子设备和软件环境了。电子邮件系统、电子银行系统、证券交易所股票交易软件、数字图书馆以及报刊杂志电子栏目等已是实现人们相互通讯、方便现金存取、投资理财、查阅资料文献的常用工具。它们提供的服务不能发生差错，必须安全准确。比如银行系统 ATM（自动取款机），绝不可以让假冒他人身份者取到现金。因此，当用户从 ATM 提取现金时，ATM 先要确定用户身份。用户持银行卡和密码是证明自己身份的常用方法。

身份认证是许多应用系统的第一道防线，也是实施访问控制的依据。身份认证的目的在于识别用户的合法性，获得合法用户身份的准确信息并能够阻止非法用户进入系统，从而能够为正确实施访问控制策略提供保证。如 ATM 必须对银行卡和密码声称的身份进行验证确认，银行系统中确实存在这么一个帐户，且身份验证信息与用户声称（银行卡和密码）的相符，如果验证通过，银行访问控制系统将根据用户提供的口令类别提供相应的服务，比如：用户输入查询口令则无法从 ATM 中取款，只能完成查询功能。

本章将对身份认证的作用、原理及实现方法作重点介绍。

2.1 什么是身份

身份是实体的一种计算机表达。通常，一个用户就是约束为一个独立实体的身份。特定系统可能会附加不同的约束条件。各种系统使用多种不同方式来表达用户身份（例如，可以表达为一个整数或一个字符串），同一个系统在不同场合也会使用不同的身份表达方式。同一个参与者可以有多个不同的身份。一张证书可通过角色标识参与者，也可以通过参与者的工作、地址进行标识；Internet 上的主机有多个地址，每一个地址都是它的身份。通常，每一个身份都具有一项特定的功能。

通常一项事务的参与者是一个唯一确定的实体，一个身份指定一个参与者。通过认证，可以将一个参与者绑定为计算机内部的一个身份表达。不同的系统有不同的身份表达方式，但所有的访问策略和资源分配操作都要假定这种绑定是正

确的。使用身份有若干目的，两个主要目的是：访问控制与可追查性。

每一种访问控制机制都要基于某种类型的身份。身份与参与者绑定的力度与准确性决定了系统使用身份的效果。访问控制要求身份能用于访问控制机制以确定是否允许特定的操作（或操作类型）。大部分系统根据进程执行者的身份来设置访问权限。

可追查性要求身份能够跟踪所有操作的参与者，以及跟踪其身份的改变，使得参与者的任何操作都能被明确地标识出来。可追查性需要依赖日志与审计两种技术。可追查性要求参与者有明确的身份表达，然而在许多系统中，这是不可能实现的。取而代之的是将被记录的身份映射为用户帐号、群组或角色。

2.2 认证基础

主体代表某些外部实体进行操作，而实体的身份则控制了与其关联的主体的行为。因此，主体必须与外部实体的身份绑定，认证就是将某个身份与某个主体进行绑定。

外部实体必须提供信息使得系统能够证明它的身份，这些信息来自以下某个（或多个）方面：

实体知道什么（如口令或秘密信息等）。

实体拥有什么（如证章、卡片等）。

实体的生物特征（如指纹、视网膜特征等）。

实体的行为特性（如用户的下意识动作）。

认证过程包括从某个实体获取认证信息、分析数据、确定信息是否与该实体相关等环节。这意味着计算机必须存储实体的某些信息，同时必须具有管理这些信息的机制。认证系统的要求可归纳为以下 5 个方面：

认证信息集合 A：实体用于证明其身份的特定信息的集合。

补充信息集合 C：系统存储并用于验证认证信息的信息集合。

补充函数集合 F：根据认证信息生成补充信息的函数集合。即，

对 $f \in F, f : A \to C$ 。

1. 认证函数集合 L：用于验证身份的函数集合。即，

   对 $l \in L, l : A \times C \to \{true, false\}$ 。

2. 选择函数集合 S：使得一个实体可以创建或修改认证信息和补充信息。

2.3 根据实体知道什么进行身份认证

2.3.1 口令

口令是认证机制的一种实例，它基于实体所知道的信息：用户提供一个口令，计算机验证该口令。如果输入的口令是和该用户相对应的口令，则用户的身份得到认证，否则就拒绝该口令，同时认证失败。

最简单的口令是字符序列，在这种情况下，口令空间就是能够成为口令的所有字符序列的集合。例如，要求用户选择一个由 10 个数字的序列做口令，那么口令空间 A 就有 10^{10} 个元素（从“0000000000”到“9999999999”）。

依据补充函数的性质，补充信息集合可以包含比 A 更多或更少的元素。最初，大部分系统把口令存储在一个受保护的文件里，但是这种文件的内容可能会因偶然事件而泄露。如果使用数据加密的方法将口令以密文的形式存放在系统中，则加密密钥的保存又成了一个严重的安全问题，一旦这个密钥泄露，则所有的口令都有可能泄露。所以口令在系统中的保存不单应该是密文形式，而且要想根据密文还原出明文口令在计算上也是不可能的。解决的办法是使用一个单向散列函数将口令加密，即对口令进行加密的算法是单向的，口令经加密后，脱密是不可能的。

认证系统的目标是要保证正确地识别实体。如果一个实体可以猜测出另一个实体的口令，那么猜测者就可以冒充他人。认证模型为这类问题的分析提供了一种系统方法，其目标是找到一个 $a \in A$，使得对于 $f \in F, f(a) = c \in C$，且 c 对应于一个特定实体（或者任意一个实体）。由于通过计算 $f(a)$或通过认证 $l(a)$，就能够确定 a 是否与某个实体关联，因此这里同时使用两种方法来保护口令。

1. 隐藏足够多的信息使得 a, c 或 f 中的一个不能被发现。

   例：许多系统中包含补充信息的文件只对管理员用户可读，这样的方案令实际应用中的补充信息集 c 不可读。因此，攻击者得不到足够的信息来确定 $f(a)$是否与某个用户关联。类似地，某些系统令补充函数集为不可读，则不可能计算 $f(a)$的值。

2. 禁止访问认证函数 L。

例：某些站点不允许管理员用户从网络登录。认证函数存在，但总是访问失败。因此，攻击者不可能通过网络访问认证函数来对管理员用户的认证进行测试。

攻击基于口令的系统的最简单方法是猜测口令。目前猜测口令用的最多的是字典攻击。字典攻击通过重复试验的方式来猜测口令，其名字来源于用于猜测口令的单词表（即“字典”）。字典可以是随机排列的字符串集合，或者（更普遍的）是以选择可能性大小的降序排列的字符串集合。

字典攻击类型一：如果补充信息和补充函数是可以获得的，则字典攻击对每一个猜测 g 和每个 $f \in F$ 计算 $f(g)$。如果 $f(g)$对应于实体 E 的补充信息，则 g 在 f 下可以认证实体 E。

字典攻击类型二：如果补充信息或者补充函数不可用，那么就可能要用到认证函数 $l \in L$。若一次猜测 g 在 l 中返回真，则 g 就是正确的口令。

口令猜测需要知道补充函数集和补充信息集，或者能够访问认证函数。在这两种方法中，防御者的目标是要最大限度地增加猜测口令的时间。重用的口令很容易遭到类型一的字典攻击。随机口令的目标就是使得口令猜测的时间最大化。

如果实际的补充信息或补充函数不能公开得到，那么猜测口令的唯一方法就是使用系统提供的、授权用户用于登录系统的认证函数。这种攻击是不能防止的，因为认证函数必须公开可用，使得合法用户可以访问系统。

要对抗这种攻击，就必须要求对攻击者而言，认证函数的使用是非常困难的，或者使得认证函数以非常规的方式进行交互。通常有四种实现技术：

第一种技术可以统称为“后退”技术。最常见的是指数后退技术，当一个用户尝试认证并失败后就开始。假设 x 是系统管理员选择的参数。如果用户登录首次失败，在提示输入用户名和口令前系统等待 $x^0=1$ 秒。如果用户登录再次失败，系统要等待 $x^1=x$ 秒后才提示输入用户名和口令。在 n 次登录失败后，系统会等待 x^{n-1} 秒才允许重新登录。其他一些后退技术使用算术级数而不是几何级数（立即重新提示，然后等待 x 秒，然后等待 $2x$ 秒，依次类推）。

第二种技术涉及到断开连接。经过许多次失败的认证尝试后，连接断开，用户必须重新建立连接。当建立一个连接需要一定量的时间时，比如重拨一个电话号码，这种技术最为有效。当连接非常快速时，比如通过网络的连接，这种技术

的效率就相对较低。

第三种技术使用了禁用机制。如果一个账户连续 n 次登录失败，这个账户就被禁用，直到安全管理者重新启用它。这可以防止一个攻击者多次尝试口令，同样也可以向安全人员发出攻击警告，使得他们可以采取合适的措施来对抗这种威胁。但应该考虑清楚的是，是否需要禁用一个账户，禁用哪个帐户。

第四种技术称为监禁。让非认证用户访问系统的有限部分，并欺骗攻击者，使得他相信自己拥有了系统的合法访问权限，然后记录下攻击者的行为。这种技术可用于确定攻击者想做什么，或者只是浪费攻击者的时间。监禁技术的另一种形式是在一个运行系统中植入一些假的数据，当攻击者入侵后，他就会获得这些数据。当攻击者浏览或下载这些假数据时，系统就可以对其进行跟踪。这种技术也称为蜜罐，经常用于入侵检测系统。

猜测口令需要能够访问补充信息、补充函数或者获得认证函数。当口令被猜测出来后，如果以上三种信息都不变，那么攻击者就可以使用猜测出来的口令访问系统；否则，即使口令被猜测出来，口令已不再有效了。

为了安全，口令在每隔一段时间后，或者经过一些事件后，必须修改。假设猜测到一个口令的期望时间是 180 天，如果在不到 180 天的时间就改变口令，理论上能减少攻击者能够猜测出口令的概率。实际上，时效性本身并不能确保安全，因为猜测一个口令的估计时间是个平均值，它均衡了那些容易猜测的口令和那些不容易猜测的口令。如果用户选择的是很容易就可猜测出来的口令，对期望猜测时间的估计应该选择最小值，而不是平均值。因此，口令的时效性只有在和其他机制一起使用时才能发挥更好的作用。

口令验证是根据实体知道什么来进行认证的一个例子，是目前应用最广泛的身份认证方法。虽然其安全性比其他几种方法差，但简单易行，如果使用恰当，可以提供一定程度的安全保证。口令一般是由字母、数字、特殊字符、控制字符等组成的具有一定长度的字符串，其选择原则为：

1. 用户容易记忆；
2. 难于被他人猜中或发现；
3. 抗分析能力强；
4. 限制使用期限，可经常更换。

防止口令泄露是这一方法中的关键问题。

目前主要有两种口令生成方法，一是由用户自己选择口令，另外一种是由系统自动生成随机的口令。前者的优点是用户很容易记住口令，一般不会忘记，因为用户所选择的口令往往与用户的日常经验有关，如生日、街道名、配偶名、汽车牌号、房间号码、电话号码等，正因为这样，口令很容易被猜出，泄露的机会较大。另一种较好的方法是口令生成器，随机地为用户生成口令。这种方法带来的问题是用户记忆起来非常困难，即使这个字符串不长，要让一个人记住一个有字母、数字等随机组成的口令也不是一件容易的事。

2.3.2 挑战-回答

口令重用是口令使用的一个基本问题。如果攻击者获得一个口令并使用该口令登录系统，系统没法区分攻击者与合法用户，只好允许他访问。解决这一问题可用挑战-回答的办法来进行认证，并改变每次传输的口令。那么，如果攻击者重复使用以前使用过的口令，它就会遭到系统拒绝。挑战-回答技术的具体实现过程如下：

假设用户 U 想向系统 S 证明自己。设 U 和 S 有个协商好的秘密函数 f。挑战-回答认证系统就是这样一个系统：S 发送一个随机消息 m（挑战）给 U，用户 U 回应以 m 的变形 $r=f(m)$（回答）。S 通过独立计算 r 来验证 r。其中秘密函数 f 也称为为通行证算法。

考虑如下的一个例子：用户必须根据系统的挑战和通行证算法（偶数位置上的字母组成的字符串）来产生口令。当系统给出的挑战是“abcdefg”，则正确的口令应该是“bdf”；如果系统给出的挑战是“zyxwvut”，则正确的口令应该是“zxvt”。可以看出，假如通行证算法对每个用户都是一样的，则身份认证就会失去意义。公钥密码算法在大密钥空间情况下可以保证不同的用户所拥有的私钥是不同的，假如通行证算法 f 是用户使用自己的私钥对挑战进行加密，就可以保证不同的用户产生的回答是不同的。

一次性口令即一个口令只对一次使用有效，这是口令时效性的一种极端形式。在某些场合，挑战-回答系统使用一次性口令。假设把回答作为口令，由于连续认证的挑战是不同的，因此回答也是不同的，因此每个回答（口令）在被使用后就变为无效。

一次性口令方案的问题在于随机口令的产生和用户与系统之间的同步。第一个问题可以使用单向散列函数或者加密函数（如 DES 或 AES）来解决；后一个问题可以通过让系统通知用户它所需要的口令来解决。

使用硬件来支持一次性口令相对简单，因为口令不需要打印在纸上或者输出到某些中间媒体。硬件支持的挑战-回答程序有两种形式：通用的计算机和专门的硬件。两者都可以实现相同的功能。

第一种类型的硬件提供一种对消息进行单向散列和加密的机制。在使用这种设备时，系统首先发送一个挑战。用户把这个挑战输入到设备中，设备返回一个相应的回答。一些设备要求用户输入其个人身份证号码或者口令，并把这作为密钥，或结合挑战一起用来产生回答。

第二种类型的硬件的原理是基于时间的。用户拥有一个设备，每隔一定时间，比如 60 秒，它就显示一个不同的数字，这些数字的变化范围是 0 到 10^n-1。有一个类似的设备与系统连接，它知道每一个注册用户所拥有的设备为用户显示的数字是什么。为了认证，用户提交其用户名和口令，并输入设备显示的数字。系统验证用户口令是否正确，并验证用户提交的数字是否是当时系统所期望的数字。这种类型的挑战-回答系统目前被广泛的用于网上银行业务，如银行为每个办理网银业务的用户发放的动态数字口令生成器。

采用了“加密的密钥”技术的挑战-回答系统对第一类型的字典攻击是免疫的。它确保挑战不以明文形式发送，由于挑战是随机的，对攻击者来说是不可知的，即使攻击者能够解密它，也不能对它进行验证。考虑下面的一个例子：

1. Alice 用共享的口令 s 加密一个随机选择的公钥密码系统的公开密钥 p，接着 Alice 把这个密钥和她的姓名发送给 Bob。
2. Bob 用共享的口令解密得到这个公开密钥 p，然后产生一个随机密钥 k，并把它用 p 加密，把得到的结果用共享口令 s 加密，然后发送给 Alice。
3. Alice 解密消息得到 k，现在 Bob 和 Alice 就有了一个共享的秘密密钥。此时，挑战-回答阶段开始了。

Alice 产生一个随机的挑战 R_A，用 k 对 R_A 进行加密，并把 $E_k(R_A)$发送给 Bob：

4. Bob 用 k 解密 R_A。然后他产生一个随机的挑战 R_B，并用 k 加密这两个挑战得到 $E_k(R_AR_B)$，然后把它发送给 Alice。

5. Alice 解密消息，验证 R_A，并得到 R_B。她用 k 对 R_B 加密，然后把回答 $E_k(R_B)$发送给 Bob。
6. Bob 解密回答并验证 R_B。

此时，Alice 和 Bob 都知道他们在共享随机密钥 k。为了理解为什么这个系统对第一类型的字典攻击免疫，请仔细思考每一轮的交互。因为在每一次交换中，数据都是随机产生的，并对于攻击者来说是不可见的，攻击者不可能知道何时对消息进行了正确的解密。

2.4 根据实体拥有什么进行身份认证

利用合法用户所持有的某种东西来验证身份，对大多数人来说并不陌生，比如通行证或证章等。磁卡是目前最广泛使用的一种，它是一种嵌有磁条的塑料卡，这种卡已经越来越多地用于身份识别，如 ATM、信用卡、磁卡锁等。

磁卡中最主要的部分是磁条。在磁条中，不仅存放着数据，而且也存放着用户的身份信息。一般来讲，磁卡与个人识别号 PIN（Personal Identifying Number）一起使用。在脱机系统中，PIN 必须以加密的形式存储在磁条中，识别设备首先读取卡中的身份信息，然后将其中的 PIN 脱密，并要求用户输入 PIN，识别设备将这两个 PIN 进行比较以判断该卡的持有者是否合法。在联机系统中，PIN 可以不存储在卡上，而存储在主机系统中。进行认证时，系统把用户输入的 PIN 与主机系统中的 PIN 进行比较，据此来判断该卡的持有人是否具有相应的身份。

正如前面对口令的讨论一样，PIN 的选择应该是不容易被其他人猜测出来的，而且用户应该记住 PIN，而不是将其记在纸上或其他媒体上。但是，如果用户不只拥有一张卡，各张卡的 PIN 又都不相同，要想记住许多的 PIN 则是一件非常不容易的事情。

较新的方法是一种称作“智能卡”（Smart Card）的磁卡。这种卡与普通的磁卡的区别在于这种磁卡带有智能化的微处理器和存储器，它将微处理器芯片嵌在卡上，存储信息量远大于磁条，且具有处理功能。卡上的处理器有处理程序和小容量的 EPROM，有的甚至有液晶显示和对话功能。智能卡相对于普通的磁卡其安全性有较大的提高，因为攻击者难以改变或读取卡中的数据。

2.5 根据实体的生物特征进行身份认证

使用物理特征作为身份证明与人类历史一样悠久。通过外表、声音来识别一个人，通过化妆来冒充一个人，这在古代就已被广泛应用。在计算机系统中使用生物特征来辨识一个人，可以理想地消除认证中的错误。生物特征识别技术作为一种身份认证的手段，具有独特的优势，近年来已逐渐成为国际上的研究热点。因为生物特征不会像口令那样容易被忘记和破解，也不会像持有物那样容易被窃取或转移，因此人们认为生物特征识别将是一种更加可靠、方便、快捷的大众化身份认证手段。

人的任何生物特征只要满足下面的条件，原则上就可以作为生物特征用于身份认证：

1. 普遍性，每个人都具有；
2. 唯一性，任何两个人都不一样；
3. 稳定性，这种特征至少在一段时间内是不变的；
4. 可采集性，可以定量测量。

然而，满足上述条件的生物特征对一个实际的系统却未必可行，因为实际的系统还必须考虑：

1. 性能，即识别的准确性、速度、鲁棒性以及为达到所要求的准确性和速度所需要的资源
2. 可接受性，人们对于特定的生物特征识别在日常生活中的接受程度；
3. 可欺骗性，用欺诈的方法骗过系统的难易程度。

因此，一个实际的生物特征识别系统应做到：

1. 在合理的资源需求下实现可接受的识别准确性和速度；
2. 对人没有伤害而且可为人们所接受；
3. 对各种欺诈方法有足够的鲁棒性。

目前人们研究和使用的生物特征识别技术主要有：人脸识别、眼睛识别、手形识别、指纹识别、掌纹识别、声音识别。

典型的生物特征识别系统逻辑上包括两个模块：注册模块和识别模块。在注册模块中首先登记用户名，通过生物特征识别传感器得到用户的生物特征信息，然后从获取的数据中提取出用户的特征模式，创建用户模板，存储在数据库中。

在识别模块中同注册过程一样获取用户的生物特征信息，提取出特征模式，然后与事先注册在数据库中的模板相匹配，检验用户的身份。

1. 人脸识别

人脸识别是一个活跃的研究领域。虽然人脸识别的准确性要低于虹膜、指纹的识别，但由于它的无侵害性和对用户最自然、最直观的方式，使人脸识别成为最容易被接受的生物特征识别方式。

人脸识别主要有两方面工作：在输入的图像中定位人脸；抽取人脸特征进行匹配识别。目前的人脸识别系统，图像的背景通常是可控或近似可控的，因此人脸定位相对而言容易解决。而人脸识别由于表情、位置、方向以及光照的变化都会产生较大的同类差异，使得人脸的特征抽取十分困难。现在主要的人脸识别方法有：

1. 基于脸部几何特征的方法；
2. 基于特征脸的方法；
3. 神经网络的方法；
4. 局部特征分析的方法；
5. 弹性匹配的方法。

基于几何特征的识别是通过提取眼睛、眉毛、鼻子、嘴等重要器官的几何形状作为分类特征。特征脸是根据一组训练图像，利用主元分析的方法，构造主元子空间，这种方法是一种最小距离分类器，当光照和表情变化较小时性能很好，但当其变化较大时性能会显著降低。神经网络方法是将图像空间投影到隐层子空间，由于投影变换具有非正交、非线性的特性，而且可根据不同的需求构造不同的网络，因此识别效果较好。局部特征分析方法是考虑到人脸显著的特征信息并不是均匀分布于整个脸部图像的，可能少量的局部区域却传达了大部分的特征信息，而且这些局部特征在投影前后的关系保持不变。弹性匹配方法是将人脸建模为二维或三维网格表面，应用塑性图形或可变形曲面匹配技术进行匹配。

2. 指纹识别

指纹识别是最古老的生物特征识别技术，在很多领域中都得到了成功的运用。指纹指的是指尖表面的纹路，其中突起的纹线称为脊，脊之间的部分称为谷。指纹的纹路并不是连续、平滑流畅的，而是经常出现中断、分叉或转折，这些断

点、分叉点和转折点，称为细节，就是这些细节提供了指纹唯一性的识别信息。指纹的识别主要包括三部分：特征抽取，指纹分类，匹配决策。

1. 特征提取：从输入的指纹图像中提取出细节，包括，方向场估计、脊线抽取及细化、细节抽取。
2. 指纹分类：在身份认证中，为了提高识别速度，通常先将指纹图像分类。分类算法可采用：利用奇异点等标志信息，利用脊的方向和结构信息，应用句法模式识别方法。
3. 匹配决策：决定两个指纹是否来自同一手指。匹配方法有：基于串的匹配，基于 Hough 变换的匹配，基于 2D 动态规整的匹配。

3. 眼睛识别

通过眼部特征来认证，使用的是眼睛的虹膜和视网膜。

虹膜是一个位于瞳孔和巩膜之间的环状区域。与其它的生物特征相比，虹膜识别具有：

1. 高独特性，虹膜的纹理结构是随机的，其形态依赖于胚胎期的发育；
2. 高稳定性，虹膜可以保持几十年不变，而且不受除光线之外的周围环境的影响；
3. 防伪性好，虹膜本身具有规律性的震颤以及随光强变化而缩放的特性，可以识别出图片等伪造的虹膜；
4. 易使用性，识别系统不与人体相接触；分析方便，虹膜固有的环状特性，提供了一个天然的极坐标系。

虹膜识别算法包括：虹膜定位，虹膜对准，模式表达，匹配决策。

1. 虹膜定位：虹膜从整幅图像中分割出来。为此必须准确定位虹膜的内外边界，检测并排除侵入的眼睑。典型的算法是利用虹膜内外边界近似环形的特性，应用图像灰度对位置的一阶导数来搜索虹膜的内外边界。
2. 虹膜对准：确定两幅图像之间特征结构的对应关系。将原始坐标映射到一个极坐标系上，使虹膜组织的同一部位映射到这个坐标系的同一点；应用图像配准技术来补偿尺度和旋转的变化。
3. 模式表达：为了捕获虹膜所具有的独特的空间特征，可以利用多尺度分析的优势。

4. 匹配决策：用两幅图像虹膜码的汉明距离来表示匹配度，这种匹配算法的计算量极小，可用于在大型数据库中识别；计算两幅图像模式表达的相关性，其算法较复杂，仅应用于认证。

视网膜扫描依赖于眼睛背面血管的模式唯一性，它要求有穿透性很强的激光束照在视网膜上。这种方法只用于需要最高级别的安全设备之上。

4. 手形识别

手形的测量比较容易实现,对图像获取设备的要求较低，手形的处理相对也比较简单，在所有生物特征识别方法中手形认证的速度是最快的。然而手形特征并不具有高度的唯一性，不能用于识别，但是对于一般的认证应用，它足可以满足要求。目前手形认证主要有两种方法：基于特征矢量的方法和基于点匹配的方法。

基于特征矢量的手形认证：大多数的手形认证系统都是基于这种方法。典型的手形特征包括：手指的长度和宽度、手掌或手指的长宽比、手掌的厚度、手指的连接模式等。用户的手形表示为由这些特征构成的矢量，认证过程就是计算参考特征矢量与被测手形的特征矢量之间的距离，并与给定的阈值进行比较判别。

基于点匹配的手形认证：上面方法的优点是简单快速，但是需要用户很好地配合，否则其性能会大大下降。采用点匹配的方法可以提高系统的鲁棒性，但这是以增加计算量为代价的。点匹配方法的一般过程为:抽取手部和手指的轮廓曲线;应用点匹配方法，进行手指的匹配；计算匹配参数并由此决定两个手形是否来自同一人。

5. 掌纹识别

与指纹识别相比，掌纹识别的可接受程度较高，其主要特征比指纹明显得多，而且提取时不易被噪声干扰。另外，掌纹的主要特征比手形的特征更稳定和更具分类性，因此掌纹识别应是一种很有发展潜力的身份识别方法。

手掌上最为明显的 3～5 条掌纹线，称为主线。在掌纹识别中，可利用的信息有：几何特征，包括手掌的长度、宽度和面积；主线特征；皱褶特征；掌纹中的三角形区域特征;细节特征。目前的掌纹认证方法主要是利用主线和皱褶特征。

掌纹特征抽取有两类方法，一是抽取特征线，二是抽取特征点。抽取特征线的优势在于可以用于低分辨率和有噪声的图像,抽取特征点的好处是抽取的速度

快。

掌纹特征匹配对应于掌纹特征的抽取。特征匹配分为特征线匹配和特征点匹配。特征线匹配是计算两幅图像对应特征线参数之间的距离，特征点匹配是两幅图像的两个点集之间的几何对准过程。

6. 声音识别

声音的变化范围比较大，很容易受背景噪声、身体和情绪状态的影响。一个声音识别系统主要由三部分组成：声音信号的分割，特征抽取和说话人识别。

1. 声音信号的分割：目的是将嵌入到声音信号中的重要语音部分分开，通常采用以下几种方法：能量阈值法；零交叉率和周期性的测量；声音信号倒频谱特征的矢量量化；与说话人无关的隐马尔可夫字词模型。
2. 声音特征抽取：人的发声部位可以建模为一个由宽带信号激励的时变滤波器，大部分的语音特征都与模型的激励源和滤波器的参数有关。倒频谱是最广泛使用的语音特征抽取技术，由标准倒谱发展了 mel 整形倒谱和 mel 频率倒谱系数。此外，语音特征参数还包括全极点滤波器的脉冲响应、脉冲响应的自相关函数、面积函数、对数面积比和反射系数。
3. 说话人识别：说话人识别的模型有两种，参数模型和非参数模型。两个主要的参数模型是高斯模型和隐马尔可夫模型（HMM）。HMM 是当前最为流行的说话人识别模型。非参数模型包括参考模式模型和连接模型。参考模式模型将代表说话人的声音模式空间作为模板储存起来，应用矢量量化、最小距离分类器等进行匹配。连接模型包括前馈和递归神经网络，多数神经网络被训练作为直接将说话人分类的判决模型。

2.6 根据实体的行为特征进行身份认证

人的下意识动作也会留下一定的特征，不同的人的同一个动作会具有不同的特征。手书签字是这方面最常见的例子。由于发展的需要，机器自动识别手书签字的研究得到了广泛的重视，成为模式识别中的重要研究课题之一。目前签名大多还只用于认证。签名认证的困难在于，数据的动态变化范围大，即使是同一个人的两个签名也绝不会相同。签名认证按照数据的获取方式可以分为两种：离线（offline）认证和在线（online）认证。离线认证是通过扫描仪获得签名的数字

图像；在线认证是利用数字写字板或压敏笔来记录书写签名的过程。离线数据容易获取，但是它没有利用笔划形成过程中的动态特性，因此较在线签名容易被伪造。

从签名中抽取的特征包括静态特征和动态特征，静态特征是指每个字的形态，动态特征是指书写笔划的顺序、笔尖的压力、倾斜度以及签名过程中坐标变化的速度和加速度。目前提出的签名认证方法，按照所应用的模型可以归为三类：模板匹配的方法，隐马尔可夫模型（HMM）方法，谱分析法。模板匹配的方法是计算被测签名和参考签名的特征矢量间的距离进行匹配；HMM 是将签名分成一系列帧或状态，然后与从其它签名中抽取的对应状态相比较；谱分析法是利用倒频谱或对数谱等对签名进行认证。

一个人敲击键盘的行为特征也可以被用来进行身份认证。击键特征是一种基于击键间隔、击键压力、击键持续时间、击键位置（在按键的边缘或中间）的行为特征。这种特征是唯一的，和手书签名一样。击键识别可以是动态的也可以是静态的。静态识别一次性完成，通常在认证时需要输入一个固定的或已知的字符串。静态认证一旦完成，攻击者可以在不被察觉的情况下获取这个连接，比如趁合法用户暂时离开时控制其终端。动态识别则贯穿整个会话过程，因此上述的攻击是不可行的。从某个用户的击键行为得到的统计信息将用于整个统计测试过程，依据技术的不同，也许会丢弃一些无效数据。

2.7 认证协议

认证协议按照验证的方向可分为双向认证协议和单向认证协议，按照使用的密码技术又可分为基于对称密码的认证协议和基于公钥密码的认证协议。

单向认证有时不需要通信双方同时在线，如电子邮件。一方在向对方证明自己的身份的同时，即可发送数据，另一方收到后，首先验证发送方的身份，如果身份有效，就接受数据。单向认证协议中只有一方向另一方证明自己的身份，其过程相对简单。

双向认证就是使通信双方互相验证对方的身份，适用于双方同时在线的场合。双向认证协议包括了基于对称密码的双向认证协议、基于公钥密码的双向认证协议。

2.7.1 几种常用的认证协议

身份认证方式是基于静态口令的认证方式，它是最简单、应用最普遍的一种身份认证方式。但它存在很多安全问题：它是一种单因素的认证，安全性仅依赖于口令，口令一旦泄露，用户即可被冒充；易被攻击，采用窥探、字典攻击、穷举尝试、网络数据流窃听、重放攻击等很容易攻破该认证系统。在维护网络安全的实际操作中，常常是将身份认证的几个基本方式加以组合（即标记（Token）和口令相结合的方式）来构造实际的认证系统，提高认证的安全性和可靠性。

1 一次性口令认证

窃取系统口令文件和窃听网络连接以获取用户 ID 和口令是网络环境下最常见的攻击方法。在网上传递的口令只使用一次的情况下，攻击者是无法用窃取的口令的方式来访问系统。一次性口令系统（OTP：One Time Password）就是为了抵制这种重放攻击而设计的。一次性口令认证也称为动态口令认证。

一次性口令认证的主要思路：在登录过程（即身份认证过程）中加入不确定因素，使每次登录过程中传送的信息都不相同，以提高登录过程的安全性。例如：登录密码=Hash(用户名＋口令＋不确定因素)，系统接收到登录口令后做一个验算即可验证用户的合法性。Hash 指单向杂凑函数，这样即使攻击者窃听到网络上传输的数据，采用重放攻击方式试图进入系统时，由于不确定因素的变化，使之不能登录。

根据因素的不确定选择不同的方式，一次性口令系统大致分为以下几种：

（1）口令序列：口令为一个单向的前后相关的序列，系统只用记录第 N 个口令。用户用第 N－1 个口令登录时，系统用单向算法计算出第 N 个口令，并与自己保存的第 N 个口令匹配，来判断用户的合法性。由于 N 是有限的，用户登录 N 次后必须重新初始化口令序列。1991 年贝尔通信研究中心（Bellcore）开发的 S/KEY 产品就是采用的口令序列，它是一次性口令系统的首次实现。

（2）挑战-回答：用户要求登录时，系统产生一个随机数发送给用户作为挑战（Challenge），用户用某种单向 Hash 函数将自己的秘密口令和随机数混合起来计算出一个“杂凑值”发送给系统作为回答（Response），系统用同样的方法进行验算即可验证用户的身份。由于 Hash 函数的单向性，用户和系统都很容易算出“杂凑值”（hash 值）来传输和判断；而对于攻击者来说，由 hash 值来推出

用户的口令是不可能的。目前，这一机制已经得到了广泛应用，Windows NT 的用户认证和 IPSec 协议中的密钥交换（IKE）就采用了这一技术。

（3）时间同步：以用户登录时间作为随机因素。美国 RSA 公司的产品 SecureID 中就是采用的这种一次性口令技术。发给每个用户一个身份令牌，该令牌以一定的时间间隔产生新的口令，验证服务器会跟踪每个用户的 ID 令牌产生的口令相位。这种方式对双方的时间准确度要求较高，一般采取以分钟为时间单位的折中办法。在 SecureID 产品中，对时间误差的容忍可达±1 min。

（4）事件同步：以挑战/回答方式为基础，将单向的前后相关序列作为系统的挑战信息，以避免用户每次输入挑战信息。但当用户的挑战序列与服务器产生偏差后，需要重新同步。

2　Kerberos 认证

Kerberos 是麻省理工学院为分布式网络设计的可信第三方认证协议。网络上的 Kerberos 服务起着可信仲裁者的作用，它可提供安全的网络认证，允许个人访问网络中不同的机器。Kerberos 基于对称密码技术（采用 DES 进行数据加密，但也可用其他算法替代），它与网络上的每个实体分别共享一个不同的密钥，是否知道该密钥便是身份的证明。

（1）Kerberos 协议原理

在 Kerberos 协议中，AS 为认证服务器，在用户登录时确认用户身份。AS 与密钥分配中心 KDC 类似，与每个用户共享一个密钥。TGS 为票据分配服务器，为用户之间的通信分配票据，使应用服务器相信 TGS 持有者身份的真实性。Kerberos 协议具体实现过程如图 2.1 所示。

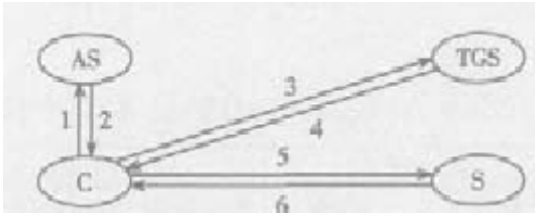


图 2.1　Kerberos 协议的认证过程

从图 2.1 可知，用户 C 要访问目标服务器 S，需要进行六次协议交换，即：

① C->AS:C,TGS,Addr,TS1。

② AS->C:{Kc,tgs}Kc,TGT,TGT:{TGS,C,Addr,TS2,Lifetime2,Kc,tgs}Ktgs。

③ C->TGS:S,TGT,Authenticator1;Authenticator1:{C,Addr,TS3}Kc,tgs。

④ TGS->C:{Kc,s}Kc,tgs,Ts;Ts:{S,C,Addr,TS4,Lifetime4,Kc,s}Ks。

⑤ C->S:S,Ts Authenticator2;Authenticator2:{C,Ad-dr,TS5}Kc,s。

⑥ S->C:{TS5+1}Kc,s。

（2）新 Kerberos 协议规范

Kerberos 协议是建立在一些假设之上的，即只有在满足如下假定的环境中它才能正常运行：不存在拒绝服务攻击;主体必须保证他们的私钥安全；Kerberos 无法应付口令猜测攻击；网络上每个主机的时钟必须是松散同步的。因此，Kerberos 协议不是完全安全的，并且也不能自动提供安全。ITEF 对 Kerberos 协议规范(称为旧规范) 进行了改进，改进的 Kerberos 协议规范(称为新规范)，新规范详细阐明了很多在旧规范上没有清楚描述的条款，增加了一些推荐性的执行选项和一些新操作。

新 Kerberos 协议主要改进有如下几方面:

① 为了与当前应用相适应，新规范采用了新的加密和校验方法，这是最主要的改进。同时删除了一些已不够强壮的方法，如 DES 和 MD5，而采用 AES。表 2-1 为 DES 与 AES(Rijndael)加密算法的性能比较。

表 2-1 DES 与 AES(Rijndael)加密算法的性能比较

ITEM	DES	AES(Rijndael)
密钥长度	密钥长度是 64bit,而有效的密钥长度为 56bit。运行速度稍慢	Rijndae 算法根据安全级别的高低可以自由选择密钥长度(128/192/56)三种,明显提高了算法的灵活性和安全性,运行速度也快
是否存在弱密钥	存在弱密钥和半弱密钥,降低了 DES 算法的安全性	Rijndael 算法由于其密钥扩展函数的特点,所产生的轮密钥的随机性强,对初始密钥的选取没有特别的限制
是否具有对称性	具有。互补对称性可以使 DES 在选择明文攻击下所需要的工	Rijndael 的均匀对称结构既可提高执行的灵活度,又可有效防止差分攻击和线性攻击

	作量减半	

针对密钥的生成，在新规范中，用户模式下私钥“可能”来自用户的口令，因为用户密钥可能存储在智能卡上，或者可以直接获得而与口令无关，而以前用户私钥仅通过用户输入口令生成。

② 新规范依赖 KDC 检查传输域,并将检查标志包含在票据内，以表明该检查已经执行。目前 Kerberos 的执行即使忽略域标志或者不设置域标志也不存在安全隐患。新规范增强了解析主机名的能力，当 Kerberos 为一个命名主体提供认证时,能够确保它所认证的主体名字是完整的,并且就是它所期望通信的对象。

③ 新规范首次在参考文献中提出应用公开密钥算法对认证进行初始化。

④ 新规范增强了 Kerberos 的扩展性及兼容性。

当然，新规范虽然弥补了旧规范的许多环境缺陷和技术缺陷，但由于历史原因,它还是存在许多局限性。作为一个认证服务，Kerberos 在网络上为主机身份的验证提供了一种方法，但 Kerberos 本身并不提供认证。如应用程序不应该接受 Kerberos 服务器上所发布的服务票据作为授权票据，因为在这种情况下可能会使应用程序在其他密钥分发系统内部变得十分脆弱。

3 公钥认证体系

采用前述两种认证协议其主要特点是必须拥有一个密钥分配中心(KDC)或中心认证服务器，该服务器保存所有系统用户的秘密信息，这对于一个比较方便进行集中控制的系统来说是一个较好的选择。当然，这种体制对于中心数据库的安全要求是很高的，因为一旦中心数据库被攻破，整个系统将崩溃。

随着网络应用的普及，对系统外用户进行身份认证有时也是必须的，即某个用户没有在一个系统中注册，但也要求能够对其身份进行认证，尤其是在分布式系统中，这种要求格外突出，也显示了公钥认证技术的独特优越性。

公钥认证协议中每个用户被分配给一对密钥(也可由自己产生)，称之为公钥和私钥，其中私钥由用户妥善保管，而公钥则向所有人公开。这一对密钥是配对使用的，如果用户能够向验证方证实自己持有私钥，就证明了自己的身份。当它用作身份认证时，验证方需要用户方对某种信息进行数字签名，即用户方以用户私钥作为加密密钥，对某种信息进行加密，传给验证方，而验证方根据用户方预

先提供的公钥作为解密密钥，就可以将用户方的数字签名进行解密，以确认该信息是否是该用户所发，进而认证该用户的身份。

公钥认证体制中要验证用户的身份，必须拥有用户的公钥，而用户公钥是否正确，是否是所声称拥有人的真实公钥，在认证体系中是一个关键问题。常用的办法是找一个值得信赖而且独立的第三方认证机构充当认证中心（Certificate Authority, CA），来确认声称拥有公开密钥的人的真正身份。任何想发放自己公钥的用户，可以去认证中心申请自己的证书。CA 中心在认证该人的真实身份后，颁发包含用户公钥的“数字证书”，数字证书又叫“数字身份证”、“数字 ID”，它是包含用户身份的部分信息及用户所持有的公钥相关信息的一种电子文件，可以用来证明数字证书持有者的真实身份。CA 利用自身的私钥为用户的“数字证书”加上数字签名，可以保证证书内容的有效性和完整性。其他用户只要能验证证书是真实且完整的（用 CA 的公钥验证 CA 的数字签名），并且信任颁发证书的 CA，就可以确认用户的公钥。

所有 CA 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”即公认权威机构位于最顶端，没有上一级节点，故不受此限。在两方通信时，通过出示由某个 CA 签发的证书来证明自己的身份，如果对签发证书的 CA 本身不信任，则可验证 CA 的身份，依次类推，一直到“Root CA”处，就可确定证书的有效性。

要建立安全的公钥认证系统，必须先建立一个稳固、健全的 CA 体系，尤其是公认的权威机构，即“Root CA”，这也是当前公钥基础设施（PKI）建设的一个重点。CA 目前采用的标准是 X.509，X.509 中定义的证书结构和身份认证协议已经在各种环境中实际使用了，它对所用具体加密、数字签名、公钥密码以及 Hash 算法未作限制，必将会有广泛的应用，已纳入 PEM（Privacy Enhanced Mail）系统中。X.509 是定义目录服务建议 X.500 系列的一部分，其核心是建立存放每个用户的数字证书的目录(仓库)。用户数字证书由可信赖的 CA 创建，并由 CA 或用户存放于目录中供检索。

在实际应用中，若验证方想获得用户的公钥，验证方先在目录中查找用户的“数字证书”，利用 CA 的公钥和 Hash 算法验证用户“数字证书”的完整性，从而判断用户的公钥是否正确。

采用数字证书进行身份认证的协议有很多，SSL(Secure Socket Layer)和SET(Secure Electronic Transaction)是其中的典型例子。验证方向用户提供一个随机数，用户以其私钥（Kpri）对该随机数进行签名，将签名和自己的证书 Cert 提交给验证方，验证方验证证书的有效性，从证书中获得用户公钥（Kpub），以 Kpub 验证用户签名的随机数。

2.7.2 常用认证协议的分析与比较

从目前的发展来看，一次性口令认证协议实现最为简便，而 Kerberos 实现起来较为繁琐，用户方和服务器方共享一个秘密信息，以加密的方式传送该秘密信息，服务器方保存所有用户的秘密信息以备进行认证，两者都适用于系统对用户的单向认证。随着电子商务的广泛开展，对于系统与系统之间的双向认证，公钥认证显得越发重要，而要安全正确地使用公钥认证，就必须大力加强 PKI 建设。PKI 是在公开密钥理论和技术基础上发展起来的一种综合安全平台，能够为所有网络应用透明地提供采用加密和数字签名等密码服务所必需的密钥和证书管理，从而达到保证网上传递信息的安全、真实、完整和不可抵赖的目的。

2.8 分布式环境与移动环境下的身份认证

2.8.1 分布式计算环境下的身份认证

在分布式计算环境之下，用户访问系统时的位置是可变的，同时用户所要访问的系统资源也不是固定的。Kerberos 提供了一种具有较高安全性能的用户身份认证和资源访问认证的机制。在 Kerberos 认证体制中，除了认证服务器 AS 外，还有另外一种授权服务器 TGS（Ticket Granting Server）。AS 中保存了所有用户的口令。用户登录系统并表明访问某个系统资源时，系统并不传送用户口令，而是由 AS 从用户口令产生一个密钥 $K_{U,AS}$，并传送给用户 U 一个可以访问 TGS 的门票 Ttgs。用户 U 将获得的 Ttgs 连同其个人化信息发送给 TGS。TGS 对用户身份信息认证后，发送给用户 U 一个可以访问某个服务器的门票 TS。用户 U 将获得的 TS 连同其个人化信息发送给 Server。Server 对信息认证后，给用户提供相应的服务。协议认证的具体过程见图 2.1。

Kerberos 的特点：

①与授权机制相结合。

②实现了一次性签放机制，且签放的票据有一个有效期。

③支持双向的身份认证，即服务器可以通过身份认证确认客户方的身份，如果需要，客户也可以反向认证服务方的身份。

④通过交换“跨域密钥”来实现分布式网络环境下的认证机制。

⑤安全性强，能防止攻击和窃听，能提供高可靠性和高效的服务，具有透明性（用户除了发送 Password 外，不会觉察出认证过程），可扩充性好。

Kerberos 的设计是与M IT 校园网环境结合的产物, 在分布式系统中应用时也存在一些局限性.。首先, 原有的认证码很有可能被存储或替换. 虽然时戳是专门用于防止这类重放攻击的, 但在票据的有效时间内仍然可能奏效。假定在一个 Kerberos 服务域内的全部时钟保持同步, 收到消息的时间在规定范围内(一般规定 t= 5 分钟) 就认为该消息是新的。而事实上, 攻击者可以事先把伪造的消息准备好, 一旦得到认证码就马上发出, 这在 5 分钟的时间内是难以检查出来的。其次, 认证码的正确性是基于网络中所有的时钟保持同步, 如果主机的时间发生错误, 原来的认证码毫无疑问可以被替换。大多数网络的时间协议都是不安全的, 而在分布式系统中这将导致极为严重的问题。再次, Kerbero s 防止口令猜测攻击的能力很弱, 攻击者可以收集大量的票据, 通过计算和密钥分析进行口令猜测. 当用户选择的口令不够强时, 更不能有效地防止口令猜测攻击。实际上, 最严重的攻击是恶意软件攻击，Kerberos 认证协议依赖于 Kerberos 软件的绝对可信, 而攻击者可以用执行 Kerberos 协议和记录用户口令的软件来代替所有用户的 Kerbero s 软件, 达到攻击目的. 一般而言, 装在不安全计算机内的密码软件都会面临这一问题. 另外, 在分布式系统中, 认证中心星罗棋布, 域间会话密钥的数量惊人, 密钥的管理、分配、存储等都是很严峻的现实问题。

为了解决上述问题，有很多学者提出了改进的 Kerberos 分布式认证协议。例如有学者采用 Yaksha 体制的思想, 对 Kerberos 进行优化， 联网时可不使用口令, 提高了安全性；在认证过程中, 由用户对自己加盖的时戳进行验证, 解决了 Kerbero s 时间同步问题, 并有效地防止重放攻击；采用 RSA 公钥加密算法进行认证虽比单钥加密算法速度慢, 但简化了认证步骤, 减少了加解密次数, 缩短

了用户机的等待时间, 还是能够达到实时认证的速度要求的。

2.8.2 移动环境下的用户身份认证

近些年来移动通信的发展日新月异，移动平台上传输的内容不再局限于传统的语音业务，大量敏感的数据业务，如移动电子商务、移动电子银行、移动购物等也出现在移动平台上，在移动通信中，用户和固定的通信机站之间并不存在一条通信线路#，电波在开放的空间中传送使得任何人都可以轻易的接收到通信的信号，此外由于用户可以在任意的区域内移动而不受限于某一固定点的通信，也使攻击者有了窃听和非法使用的机会，因此移动通信的安全性成为一种必不可少的需求。

移动环境下的安全性要解决的问题就是保证数据在传输过程中的机密性、完整性、有效性。而对于移动数据业务来说，首先就需要解决有效性，即实现对用户身份的识别－身份认证。由身份认证系统来解决移动通信实体的身份确认，使得通信双方能够确信对方就是声明和自己通信的实体。

(1)GSM 认证体制

GSM 是全球最成熟的数字移动电话网络标准之一，在全世界的 162 国家已经建设了 400 个通信网络，全球 80%以上的移动用户为 GSM 用户，用户数已经超过了 10 亿人。

在 GSM 中并不是通过手机号码来进行身份认证的，而是通过存储在手机 SIM(Subscriber Identity Module，用户识别模块)中的用户认证密钥 K 和 IMSI(International Mobile Subscriber Identity, 跨国移动用户识别) 或 TMSI(Temperar Mobile Subscriber Identity 临时移动用户识别)号码来确定使用者的身份，也就是说 GSM 中采用了两个认证协议：IMSI 认证协议和 TMSI 认证协议

在 IMSI 认证协议中，当用户初次接入 GSM 时，由移动台（Mobile Station）传送 SIM 中的 IMSI 给 VLR(异地系统注册中心，处理用户所在地访问的实时证实和接入控制，临时注册)，再由 VLR 传送 HLR（本地系统注册中心，处理本地的实时证实和接入控制，永久注册）进行身份认证。而后由 VLR 为用户分配一个 TMSI 给用户，并在成功认证后以加密形式将其传给 MS，MS 解密后存储 TMSI

到 SIM 中，此后 MS 用 TMSI 进行认证，直到新的 TMSI 被分配给用户。

TMSI 的设置是为了防止非法个人或团体通过监听无线电波而窃得移动用户的 IMSI 或跟踪移动用户的位置。TMSI 是由交换中心分配，并不断地进行更换，更换周期由网路运营者设置。

用户从一个 VLR 移动到新的 VLR 时候， 新的 VLR 向原 VLR 要求 IMSI，TMSI，LAI（Location Area Identification）和三元组（RAND，SRES，Kc），其中，SRES 是一个由 K 以及随机数 RAND 生成的 HASH。在 TMSI 认证过程中，首先用户传送 SIM 中的 TMSI 给 VLR，VLR 发送 RAND 给用户。用户计算 SRES 和 Kc 后，将 SRES 传给 VLR，由 VLR 与原存贮的 SRES 进行比较。认证通过后，通信双方就可以通过密钥 Kc 进行数据的加密，如图 2.2 所示。

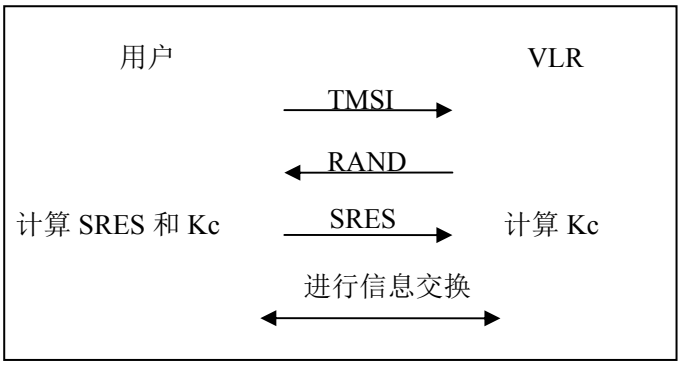


图 2.2 TMSI 认证示意图

GSM 认证也存在一些缺陷，比如: 是单向身份认证，MS 不能认证网络，无法防止伪造网络设备如基站的攻击；加密密钥 Kc 及认证数据等安全相关信息在网络中使用明文进行传输，易造成密钥信息泄露；基站和基站之间均是明文传送，无加密和认证措施；用户身份认证密钥 K 不可变，无法抗击重放攻击；无消息完整性认证，无法保证数据在链路中传输过程中的完整性；用户漫游时，从一个网络进入另一个网络没有相关性等等。这些安全缺陷都应当考虑进行改进和解决。

(2) WAP 移动认证系统

随着电子商务架构逐步服务于无线终端，WAP(Wireless Application Protocol)无线应用协议已经在移动经济中占有重要的一席之地#。目前 WAP 的应用包括股票买卖、网上账单缴付、互联网游戏、新闻等等。 对于 WAP 系统的关键就是要对使用者的身份进行确认以便作为移动业务消费记录和形成账单的依据。

移动系统由于受到移动通信设备和移动通信技术上的限制，而不能照搬固定网络中的安全技术。 比如手机终端的加密解密能力有限，不能进行大数据量的计算，手机容易丢失或遭窃。他人可能使用该手机冒名与 WAP 服务器通信等等。 因此必须有一套完整且有效的认证协议，以保证无线通信过程的安全。

WAP 的身份认证可以通过调用 WMLScript 在应用层实现认证，也可以通过 WAP 网关结合 CA 证书的方式进行认证。

目前大部分移动终端都能够支持 WML 和 WMLScript，可以通过 WMLScriptCrypto API 在应用层实现客户端的加密、解密、签名和验证。WMLScript Crypto API 是 WAP2.0 标准中重要内容，在库中存有许多密码算法如 RSA，SHA，DES 等常用加密算法。它可以加强手机终端的安全能力，服务商只要通过在网页中调用脚本加密函数库就可以完成提交信息的加密、签名以及证书的认证等操作。在应用层对数据加密后 WAP 网关获得的数据是经过加密后的数据，从而最大程度的保障数据端到端的安全性。

随着技术的发展，SIM 已经越来越不能满足应用的需要。 未来的手机身份认证卡是一种具有更安全加解密功能和更大存储空间的智能卡。如 WIM（WAP Identity Module）。它能够存储数字证书和用户密钥并能够进行加密解密运算，例如 Gemplus 公司就开发了几款具有 PKI 功能的应用于移动环境下的智能卡，有很高的安全性"，这种卡很适合用于移动银行和移动电子商务、股票交易、购票等对安全性要求很高的业务。

习题二

1 外部实体必须提供信息使得系统能够证明它的身份，根据你的想法对这些信息（认证方式）进行分类，并说明其合理性。

2 列举身边的几项身份认证实例，并探讨其使用的技术。

3 查阅相关文献，总结应用于 Web 的身份（IP 地址、主机名）认证与通常的身份认证有何异同。

4 Kerberos 协议和新 Kerberos 协议的具体认证过程包括哪些？

5 查阅相关文献，总结 Kerberos 协议和新 Kerberos 协议的优势与缺陷。

6 身份认证技术还有哪些有希望的发展方向。

第 3 章　访问控制基础知识

当用户通过了身份鉴别的验证后，便成为系统的合法用户，取得了对系统合法的访问权限。但是，当他对系统资源进行具体访问之前，还必须受到系统的检查和控制，以使他只能在系统对其授权的范围内活动。本章介绍访问控制涉及的一些基本概念和最基础的知识。

3.1　基本概念

访问控制涉及的对象主要有两类：客体和主体。客体是访问控制要保护的对象，主体是访问控制要制约的对象。

在系统中，包含有信息，又可以被访问的实体称之为**客体**（Object）。它们是一种信息实体，能从主体或其他客体接收信息。诸如操作系统中的文件、存储页、存储段、目录、进程间的报文、I/O 设备；数据库中的表、记录、网络节点等。

能访问或使用客体的活动实体称为**主体**（Subject），它可使信息在客体之间流动。用户是主体，系统内代表用户进行操作的进程自然被看作是主体。

与其它类型的数据一样，驻在内存或存于磁盘上的程序被看作是客体。但是，一旦程序运行，它就成为主体或进程的一部分。因此，对于一个程序来说，有时它的身份是客体，而有时它的身份又是主体。系统内所有的活动都可看作是主体对客体的一系列操作。

访问权（access right）描述主体访问客体的方式。通常包含以下几种方式：

读（read）：主体可以查看系统资源的信息。如文件、记录或记录中的字段。

写（write）：主体可以对系统资源的数据进行添加、修改或删除。写权限往往包括读权限。

添加（append）：仅允许主体在系统资源的现有数据上添加数据，但不能修改现存的数据。例如在数据库表中增加记录。

删除（delete）：主体可删除某个系统资源，如文件或记录。

执行（execute）：主体可以执行指定的程序。

另外还有两个特殊的权限：

拥有（Own）：若客体 o 是由主体 s 所创建，则主体 s 对 o 具有拥有权，称 s 是 o 的拥有者。

控制（control）：主体 s 对客体 o 的控制权表示 s 有权授予或撤销其他主体对 o 的访问权。

在实际系统的开发中，开发者可根据需要定义各种方式的访问权，赋予各自特定的含义，然后加以控制，如也可定义用户创建新文件或记录等的**创建**（create）权。

3.2 基本的访问控制方法

访问控制的方法不是唯一的，通常是根据实际系统的安全需求来决定采用什么样的控制方法。也可能在一个系统中同时采用多种控制方法，以实现在最大限度提供信息资源服务的情况下确保系统的安全。目前人们常用的有如下三种方法：自主访问控制，强制访问控制和基于角色的访问控制。

3.2.1 自主访问控制

自主访问控制(Discretionary Access Control,简称 DAC)是这样的一种控制方式，对某个客体具有拥有权的主体能够将对该客体的一种访问权或多种访问权自主地授予其它主体，并在随后的任何时刻将这些授权予以撤销。也就是说在自主访问控制下，用户可以按自己的意愿，有选择地与其它用户共享他的文件。

若主体 s 创建了客体 o，则 s 是 o 的拥有者，对 o 具有拥有权，与此同时，s 也具有了对客体 o 所有可能的访问权，例如对 o 的读、写、添加、删除等权限。若 o 是程序，那么还有执行权。特别是，s 也自动具有了对 o 的控制权，即 s 能在系统中决定哪些主体对 o 有访问权，有什么样的访问权。

自主访问控制是保护系统资源不被非法访问的一种有效手段。但这种控制是自主的，即它是以保护用户的个人资源的安全为目标并以个人的意志为转移。虽然这种自主性满足了用户个人的安全要求，并为用户提供了很大的灵活性，但它对系统安全的保护力度是相当薄弱的。当系统中存放有大量数据，而这些数据的属主是国家，是整个组织或整个公司时，谁来对它们的安全负责呢？为了保护系统的整体安全，必须采取更强有力的访问控制手段来控制用户对资源的访问，这

就是强制访问控制。

3.2.2 强制访问控制

强制访问控制(Mandatory Access Control,简称 MAC)是指，一个主体对哪些客体被允许进行访问以及可以进行什么样的访问，都必须事先经过系统对该主体授权，这种授权与系统的应用背景密切相关。一般来说，系统根据用户在应用业务中的职务高低或被信任的程度，以及客体所包含的信息的机密性或敏感程度来决定用户对客体的访问权限的大小，这种控制往往可以通过给主、客体分别赋以安全标记，并通过比较主、客体的安全标记来决定。也可以通过限制主体只能执行某些程序来实现。

自主访问控制和强制访问控制实际上是人们日常各种政治，经济，商业等活动中对信息资源安全性保护用计算机来实现的两种不同方式。例如，某人甲写了一篇文章，想在相关专业的刊物上发表，为了使文章不出现错误并且有更高的质量，在寄出去之前他希望同事乙帮助看一看，提些意见和建议，因此他愿意让乙阅读这篇文章。其他的人，如果没有他的允许就不可能看到他的文章，而只有在文章发表之后才能看到。但是如果甲是受机构委托撰写一份文件，要求在文件定稿并正式下发之前不能向外泄漏，那么，这时甲就不能自作主张将文件草稿给谁看。而且成文后，文件下发的范围也必须由机构的相关领导决定，而不能由甲个人决定。这就是机构对这份文件访问权的强制性控制。

有些系统往往将自主访问控制与强制访问控制结合使用。在这种系统中，一个主体只有既通过了自主访问控制的检查又通过了强制访问控制检查，才能访问某个客体。用户可以利用自主访问控制来防范其它用户对自己所拥有的客体的攻击，强制访问则提供了一个不可逾越的更强的安全保护层。

强制访问控制不仅能阻止对系统的恶意攻击，也可以防止由于程序错误或用户的误操作所引起的泄露和破坏。

系统如何进行访问权限的强制控制呢？系统根据什么来决定允许或不允许某个主体对某个客体的访问呢？根据什么来决定允许某个主体对某客体进行这样的访问而不能进行那样的访问呢？这往往是由系统的应用背景的安全需求所决定的。不同的应用有不同的安全需求，系统必须根据这些不同的安全需求，制定出不同的安全策略，来描述系统必须达到的安全目标或它必须抵抗的威胁。

3.2.3 基于角色的访问控制

随着商业和民用信息系统的发展，安全需求也在发生变化并呈现出多样化，这些系统对数据完整性的要求可能比对保密性的要求更突出。而且，由于诸如部门增加、合并或撤销，公司职员的增加或裁减，使得系统总是处于不断变化之中，这些变化使一些访问控制需求难以用 DAC 或 MAC 来描述和控制。而且在许多机构中，即使是由终端用户创建的文件，他们也没有这些文件的“所有权”。访问控制需要由用户在机构中承担的职务或工作职责，或者说由用户在系统中所具有的角色来确定。例如，一个银行的出纳员、会计和行长，一个学校的行政职员、教师和校长等都是一些不同的角色，他们具有不同的职责，对系统的访问权限也就不同。因此利用社会中角色的概念帮助系统进行访问控制管理的思想应运而生。

早在二十世纪七十年代初，多用户、多应用联机系统出现时，就有人提出了角色的概念，但一直没有得到专家们的重视。直到八十年代末、九十年代初，NIST（美国国家标准技术研究所）的安全专家们如 David F. Ferraiolo 等对 TCSEC（可信计算机系统评估准则）的安全标准是否可靠，传统的 MAC 和 DAC 是否真正能够胜任不断增长的安全需求提出疑问，并且提出了一种新的访问控制技术，基于角色的访问控制(Role-Based Access Control,简称 RBAC)技术，才逐步引起了安全专家们的注意。一些著名的访问控制专家，如 George Mason University 的 Ravi Sandhu 教授也将研究转向了 RBAC。1995 年 Ravi Sandhu 等人对 NIST 的成员在 1992 年提出的 RBAC 模型和理论进行了扩展，提出了 RBAC 的基本模型 RBAC0 模型，进而构造了 RBAC96 模型族。

传统的自主访问控制和强制访问控制都是将用户与访问权限直接联系在一起，或直接对用户授予访问权限，或根据用户的安全级来决定用户对客体的访问权限。在基于角色的访问控制中，引入了角色的概念，将用户与权限进行逻辑上分离。

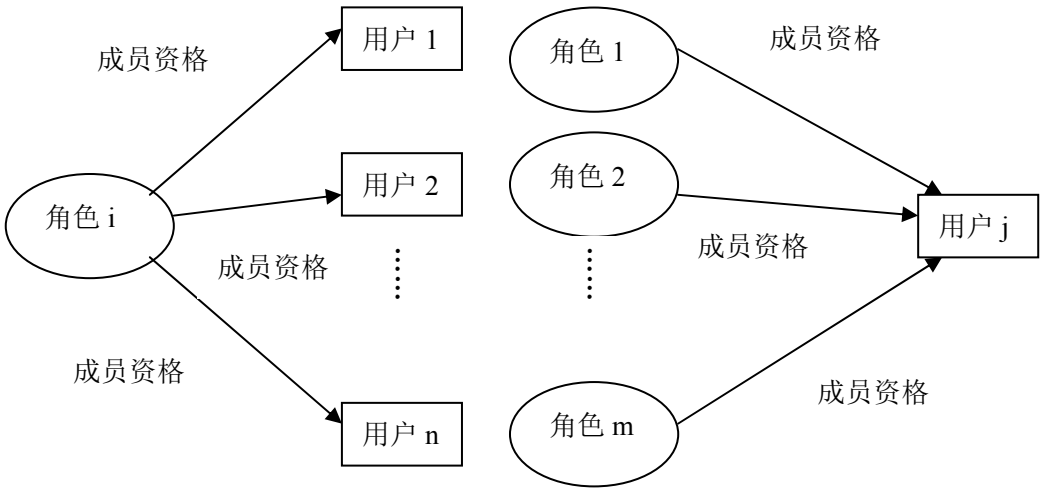
基于角色的访问控制是指在一个组织机构里，系统为不同的工作岗位创建对应的角色，对每一个角色分配不同的操作权限（或操作许可）；另一方面，系统根据用户在机构中担任的职务或责任为其指派相应的角色。用户通过所分配的角色获得相应的操作权限，实现对信息资源的访问。

用户、角色和操作权限三者之间的关系如图 3.1 所示。将操作权限分配给角色，将角色的成员资格分配给用户，用户由所取得的角色成员资格而获得该角色相应的操作权限。这种访问控制不是基于用户身份，而是基于用户的角色身份，

用户——>角色<——操作权限

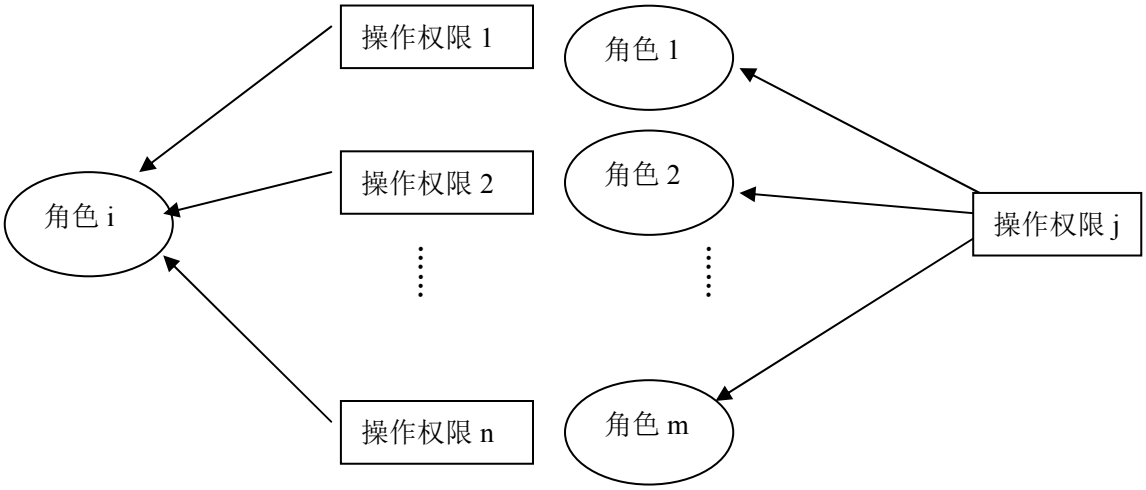
图 3.1 用户、角色和操作权限三者之间的关系

同一个角色身份可以授给多个不同的用户，一个用户也可以同时具有多个不同的角色身份。一个角色可以被指派具有多个不同的操作权限，一种操作权限可以指派给多个不同的角色。这样一来，如图 3.2 和图 3.3 所示，用户与角色，角色与操作权限之间构成多对多的关系。通过角色，用户与权限之间也形成了多对多的关系，即一个用户通过一个角色成员身份或多个角色成员身份可获得多个不同的操作权限，另一方面，一个操作权限通过一个或多个角色可以被授给多个不同的用户。



（a）角色与用户的 1 对多关系　　（b）角色与用户的多对 1 关系

图 3.2 角色与用户的多对多关系



（a）角色与操作权限的 1 对多关系　　（b）角色与操作权限的多对 1 关系

图 3.3 角色与操作权限的多对多关系

这里所说的操作权限与前面的访问控制权略有不同，操作权是指某种访问权施加于某特定客体的权限，即操作权实际上由两部分组成，可表示为一个二元组（访问权，客体）。

角色是 RBAC 机制中的核心，它一方面是用户的集合，而另一方面又是操作权限的集合，作为中间媒介将用户与操作权限联系起来。角色与组概念之间的主要差别是，组通常是作为用户的集合，而并非操作权限的集合。

基于角色访问控制的典型例子是医疗系统，在这种系统中，可以定义如下各种角色：外科医生，内科医生，儿科医生……等各种专科医生，护士，药剂师，检验师……等各种医辅职务。每一种角色都有许多成员，他们在系统中具有相同的操作权限。当一个用户的角色成员资格发生改变时，他所拥有的操作权限也相应会改变。例如，若某医生由内科医生改行为神经科医生，那么他在该系统中的操作权限就不再由内科医生这一角色的权限来决定，而只能在神经科医生这一角色的操作权限范围内。

3.3 安全策略与安全模型

3.3.1 安全策略

系统对主体实施访问控制时必须遵循一定的规则，即哪些主体对哪些客体可以具有什么样的访问权限，而哪些主体对哪些客体的哪些访问又是不允许的。执行访问控制的部件必须严格按照这些规则去控制，然而这些规则又是怎样确定的呢？这些规则是由实施访问控制的系统的安全需求确定的，不同的应用系统有各自不同的安全需求。例如军事系统，还有政府办公系统最重要的安全需求是信息的机密性，不同职务的用户能看到的信息多少和信息的重要程度是不一样的。在商业和金融系统中，信息的机密性虽然也很重要，但更重要的安全需求是信息的完整性。即严防对数据的非法修改，伪造和错误。

反映系统安全需求的规则称为安全策略。因此计算机系统的安全策略是为了描述系统的安全需求而制定的对用户行为进行约束的一整套严谨的规则，这些规则规定了系统中所有授权的访问，是实施访问控制的依据。系统在实施访问控制之前，制定其安全策略是十分必要的。

3.3.2 安全策略举例

1、军事安全策略

军事系统中最重要的安全需求是信息的机密性，他往往根据信息的机密程度对信息分类，将各类机密程度不同的信息限制在不同的范围内，防止用户取得他不应该得到的密级较高的信息。多级安全(Multilevel security)的概念始于 20 世纪 60 年代，是军事安全策略的一种数学描述，并用计算机可实现的方式来定义。

（1）系统中每个主体和每个客体都有安全标记

在实施多级安全策略的系统中，系统为每一个主体和每一个客体分配一个安全级（即安全标记）。对于主体来说，他的安全级表示他在系统中被信任的程度或他在系统中访问信息的能力，有时也称为该主体的许可证。对于客体来说，他的安全级表示该客体所包含信息的敏感程度或机密性。

（2）安全标记由两部分组成

安全级由两部分组成，用有序二元组（密级，范畴集）表示。其中密级用来反映信息（或可访问信息）的机密程度，通常由一般、秘密、机密和绝密四个级

别组成，根据需要可以将其扩充到任意多个级别。它们之间的关系用全序：一般≤秘密≤机密≤绝密来表示，这意味着，若某主体具有访问密级 a 的能力，则对任意 b≤a，该主体也具有访问 b 的能力，反之，则不然。

范畴(Category)集也可理解为部门或类别集，对于客体来说，它的范畴集可以定义为该客体所包含的信息所涉及的范围，如所涉及的部门或所具有的类别属性。对于主体来说，它的范畴集可以定义为该主体能访问的信息所涉及的范围或部门。

例如，假设某应用系统中密级定义为如上四种，分别用 U、C、S 和 TS 表示一般、秘密、机密和绝密。令 A={U, C, S, TS}，这里 U≤C≤S≤TS，“≤”是 A 上的全序，他们构成偏序集<A; ≤>。

假设部门集 B={科技处，生产处，情报处，财务处}，集合 B 的幂集 P_B={S|S⊆B}， P_B（也可表示为 2^B）中的元素均是 B 的子集。显然集合的包含关系“⊆”是 P_B 上的一个偏序关系，因此<P_B; ⊆>也构成偏序集。

笛卡尔积 A×P_B={(a, H)| a∈A, H∈P_B}中的元素可用来表示系统中主、客体的安全级。例如，假设系统中有主体 u 和三个客体 o_1，o_2，o_3，它们的安全级分别如下：

class(u) =(S，{科技处，财务处})

class(o_1) =(C，{科技处})

class(o_2) = (TS, {科技处，情报处，财务处})

class(o_3) =(C，{情报处})

通过比较主、客体的安全级，便可决定是否允许主体对客体的访问以及什么样的访问。

（3）如何比较主、客体的安全级

我们注意到，可以在笛卡尔积 A×P_B 上定义一个二元关系“≤”：对任意（a_1,H_1），（a_2,H_2）∈A×P_B, 当且仅当 a_1≤a_2，H_1⊆H_2 时，有（a_1,H_1）≤（a_2,H_2）。容易证明：“≤”是 A×P_B 上的一个偏序关系，即<A×P_B; ≤>构成一个偏序集(参看参考文献[ ])。

根据上述主体 u 和客体 o_1，o_2 和 o_3 的安全级，可知

class(u)≤class(o_2)，class(o_1)≤class(u)

class(u)与 class(o_3)不可比。

在一偏序集< L; ≤ >中，对任意 l_1，$l_2 \in L$，若 $l_1 \leq l_2$ 则称 l_2 支配 l_1，因此，在这里我们称主体 u 的安全级支配客体 o_1 的安全级，客体 o_2 的安全级支配主体 u 的安全级，但主体 u 和客体 o_3 的安全级相互不可支配。

（4）访问控制规则

多级安全策略可以粗略地描述为：

仅当主体的安全级支配客体的安全级时，允许该主体读访问该客体。

仅当客体的安全级支配主体的安全级时，允许该主体写访问该客体。

这一策略可简称为“向下读，向上写”，更精确地说法是，“不向上读，不向下写”。执行的结果是信息只能由低安全级的客体流向高安全级的客体，高安全级的客体的信息不允许流向低安全级的客体。若要使一个主体既能读访问客体，又能写访问这个客体，两者的安全级必须相同。

根据这一策略，上述的主体 u 可以读访问客体 o_1，可以写访问 o_2，但 u 对于 o_3，既不能读访问，也不能写访问。

在第 4 章 Bell-La Padula 模型的介绍中，我们将会看到，在系统具体实施访问控制时，其控制更为精细。

在多级安全策略中，“写”访问权其含义实际上指的是前面所说的“添加”权。

多级安全策略不仅适用于军事系统，也可适用于政府及企业的办公自动化系统，具有层次结构的组织机构均可使用多级安全策略来保护信息的机密性。

2、商业安全策略

在商业和金融系统中，信息的机密性虽然重要，但更重要的安全需求是信息的完整性，它必须防止非授权的修改，防止数据的伪造和错误。系统的任何用户，即使是授权用户也不允许对数据随意修改，因为这样可能会使公司的资产或帐户记录丢失或毁坏。有两种防伪造和错误的控制方法：良性事务和职责分离。

（1）良性事务

良性事务(Welll-formed Transaction)限制用户对数据的操纵不能任意进行，而应按照可保证数据完整性的受控方式进行。因此，用户实际上受到了他可执行什么样的程序的限制，而他对数据的读、写方式隐含在那些程序的动作之中。

对用户的控制还可以通过一个简单的机制来实现，把用户所有对数据的修改在一个日志中记录下来，使得所有的行为在必要时都可以在事后被审计。删除操作将受到严格的控制，因为任何删除行为都有可能意味着伪造。

财务系统就是良性事务的一个结构化的例子，它们的事务处理遵循双入口规则，双入口规则通过要求记录下的修改部分之间保持平衡以保证系统内部数据的一致性。例如，每签发了一张支票（意味着要进入一次现金帐户），则要求有一个相应的可支付的帐户入口，如果其中一个入口出差错，各部分之间就会不一致，这可以由一个平衡帐簿的独立的测试程序来检查，它可以查出非授权支票等诈骗行为。

（2）职责分散

第二种控制伪造和诈骗的机制是职责分散(Separation of Duty)。把一个操作分成几个子操作，并要求不同的子操作由不同的用户来执行，这样可间接保证数据客体和它所描述的外部世界之间的一致性。例如，一个购买货物并付款的过程，可以由以下几个子操作来完成：授权购买订单、记录到货、记录到货发票、授权付款。最后一个步骤只有在前三个步骤完成之后才能执行。如果每个步骤由不同的用户来执行，外部和内部描述就会一致，除非有人暗中勾结。如果一个人可以执行所有这些步骤，那么伪造是可能的。置入一个订单，支付给一个没有任何发货的虚假公司。在这种情形下，帐面是平衡的。错误发生在真实的事物和记录的货物清单之间的不一致。

职责分散的最基本的规则是，被允许创建或验证良性事务的人，不能允许他去执行该良性事务，这个规则使得若要对一组良性事务进行修改，至少需要两个人的参与才可进行。

如果职员不暗中勾结，则职责分散是有效的。这里的假设可能是不安全的。但在防诈骗的实际控制中已被证明这种方法是有效的。我们还可以采取一些措施使职责分散的威力变得更强大。例如，随机选取一组职员来执行一组操作，合谋的机会就要小的多。

良性事务和职责分散是商业数据完整性保护的基本原则。用于商业数据处理的计算机系统，需要有专门的机制来实施这两条规则。为了保证数据仅由良性事务来处理，首先要保证数据只能由一组指定的程序来操纵。这些程序必须被证明

其构造正确，并要对安装和修改这些程序的能力作出控制，保证他们的合法性。为保证职责分散，每一个用户必须仅被允许使用指定的程序组，并对用户执行程序的权限进行检查以保证达到期望的控制。

这里的数据完整性控制与军事中的数据机密性控制有很大的差别。首先，在这里数据客体不是与一个特定的安全级别相关，而是与一组允许操纵它的程序相联系；第二，用户并不是被授权直接去读或写某一数据，而是被授权去执行与某一数据相关的程序。这样一来，一个用户即便被授权去写一个数据客体，他也只能通过针对那个数据客体定义的一些事务去做。

商业安全策略体现的也是一种强制访问控制，但它是与军事安全策略的安全目标、控制机制不相同的另一类强制控制。它的强制性体现在用户必须通过指定的程序来访问数据，而且允许操纵某一数据客体的程序列表和允许执行某一程序的用户列表不能被系统的一般用户所更改。

以下两条要求对军事安全和商业安全来说是相同的。它们对任何一个安全策略来说都是其不可分割的一部分。

①计算机系统必须有一种机制来保证系统实施了安全策略中的安全需求。

②系统中的安全机制必须防止篡改和非授权的修改。

3.3.3 安全模型

设计一个安全系统的关键是对系统的安全需求有全面、清晰的了解。根据安全需求制定出相应的安全策略，并对安全策略所表达的安全需求进行清晰、准确的描述，建立相应的安全模型。

一个好的安全模型应该能对安全策略所表达的安全需求进行简单、精确和无歧义的描述，它是安全策略的一个清晰的表达方式。一般来说，安全模型应具有以下特点：

1）它是精确的，无歧义的；

2）它是简单、抽象，也是易于理解的；

3）它仅涉及安全性质，不过分限制系统的功能与实现。

安全模型分为非形式化的安全模型和形式化的安全模型。

1、非形式化的安全模型

非形式化的安全模型是用自然语言对系统的安全需求进行描述，这种描述方法，直观、易于理解，但不够严谨，容易产生歧义，并且不简洁。对于安全性要求不高，或改造一个已存在的系统，增强其安全性时，可以使用这种描述方法。

2、形式化的安全模型

形式化的安全模型使用数学符号精确描述系统的安全需求，这种描述方法较非形式化描述方法显得抽象和较难理解，这需要设计或开发人员有较好的数学修养。但是这种描述方法的最大优点是简洁、准确、严谨，不会出现歧义，并可以

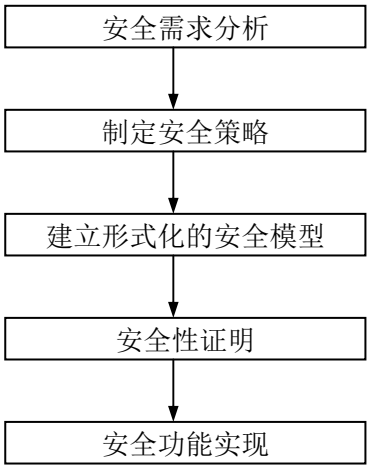


图 3.4　安全系统的开发过程

对其进行形式化的验证或证明。若要设计高安全级的计算机系统，必须建立形式化的安全模型，并对其安全性进行形式化的证明。

一个安全系统的开发过程如图 3.4 所示。

一个系统如果未达到所希望的安全性，原因可能有两个，一是对安全性的定义有缺陷，二是安全控制中有漏洞。第一个问题涉及到定义系统能做什么，不能做什么。相对于系统的功能的定义来说，系统安全的定义是比较复杂的，因为它必须非常精确。第二个问题是软件的可靠性问题，它可以通过与设计技术相关的软件工程手段来克服。

人们根据不同的安全需求，从不同的角度考虑系统安全，分别提出了一些安全模型。

1、Bell-La Padula 模型

Bell-La Padula 模型（简称 BLP 模型）是最早的也是应用较为广泛的一个安全模型。它是由 David Bell 和 Leonard La Padula 于 1937 年创立的符合军事安全

策略的计算机操作模型。这项工作产生于早期 Case Western Reserve University 所做的工作。模型的目标是详细说明计算机的多级操作规则。这种对军事安全策略的精确描述称为是多级安全策略。

因为 BLP 模型是最著名的多级安全策略模型，所以常把多级安全的概念与 BLP 模型联系在一起。事实上，其它有些模型也符合多级安全策略，每种模型试图用不同的方法来表达多级安全策略。

BLP 模型是一个形式化的模型，它使用数学语言对系统的安全性质进行描述。BLP 模型也是一个状态机模型。它形式化地定义了系统、系统状态和状态间的转换规则，定义了安全概念，并制定了一组安全特性，对系统状态和状态转换规则进行约束，使得对于一个系统，如果它的初始状态是安全的，并且经过的一系列规则都是保持安全的，那么可以证明该系统是安全的。在这里所谓“安全”，指的是不产生信息的非法泄漏，即不会产生信息由高安全级的实体流向低安全级的实体。

2、安全信息流的格模型

安全信息流的格模型是 1976 年 D. Denning 提出来的。这一模型所反映的安全需求从实质上来说与 BLP 模型是一致的。即主体不能把高安全级的客体中的数据传送到低安全级的客体。但这一模型又是对 BLP 模型的扩充，它不仅禁止用户直接读取超过他安全级的客体，而且还禁止他伙同有权访问这些客体的用户，以某种巧妙的方式间接访问这些信息。

该模型定义了主、客体的安全类，在系统安全类集合上定义了信息流关系，并用格结构来描述安全信息流。在系统中任意操作序列的执行所产生的信息流动，只能沿着格结构所定义的流关系的方向进行流动。

3、无干扰模型

该模型是 Goguen 与 Meseguer 在 1982 年提出来的。模型用无干扰的概念来描述安全策略。所谓“无干扰”是指，设有使用某一命令集的一组用户和另一组用户，如果第一组用户使用这些命令所得到的结果，对于第二组用户能访问的数据没有影响，则称第一组用户没有干扰第二组用户。

4、Biba 模型

该模型是数据完整性保护模型，

是 Biba K. J. 1977 年提出来的。该模型认为数据客体以不同的完整性级别存在，系统应防止低完整性级别的主体破坏高完整性级别的数据的完整性。因此它规定，仅当主体的完整性级别支配客体的完整性级别时，主体对该客体有修改的权限。

5、Clark-Wilson 模型

Clark-Wilson 模型是 1987 年 Clark 和 Wilson 根据商业数据处理的实践经验提出来的。它是保护数据完整性的模型，是对商业安全策略的描述。因此该模型基于以下两个基本概念：

① 良性事务：用户不能随意操纵数据，只能以受控的方式操纵数据，以确保数据内部的一致性。

② 职责分散：所有的操作被划分成若干个子操作，各个子操作由不同的用户执行。

习题三

3.1 什么是自主访问控制？什么是强制访问控制？两者的根本区别在哪里？
3.2 什么是基于角色的访问控制？与传统的自主访问控制和强制访问控制相比，它的优势体现在哪些方面？
3.3 在基于角色的访问控制中，角色的含义是什么？它起什么作用？引入“角色”给系统的访问控制带来了什么便利？
3.4 安全策略在构建安全计算机信息系统中起何作用？
3.5 军事安全策略与商业安全策略有哪些区别？
3.6 安全模型在设计安全计算机信息系统中起何作用？

第 4 章　访问控制与安全模型

目前常用的也是基本的访问控制方法主要有自主访问控制，强制访问控制和基于角色的访问控制，相应的具有代表性的安全模型分别是访问矩阵模型，BLP 模型和 RBAC 模型。本章分别对它们加以介绍。

4.1 自主访问控制与访问矩阵模型

如第三章所述，自主访问控制是基于请求者的身份和访问规则来控制访问，访问规则规定请求者可以做什么，不可以做什么，之所以称为是自主的是因为一个实体可以被授权按其自己的意愿使另一个实体能够访问某些资源。

4.1.1 访问矩阵模型

访问矩阵的概念最早是 Lampson 1969 年提出，随后 Graham，Denning 和 Harrison 等人进一步进行了改进和细化。

1　系统状态

实施了自主访问控制的系统，其状态可以由一个三元组（S, O, A）来表示。其中，

S——主体的集合；

O——客体的集合，主体也被看作客体，因此 $S \subseteq O$；

A——访问矩阵。行对应于主体，列对应于客体。矩阵中第 i 行 j 列的元素 a_{ij} 是访问权的集合，列出了允许主体 s_i 对客体 o_j 可进行的访问权。为方便起见，元素 a_{ij} 常记做 $A[s_i, o_j]$。

例 4.1　图 4.1 给出了三个主体 s_1, s_2, s_3,两个存储器段 M_1,M_2 和两个文件 F_1,F_2 的访问矩阵的一个简单例子

客体

主体	M_1	M_2	F_1	F_2
s_1	Own、R、W、E		Own、R、W、D	
s_2	R	Own、R、W	W	R

S_3	R、W		R	Own、R、W、D

图 4.1　访问矩阵 A

在这里，主体集 S={S_1,S_2,S_3}，客体集 O={M_1,M_2,F_1,F_2}，A 就是图 4.1 给出的三行四列的矩阵。有序三元组（S,O,A）反映了该系统的一个状态。例如，在这一状态下，主体 S_1 拥有 M_1 和 F_1，并对 M_1 具有读写和执行的权限，对 F_1 具有读、写和删除的权限。而对 M_2 和 F_2，S_1 不具有任何访问权限。

当某一主体 s_i 要对客体 o_j 进行访问时，系统中的监控程序检查矩阵 A 中的元素 A[s_i, o_j]以决定 s_i 对 o_j 是否可以进行 s_i 所请求的访问。监控程序可以由硬件、软件或者硬件与软件共同组成。监控程序的一个例子是检验地址的硬件，它检查该地址是否处于与一进程相联系的存储段边界内。

我们称（S,O,A）为系统的保护状态。系统的保护状态给出了在一特定的时间点每个主体对每个客体的访问权的信息集。

2　系统状态的变化

系统的保护状态是变化的，其变化是由一些命令引起的，这些命令则由改变访问矩阵的一些基本操作的序列所组成，这些基本操作是：

enter　r　into　A[s, o]　　将访问权 r 添加到 A[s, o]中；

delete　r from　A[s, o]　　在 A[s, o]中删除访问权 r；

create　Subject　s'　　生成一个主体 s'；

create　Object　o'　　生成一个客体 o'；

destroy　Subject　s'　　删除主体 s'；

destroy　Object　o'　　删除客体 o'。

这里 r 表示某一种访问权。

表 4-1 形式地定义了这些基本操作对访问矩阵的影响，其中 op 代表基本操作，Q=（S, O, A）表示操作前的系统状态，在表中定义的条件下，执行 op 引起系统状态变化，由 Q 变成 Q'=（S', O', A'）。

表 4-1　基本操作

op	条件	新状态
enter r into A[s_i, o_j]	$s_i \in S$ $o_j \in O$	S'=S O'=O A'[s_i, o_j]= A[s_i, o_j]∪ {r} A'[s_k, o_l]= A[s_k, o_l]　当(s_k, o_l)≠(s_i, o_j)
delete r from A[s_i, o_j]	$s_i \in S$ $o_j \in O$	S'=S O'=O A'[s_i, o_j]= A[s_i, o_j]－{r} A'[s_k, o_l]= A[s_k, o_l]　当(s_k, o_l)≠(s_i, o_j)
create subject s'	$s' \notin S$	S'=S∪ {s'} O'=O∪ {s'} A'[s, o]= A[s, o]　当 s∈S, o∈O A'[s', o]=ϕ，当 o∈O' A'[s, s']=ϕ, s∈S'
create object o'	$o' \notin O$	S'=S O'=O∪ {o'} A'[s, o]= A[s, o]　当 s∈S, o∈O A'[s, o']=ϕ，当 s∈S'
destroy subject s'	$s' \in S$	S'=S－{s'} O'=O－{s'} A'[s, o]= A[s, o]　当 s∈S', o∈O'
destroy object o'	$o' \in O$ $o' \notin S$	S'=S, O'=O－{o'} A'[s, o]= A[s, o]　当 s∈S', o∈O'

一条命令可能由若干个基本操作构成，例如，任何进程都可创建一个新文件。此时，系统将自动分配给创建文件的进程对该文件的拥有权和读、写权，这可用如下命令来表示：

command create file (p, f)

```
create      object   f;
enter       Own      into     A[p, f];
enter       R        into     A[p, f];
enter       W        into     A[p, f];
end.
```

访问矩阵模型是在实际系统中实现保护策略和机制的抽象表示。因此，它提供了一个帮助理解和描述保护系统的辅助概念，一个便于比较不同保护系统的共同框架和一个研究保护系统固有特性的形式模型。然而，有些策略和机制用其它模型更容易描述，这在后面的讨论中我们将会看到。

4.1.2 访问矩阵的实现

在实施自主访问控制的系统中，访问控制矩阵提供的信息必须以某种形式保存在系统中，以便系统在主体发出访问客体的请求时，监控程序作相应的安全检查，并在主体进行自主授权或撤销授权时，动态地维护这些信息。然而在操作系统实现自主访问控制时，都不是将矩阵整个地保存起来，因为该矩阵可能是很大的稀疏矩阵，这样做效率很低，实际的作法是基于矩阵的行或列来表达访问控制信息。

1　基于授权表的自主访问控制

将访问矩阵按列分解，为每一个客体产生一个授权表，该表由被授权访问该客体的所有主体及这些主体对该客体所具有的访问权限组成。

其形式如表 4-2。表长 $n\geq 0$，其中 s_i 表示主体，z_i 表示 s_i 对 o 的访问权集合。因此客体 o 的授权表由访问控制矩阵中客体 o 所对应的列中所有非空项所组成。

表 4-2 客体的授权表

主体	权限
s_1	z_1
s_2	z_2
…	…
s_n	z_n

例 4.2　图 4.1 所给出的例子中，文件 F_1 和 F_2 的授权表分别如表 4-3 和 4-4

所示。

表 4-3　文件 F_1 的授权表

主体	权限
s_1	Own、R、W、D
s_2	W
s_3	R

表 4.4 文件 F_2 的授权表

主体	权限
s_2	R
s_3	Own 、R、W、D

在实际应用中，如果对某客体可以访问的主体很多，那么授权表会变得很长，占据较大的存储空间，并且在监控程序作判别时，也将花费较多的 CPU 时间。为此，可利用分组与通配符对授权表进行简化。我们知道，在一个实际的多用户系统中，用户往往可按其所属部门或工作性质进行分类，将属于同一部门或工作性质相同的人（例如，所有的外科医生，或所有的内科医生）归为一个组。一般来说，他们访问的客体以及访问客体的方式基本上是相同的。这时，我们为每个组分配一个组名，访问判决时可以按组名进行，在授权表中相应地设置一个通配符“*”，它可以替代任何组名或主体标识符。这时授权表中的主体用如下形式标识：

主体标识=ID.GN

其中，ID 为主体标识符，GN 表示该主体所属的组名。例如，F 的授权表如表 4-5 所示。

表 4-5　客体 F 的授权表

主体标识	权限
小张.Crypto	R、E、W
*.Crypto	R、E
小李.*	R
.	N

由授权表可以看出，属于 Crypto 组的所有主体(*.Crypto)对客体都具有读与运行权，Crypto 组的小张不仅对 F 具有读和运行权，还具有写权，无论哪个组的

小李对 F 都只可进行读访问。对于其它任何主体，无论属于哪个组（*.*），对该客体都不具有任何模式的访问权，通过这样简化，授权表可大大缩小。

在客体 o 的授权表中，o 的拥有者可以通过删除某个主体在授权表中的项目来撤销该主体对 o 的访问权限，或通过增加某个主体在授权表中的项目来授予该主体对 o 的访问权限。

因为每个授权表提供了一个指定资源的信息，所以当想要确定哪个主体对某个资源具有哪些访问权限时，使用授权表很方便。但是，对于要确定一个特定主体可以使用的所有访问权时，这种数据结构就不方便了。

2 基于能力表的自主访问控制

将访问矩阵按行分解，为每一个主体产生一个能力表，该表由该主体被授权访问的所有客体及该主体对这些客体所具有的访问权限组成。其形式如表 4.6 所示，表长 $n\geq 0$，其中，o_i 表示客体，z_i 表示该主体对 o_i 的访问权集合。因此，主体 s 的能力表由访问矩阵中主体 s 所对应的行中所有非空项组成。

表 4-6　主体 S 的能力表

客体	权限
O_1	Z_1
O_2	Z_2
…..	……
O_n	Zn

例 4.3　图 4.1 所给出的例子中，主体 s_1 和 s_2 的能力表分别如表 4.7 和 4.8 所示。

表 4-7 主体 s_1 的能力表

客体	权限
M_1	Own、R、W、E
F_1	Own、R、W、D

表 4-8　主体 s_2 的能力表

客体	权限
M_1	R

M_2	Own、R、W
F_1	W
F_2	R

根据每一主体 s 的能力表，可以决定 s 可否对给定客体访问以及可以进行什么样的访问。能力表在时间效率和空间效率上都优于访问矩阵，但能力表也有许多不方便之处，如对于一个给定的客体，要确定所有有权访问它的主体、用户生成一个新的客体并对其授权或删除一个客体时都显得较为麻烦。

能力表有一个比授权表更为严重的安全问题，即能力表可能被伪造，解决这个问题的一个办法是让操作系统管理这些能力表，将这些能力表保存在用户不能访问的一块内存区域。

针对单个客体的授权表和单个主体的能力表，Sandha 等人提出了一种数据结构，它不像访问矩阵那么稀疏，但比上述的授权表和能力表更为方便。我们称之为整体授权表。整体授权表中的一行对应于一个主体对一个客体的一种访问权，若按主体排序访问该表等价于能力表。若按客体排序访问该表则等价于授权表，关系数据库很容易实现这种类型的授权表。

例 4.4　图 4.1 所给出的例子中，整体授权表如表 4-9 所示

表 4-9 图 4.1 例中的整体授权表

主体	访问模式	客体	主体	访问模式	客体
S_1	Own	M_1	S_2	Write	M_2
S_1	Read	M_1	S_2	Write	F_1
S_1	Write	M_1	S_2	Read	F_2
S_1	Execute	M_1	S_3	Read	M_1
S_1	Own	F_1	S_3	Write	M_1
S_1	Read	F_1	S_3	Read	F_1
S_1	Write	F_1	S_3	Own	F_2
S_1	Delete	F_1	S_3	Read	F_2
S_2	Read	M_1	S_3	Write	F_2

S_2	Own	M_2	S_3	Delete	F_2
S_2	Read	M_2			

4.1.3 授权的管理

在前面我们曾提到过，主体对客体的权限中，有两种特殊的权限，即拥有权（Own）和控制权(Control)。若主体 s 创建了客体 o，则称 s 是 o 的拥有者，对 o 就具有了拥有权。在系统中，客体 o 的拥有者对 o 具有全部的访问权限，他对客体 o 的访问权仅能通过超级用户（如系统管理员或系统安全员）来改变。与此同时，他也具有了对客体 o 的控制权，即它可以将客体 o 的访问权（例如，读权、写权等），全部或部分地授予其它的主体，并且可以在以后的任何时刻撤销它所授予的权限。也就是说它具有修改该客体的授权表的能力。系统的其它主体，当它被授予了对客体 o 的访问权后，他是否可以将这一访问权再授予别的主体呢？这涉及到系统内的授予权或者说控制权的管理问题，对此，有以下三种主要的处理模式。

（1）集中型管理模式

在这种管理模式下，客体 o 的拥有者对 o 具有全部的控制权，其它的主体对 o 只可能有访问权而没有控制权。拥有者是唯一（超级用户除外）有权修改 o 的授权表的主体，亦即它是唯一能决定哪些主体对 o 具有访问权和具有什么样的访问权的主体。因此，系统中的其他主体若被授予对 o 的某种访问权，那么他仅对 o 具有了相应的访问权，但他无权将这一访问权再授予其他主体。在这种模式下客体的拥有者无权将对该客体的控制权授予其他的主体。

这种方法目前已经应用在许多系统中，但它有一定的限制，拥有者是唯一能够删除客体的主体，如果拥有者离开客体所属的组织或者意外死亡，那么系统必须设立某种特权机制，当意外情况发生时系统能够删除客体。Unix 系统是一个实施这种管理模式的例子，在 Unix 系统中，利用超级用户来实施特权控制。

集中型管理模式的另一个缺点是：非拥有者的主体想要修改对该客体的访问权较为困难，它必须请求该客体的拥有者为他改变相应客体的授权表。然而，从安全的角度看，这些缺点或许正是某些系统所希望具有的优点。

（2）分散型管理模式

在分散型管理模式下，客体的拥有者不但可以将对该客体的访问权授予其它主体，而且可以同时授予他们对该客体相应访问权的控制权（或相应访问权的授予权）。这样一来，对于一个客体 o，系统中哪些主体对 o 有访问权，有什么样的访问权，不但 o 的拥有者可以决定，其他被授予控制权的主体也可决定。因此，在系统中形成了一个允许传递授权的局面。在这种管理模式下，当一个主体撤销它所授予的对某个客体的某种访问权限时，必须将由于这一授权而引起的所有授权都予以撤销。

下面来看一个这种管理模式的例子。

例 4.4　在数据库中，对关系的访问权包括：

Read　读表中的行，利用关系查询，定义基于关系的视图；

Insert　在表中插入新的行；

Delete　删去表中的某行；

Update　修改表中某一列的数据；

Drop　删去某个表。

现假设用户 A 对某关系 X 所建立的一系列授权的结果如下，A 在时刻 t=10 将对 X 的 Read 和 Insert 权授予了 B，并同时授予了 B 对这两种访问权的授予权。B 在 t=20 将这两种访问权授给了 C，并允许 C 再授权。A 在 t=15 将对 X 的 Read 权授予了 D，但没有授予 D 对 Read 的控制权。D 在 t=30 从用户 C 处又得到对于 X 的 Read 和 Insert 两访问权及其授予权。

上述授权过程可用图 4.2 中的有向图表示。图中每一结点表示一个可访问这个关系的用户，每条有向边表示访问权的授予，边上标有被授予的权限和授予的时刻。r(y)表示授予权限 r 及对 r 的控制权，r(n)表示仅授予权限 r 但不授予对 r 的控制权。

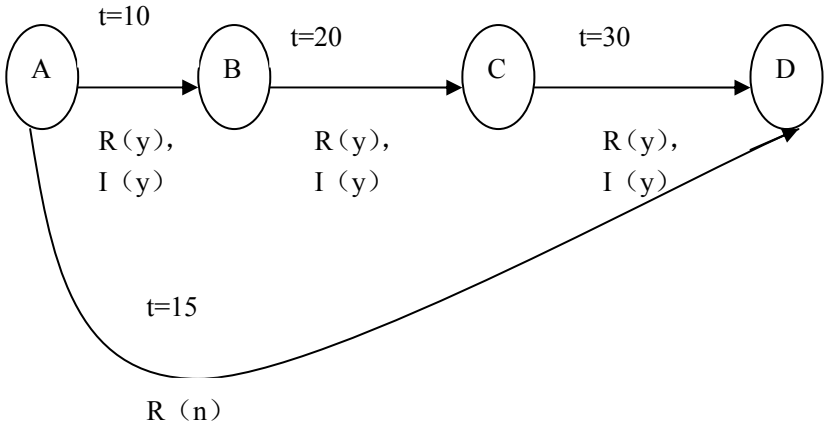


图 4.2 对关系 X 的访问权的转移

假定在时刻 t=40，用户 A 撤销了授予 B 的权限，则此时 B 授予 C 的权限及 C 授予 D 的权限也相应被撤销，虽然 D 保留了直接由 A 得到的读访问权，但 D 失去了对此访问权的授予权。

上述授权过程可以记录在一个访问表中，它的每一个元组说明了接受访问权的用户、用户被允许访问的表的名字、表的类型（视图或基表），访问权的授予者，授予的访问权及接受者对其是否具有控制权。每个访问权（除修改权外）可由表中的一列表示，且指出授予的时间（0 表示没被授予），撤销授权时利用时戳来判断访问权传播的路径。

修改哪一列可用 All（全部）、None（没有）或 Some(有些)，若说明为 Some，则对每一可修改的列，元组被放在另一个表中。

通过选择 Table=X 的所有元组，可以从上述访问表中得到一个特定关系 X 的授权表，例如图 4.2 所示的例中，关系 X 的授权如表 4-10 所示：

表 4-10 关系 X 的授权表

用户	表	授予者	读（Read）	插入（Insert）	控制
B	X	A	10	10	y
D	X	A	15	0	n
C	X	B	20	20	y
D	X	C	30	30	y

在 t=40 用户 A 撤销了对 B 的授权后，表 4-10 的状态变为：

用户	表	授予者	读（Read）	插入（Insert）	控制
D	X	A	15	0	n

分散型的管理模式使得系统中的主体对客体的访问权的控制有充分的自主权。一旦主体将对某客体的访问权及其控制权授予了某主体 B，那么 B 就可以将这种权利再分配给其它主体，而不必征得客体拥有者的同意，这对于那些对资源共享程度要求较高的系统来说，无疑大大减少了权限管理上的工作量，但对于要求充分保护拥有者的权利和保护系统资源的安全性来说又有其缺陷。因为一旦对客体的控制权分配出去，拥有者再要想控制客体就很困难了。为此，可以在分散型管理模式的基础上对授予权的传递进行一些限制。

（3）受限的分散型管理模式

受限的分散型管理模式，是将系统中对客体的访问权限制在一定的主体范围

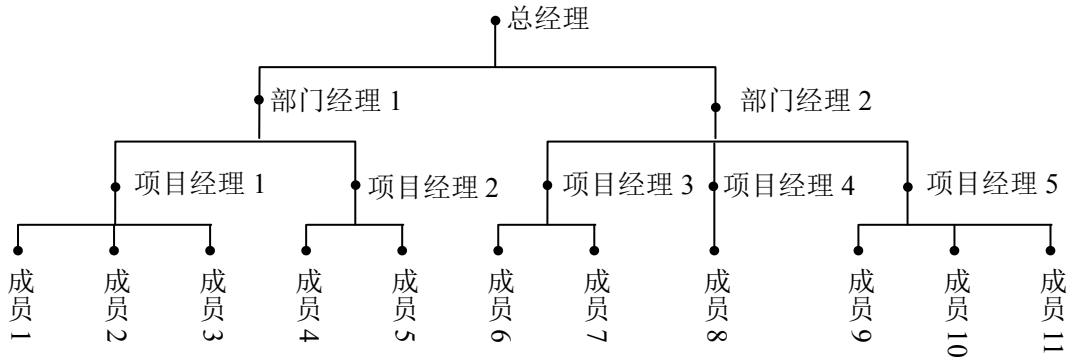


图 4.3　树型控制方式

内，限制主要根据客体拥有者的意愿进行。例如，客体 o 的拥有者可对系统内不允许对 o 进行写访问的主体发放黑令牌以阻止其它有权对 o 进行写访问的主体将这一访问权授予这些主体。又例如我们可以根据系统应用对象的需求，将授权路径限制在某一组织结构内。

假设图 4.3 表示某一公司的组织结构。顶级结点总经理在权限的管理上可看作是系统管理员或系统安全员或者系统中所有资源的拥有者，他对资源的访问权具有绝对的控制权。为了将控制权进行分散管理，而又不至于滥用，他可将其控制权分别授予两个部门经理，两个部门经理又可在各自管辖的范围内对其项目经理进行授予……在这种等级结构中，最低层的主体对任何客体都只可能具有访问权，而不具有对访问权的授予权。中间层的主体对客体可能既具有访问权，又具

有对这一访问权的授予权，但他的授权对象必须是他的下属。这种管理方式的授权路径较为固定，出现安全问题时，便于追查。

4.2 强制访问控制与 BLP 模型

如第三章所述，强制访问控制是系统基于自身的安全需求制定出相应的安全策略，并根据用户在系统中的地位和职责对其访问权进行的一种控制方法。之所以称它是强制性的，是因为在这种访问控制方法中，允许哪些主体对一个资源进行什么样的访问，不是按照资源拥有者或其他主体的意愿，而是按照系统事先制定的规则来决定的。

强制访问控制最早应用于军事部门，并通过多级安全的概念来描述其控制，BLP 模型是最著名的多级安全模型，所以人们常把多级安全的概念与 BLP 模型联系在一起。

强制访问控制可以基于不同的安全目标，例如可以通过强制访问控制实现信息机密性的保护，也可以通过强制访问控制实现信息完整性的保护。BLP 模型是实现信息机密性保护最具代表性的一个安全模型。Biba 模型则是一个信息完整性保护的模型。

本节将以较为简洁的方式介绍 BLP 模型最基本和最核心的内容，以使读者通过他了解一个系统的安全需求是如何体现在该系统的安全策略中，而相应的安全模型又如何以准确，严谨，且又便于系统设计与开发人员理解和实现的方式描述出来。

4.2.1 Bell-La Padula 模型

BLP 模型是著名的描述军事安全策略的形式化模型，已为许多操作系统所使用，BLP 模型也是一个状态机模型，它用状态变量表示系统的安全状态，用状态转换规则描述系统状态的变化。用一组安全特性对系统状态和状态转换规则进行约束，使得系统始终保持其安全性，在这里，安全指的是信息的机密性得到保护，不会产生非法泄露。

1、模型的基本元素

BLP 模型的基本元素是用来描述使用多级安全策略的信息系统中的实体以及这些实体的安全标记（即安全级）、他们的各种可能的访问请求、以及系统对

用户请求的各种可能的判定等。因为使用的都是数学符号，看起来比较抽象，实际上这些符号的意义是很直观的。

先看一个例子，某电视台拟组织一场才艺竞赛，竞赛之前经报名和条件审核，可确定参赛者名单，如张亮，王丽，….等，组织方需对每个参赛者分配一个编号，如张亮为 1 号，王丽为 2 号….用以确定竞赛时的出场次序，另外，组织方要公布允许的参赛项目，如，唱歌，跳舞，乐器独奏，单口相声….等，并根据参赛者的个人背景，对参赛项目的数量和种类的选择作出某些限制，组织方可以提供他们所要使用的道具，如小提琴，钢琴，演出服装，麦克风，伴舞者等，组织方还需要请若干评委对参赛者的表演进行评判，其评判的结果，例如可以采取百分制进行评分。一切安排就绪后，竞赛就可以开始了，我们可以将参赛者、参赛者的编号、竞赛的项目、道具以及评委评分采用的方式看做是组成这场竞赛的一些基本要件。BLP 模型所要描述的具有多级安全的信息系统，与此例有一定的类似之处。即在描述系统如何运行之前要描述组成这一系统的基本要件，我们称它为基本元素。

在介绍 BLP 模型的基本元素之前，请特别注意如下两个概念及其表示方法。

（1）集合的笛卡尔积

设 $A_1, A_2, \ldots, A_n$ 是 n 个集合，集合

$A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) | a_i \in A_i, i=1, 2, \ldots, n\}$

称为集合 $A_1, A_2, \ldots, A_n$ 的笛卡尔积。

这一笛卡尔积的元素是由 n 个个体组成的有序 n 元组$(a_1, a_2, \ldots, a_n)$，这 n 个个体按照符号 $A_1 \times A_2 \times \ldots \times A_n$ 给定的次序，分别取自于集合 $A_1, A_2, \ldots, A_n$。如果 $A_1, A_2, \ldots, A_n$ 均是有限集，则笛卡尔积 $A_1 \times A_2 \times \ldots \times A_n$ 也是有限集且其元素个数 $\#(A_1 \times A_2 \times \ldots \times A_n) = \#A_1 \times \#A_2 \times \ldots \times \#A_n$。

（2）集合的幂集

设有集合 A，A 的幂集用 P_A 或 2^A 表示，定义为：

$P_A = \{H | H \subseteq A\}$

即 A 的幂集 P_A 是由 A 的所有子集构成的集合。

关于这两个概念的详细叙述，请参看文献[洪帆，付小清. 离散数学[M]，武汉：华中科技大学出版社，2000]。在 BLP 模型的描述中，经常使用这两个概

念。

BLP 模型定义了如下集合：

① $S=\{s_1,s_2,\ldots, s_n\}$ 主体的集合，主体指用户、进程；

② $O=\{o_1,o_2,\ldots, o_m\}$ 客体的集合，客体指文件、数据、程序、存储器段等，主体也可看作客体；

主体 s_i 是对系统进行操作或者说要访问系统资源的一些实体，他们类似于上例中的参赛者，客体 o_j 是系统中的各种资源，可供主体 S_i 使用，他们类似于上例中的道具。

③ $C=\{c_1,c_2,\ldots, c_q\}$ 主体或客体的密级,$c_1<c_2<\ldots<c_q$，元素之间呈全序（线性序关系）；

④ $K=\{k_1,k_2,\ldots, k_r\}$ 范畴集，可看作某个组织中部门的集合或类别的集合；

正如 3.3.1 节中军事安全策略所介绍的一样，系统为每个主体和客体都分配一个安全级，安全级由密级和范畴集两部分组成。表示为二元组（密级，范畴集）。每一个密级都取自集合 C 中的一个元素 c_i ，3.3.1 节介绍的系统仅有 4 个密级，这里采用更一般的描述，系统可以扩展为任意多个密级。每一个范畴集都由集合 K 的一个子集 H_j 表示，因此每一个实体的安全级由（c_i，H_j）表示。集合 C 和集合 K 分别限定了密级和范畴集的取值范围。安全级类似于上例中参赛者的编号，根据参赛人数，参赛者的编号限定在一定的范围内。这里安全级的取值范围不是根据主客体的数目，而是根据系统对信息保密性的需求及系统服务对象的组织结构来决定。

⑤ A={r, w, e, a, c} 访问属性集，其中，read（只读），write（读/写），execute（执行），append（添加），control（控制）；

⑥ RA={g, r, c, d} 请求元素集，其中

g 表示 get 或 give；

r 表示 release 或 rescind；

c 表示 change 或 create；

d 表示 delete。

⑦ D={yes, no, error,?} 结果集（判定集），其中

yes：表示请求被执行；

no： 表示请求被拒绝；

error：表示系统出错，有多个规则适合于这一请求；

?： 表示请求出错，规则不适用于这一请求。

访问属性集 A 中各元素的含义与 3.1 节介绍的基本相同，实际上，读者开发应用系统时，根据需要可自行定义各种访问属性。

集合 RA 中各元素描述了用户对系统可能发出的各种类型的操作请求，其中，g 表示 get 或 give，get 表示主体要求得到对某个客体的某种访问权，give 表示主体要求授予另一个主体对某客体的某种访问权，即自主访问控制中的授权。完整的请求由一个包含有 g 的五元组表示，配合五元组中的其他元素可判断出 g 是代表 get 还是 give。以下情形类似。

Release 表示主体请求释放对某客体的访问权。rescind 表示主体撤销另一主体对某客体的访问权，此即自主访问控制中的权限回收。

Change 用于改变处于静止状态的客体的安全级，什么是静止客体以及如何使用该命令在后面再具体介绍。Create 表示主体要求创建一个客体。

Delete 表示主体要求删除某个客体，当然，按照前面所介绍的安全规则，主体必须是该客体的拥有者或者对该客体具有控制权，系统才允许该请求。

判定集 D 中各元素表示系统根据预先制定的安全规则，对用户的操作请求是否允许所做出的回应。其中 yes 表示用户请求符合安全规则，系统允许其执行。No 表示用户请求违反安全规则，其请求被拒绝。error 表示系统出错。？表示请求出错，即请求者未按规定的格式提出请求。集合 D 中的元素类似于上例中评委的评分结果。

⑧ $\mu=\{M_1, M_2, \ldots, M_p\}$ 访问矩阵的集合，μ 中的元素记作 M_k，它是 4.1.1 节中所介绍的用来描述自主访问控制授权的访问矩阵。任何一个时刻系统中自主授权的状态在 μ 中都有一个访问矩阵与其对应。M_k 是一个 $n\times m$ 的矩阵，M_k 中的第 i 行，j 列的元素在 BLP 模型中将其记做 M_{ij}。因此 M_{ij} 是 A 的子集。可以为空。

⑨ $F=C^S\times C^O\times(P_K)^S\times(P_K)^O$ 是四个集合的笛卡尔积，其中，

$C^S=\{f_1|f_1：S\rightarrow C\}$　　元素 f_1 定义系统中每一个主体的密级；集合 C^S 给出了对主体密级所有可能的定义。

$C^O=\{f_2|f_2：O\rightarrow C\}$　　元素 f_2 定义系统中每一个客体的密级；集合 C^O 给出了对客体密级所有可能的定义。

$(P_K)^S=\{f_3|f_3：S\rightarrow P_K\}$　元素 f_3 定义系统中每一个主体的范畴（或部门）集；

$(P_K)^O=\{f_4|f_4：O\rightarrow P_K\}$　元素 f_4 定义系统中每一个客体的范畴（或部门）集。

集合$(Pk)^S$和$(Pk)^O$的含义分别与 C^S 和 C^O 类似。

取定 F 中的一个元素 $f=(f_1, f_2, f_3, f_4)$，相当于对系统中的所有主体和客体均分配了密级和部门集。例如$(f_1(s_i), f_3(s_i))$，定义了主体 s_i 的安全级，$(f_2(o_j), f_4(o_j))$定义了客体 o_j 的安全级。集合 F 中的所有元素则给出了对主体和客体的安全级的所有可能的定义方式。

2、系统状态

BLP 模型用 $V=P_{(S\times O\times A)}\times\mu\times F$ 表示系统所有可能的状态，V 中的元素 v=(b, M, f)表示系统的某个状态。其中，$b\in P_{(S\times O\times A)}$或 $b\subseteq S\times O\times A$，是当前访问集，表示哪些主体取得了对哪些客体的什么样的访问权限。

例如，若 $b=\{(s_1, o_2, r), (s_1, o_3, w), (s_2, o_2, e), \ldots\}$，则表示在状态 v 下，主体 s_1 对客体 o_2 有“读”访问权，主体 s_1 对客体 o_3 有“写”访问权，主体 s_2 对客体 o_2 有执行的访问权等等。

$M\in\mu$ 是访问矩阵，它表示在状态 v 下，主体自主授权的状态，它的第 i 行，j 列的元素表示主体 s_i 被自主地授予了对客体 o_j 哪些访问权限。

$f\in F$，$f=(f_1, f_2, f_3, f_4)$，其中，f_1 和 f_3 给出了所有主体的安全级，f_2 和 f_4 给出了所有客体的安全级。

系统在任何一个时刻都处于某一种状态 v，即对任意时刻 t，必有状态 v_t 与之对应。随着用户对系统的操作，系统的状态会不断地发生变化。对应状态 v，集合 b 中的那些主体对客体的访问权限是否会带来对信息的泄漏呢？也就是说，我们必须关心系统在各个时刻的状态，特别是与状态相对应的访问集 b 是否能保证系统的安全性。只有每一个时刻状态是安全的，系统才可能是安全的。为此，BLP 模型对状态的安全性进行了定义。

3、安全特性

BLP 模型的安全特性定义了系统状态的安全性，也集中体现了 BLP 模型的安全策略。BLP 模型要求系统既具有自主访问控制又具有强制访问控制，其安全性要求如下：

（1）自主安全性

状态 v=(b, M, f)满足自主安全性当且仅当对所有的$(s_i, o_j, x)\in b$，有 $x\in M_{ij}$。

这条性质是说，若$(s_i, o_j, x)\in b$，即如果在状态 v，主体 s_i 获得了对客体 o_j 的 x 访问权，那么 s_i 必定是得到了相应的自主授权。若存在$(s_i, o_j, x)\in b$，但主体 s_i 并未得到对客体 o_j 的 x 访问权的授权（即 $x\notin M_{ij}$），则状态 v 被认为不符合自主安全性。

（2）简单安全性

状态 v=(b, M, f)满足简单安全性当且仅当对所有的$(s, o, x)\in b$， 有

（1）x=e 或 x=a 或 x=c；或

(2) (x=r 或 x=w)且$(f_1(s)\geq f_2(o), f_3(s)\supseteq f_4(o))$。

这条性质是说，若在 b 中主体 s 获得了对客体 o 的“r” 或“w”权，则 s 的密级必须不低于 o 的密级，s 的范畴集必须包含 o 的范畴集。也就是，s 的安全级必须支配 o 的安全级，这条性质的意义在于低安全级的主体不允许获得高安全级客体的信息。

注意，在 BLP 模型中，w 权表示可读、可写，即主体对客体的修改权。

（3）∗−性质

状态 v=(b, M, f)满足∗−性质当且仅当对所有的 $s\in S$，若

$o_1\in b(s: w, a)$，$o_2\in b(s: r, w)$，则 $f_2(o_1)\geq f_2(o_2)$，$f_4(o_1)\supseteq f_4(o_2)$。

其中符号 $b(s: x_1, \ldots, x_n)$表示 b 中主体 s 对其具有访问权限 $x_i(1\leq i\leq n)$的所有客体的集合。

这条性质是 BLP 模型中最重要的一条安全特性。$o_1\in b(s: w, a)$，意味着 s 对 o_1 有 w 权或 a 权，此时信息可能由 s 流向 o_1；$o_2\in b(s: r, w)$意味着 s 对 o_2 有 r 权或 w 权，此时信息可能由 o_2 流向 s。他们组成的四种情形如图 4.4 所示，其中箭头表示信息的流向。这样一来，在访问集 b 中以 s 为媒介，信息就有可能由 o_2

流向 o_1，因此要求 o_1 的安全级必须支配 o_2 的安全级，当 s 对 o_1 和 o_2 均具有 w 权时，两次运用该特性，则要求 o_1 的安全级等于 o_2 的安全级。这反映 BLP 模型中信息只能由低安全级向高安全级流动的安全策略。

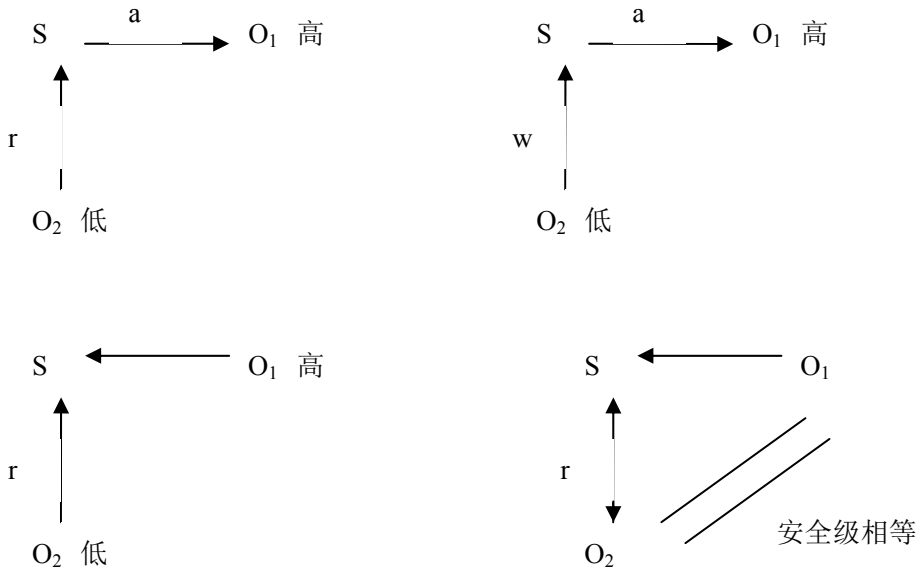


图 4.4　∗−性质解析

例如 设状态 v=（b，M，f），其中

b={(s_1,o_1,w), (s_1,o_2,r),(s_2,o_3,a),(s_2,o_2,w)}

f=(f_1,f_2,f_3,f_4)

则此时通过 s_1 信息有可能由 o_2 流向 o_1，通过 s_2 信息有可能由 o_2 流向 o_3，按照∗−性质的要求必须有

$f_2(o_1)\geq f_2(o_2)$,$f_4(o_1)\supseteq f_4(o_2)$,$f_2(o_3)\geq f_2(o_2)$,$f_4(o_3)\supseteq f_4(o_2)$-

一个状态 v 如果同时满足上述三条性质，那么 v 是安全状态。

4、主体的操作请求

主体对系统可能会有各种操作请求，如，要求对某客体进行某种访问，又如，要求授权给另一主体对某客体的某种访问权等等。BLP 模型统一用一个五元组来描述主体的各种操作请求，其操作请求涉及的对象和请求的类型由五元组中的元素表示。

令　$R=S^+\times RA\times S^+\times O\times X$ ，

R 的元素代表主体对系统的一个完整的请求，用五元组(σ_1, γ, σ_2, o_j, x)表示。因此 R 是主体的请求集。

其中 S^+=S U {ϕ}，X=A∪{ϕ}∪F。

五元组中，σ_1和σ_2均代表主体，可以为空，用ϕ表示。γ 代表请求的类型，在 RA 中取值。例如，若 γ=g，则表示某主体请求得到对某客体的某种访问权（此时 g 相当于 get）；也可能表示某主体请求授予另一主体对某客体的某种访问权（此时 g 相当于 give），至于 g 是代表前者还是后者，由五元组中是出现一个主体还是出现两个主体可以区分开来。例如，若 $R_k \in R$，$R_k=(\phi, g, s_i, o_j, r)$，则表示主体 s_i 请求得到对客体 o_j 读访问权，此时 g 代表 get，若 $R_l \in R$，$R_l=(s_\lambda, g, s_i, o_j, w)$，则表示主体 s_λ 请求授予主体 s_i 对客体 o_j 写访问权，此时 g 代表 give。o_j 代表某一客体。x 可能是某个访问权限；也可能是 f∈F，定义系统中各主、客体的安全级。x 还可能为空，用ϕ表示，x 的取值随请求的不同而不同。

5、状态转换规则

当用户发出操作请求时，系统必须根据前面定义的三条安全特性对用户的请求进行进行检查，符合安全特性的操作请求被认为是安全的，并被允许，否则拒绝执行。系统的状态随着用户的操作不断的转变。

状态转换规则用来描述系统对用户操作请求的判定和处理过程，这里面最重要的是系统对各种状态下的各类操作请求，应如何进行安全性检查，BLP 模型定义了 10 条基本规则，这些规则可以用函数 ρ :R×V—>D×V 表示。它表示对任意请求 $R_k \in R$ 和任意状态 v∈V，必有判定 $D_l \in D$，状态 $v^* \in V$，使 ρ $(R_k, v) \in (D_l, v^*)$。这意味着，在状态 v 下，当主体发出请求 R_k 时，系统会相应产生一个判定 D_l，并且可能发生状态转换，系统由 v 转换为 v^*。

下面介绍 BLP 模型的 10 条基本规则。这 10 条状态转换规则，清楚、具体地描述了 BLP 模型是如何进行安全控制的，以及它的安全目标和安全策略是如何实现的。

规则 1 用于主体 s_i 请求得到对客体 o_j 的 read 访问权。其请求的五元组 $R_k=(\phi, g, s_i, o_j, r)$，系统的当前状态 v=(b, M, f)。

Rule 1：get-read　$\rho_1(R_k, v) \equiv$

if　$\sigma_1 \neq \phi$　or $\gamma \neq g$ or $x \neq r$ or $\sigma_2 = \phi$　then

　　$\rho_1(R_k, v)=(?, v)$

if　$r \notin M_{ij}$　or　$[(f_1(s_i) < f_2(o_j)$　or　$f_3(s_i) \not\supseteq f_4(o_j))]$　then

　　$\rho_1(R_k, v)=(no, v)$

if　$U_{p_i}=\{o \mid o \in b(s_i: w, a)$ and $[f_2(o_j) > f_2(o)$ or $f_4(o_j) \not\subseteq f_4(o)]\} = \phi$　then

$\rho_1(R_k, v)=(yes, (b \cup \{(s_i, o_j, r)\}, M, f))$

else

$\rho_1(R_k, v)=(no, v)$

end

规则 1 对主体 si 的请求作了如下检查：

（1）主体的请求是否适用于规则 1；

（2）oj 的拥有者（或控制者）是否授予了 si 对 oj 的访问权；

（3）si 的安全级是否支配 oj 的安全级；

（4）在访问集 b 中，若 si 对另一客体 o 有 w 访问权或 a 访问权，是否一定有 o 的安全级支配 oj 的安全级。

若上述检查有一项不通过，则系统拒绝执行 s_i 的请求，系统状态保持不变。若请求全部通过了上述检查，则 s_i 的请求被执行，三元组(s_i, o_j, r)进入系统的访问集 b，亦即允许 s_i 对 o_j 进行读访问。系统状态由 v 转换成 $v^*=(b \cup \{(s_i, o_j, r)\}, M, f)$。

显然，检查项目（2）是系统在实施自主访问控制，项目（3）和项目（4）是系统在实施强制访问控制，只有通过了所有这些检查，才能保证系统在进行状态转换时，其安全性仍然得到保持。

规则 2 用于主体 s_i 请求得到对客体 o_j 的 append 访问权。其请求的五元组 $R_k=(\phi, g, s_i, o_j, a)$，系统的当前状态 v=(b, M, f)。

Rule 2：get-append：$\rho_2(R_k, v) \equiv$

if $\sigma_1 \neq \phi$ or $\gamma \neq g$ or $x \neq a$ or $\sigma_2=\phi$ then

$\rho_2(R_k, v)=(?,v)$

if $a \notin M_{ij}$ then

$\rho_2(R_k, v)=(no, v)$

if $U_{P_2}=\{o \mid o \in b(s_i: r, w) \text{ and } [f_2(o_j)<f_2(o) \text{ or } f_4(o_j) \not\supseteq f_4(o)]\}=\phi$ then

$\rho_2(R_k, v)=(yes, (b \cup \{(s_i, o_j, a)\}, M, f))$

else

$\rho_2(R_k, v)=(no, v)$

end

规则 2 对主体 s_i 的请求所作的检查类似于规则 1，不同的是当 s_i 请求以 append 方式访问 o_j 时，无需作简单安全性检查。

规则 3 用于主体 s_i 请求得到对客体 o_j 的 execute 访问权。其请求的五元组 $R_k=(\phi, g, s_i, o_j, e)$，系统的当前状态 v=(b, M, f)。

Rule 3：get-execute：$\rho_3(R_k, v) \equiv$

if $\sigma_1 \neq \phi$ or $\gamma \neq g$ or $x \neq e$ or $\sigma_2=\phi$ then

$\rho_3(R_k, v)=(?,v)$

if $e \notin M_{ij}$ then

$\rho_3(R_k, v)=(no, v)$

else

$\rho_3(R_k, v)=(yes, (b \cup \{(s_i, o_j, e)\}, M, f))$

end

规则 3 对 s_i 的请求只作类似于规则 1 中的（1）、（2）两项检查，因此 s_i 请求得到对 o_j 的执行权时不需求作简单安全性和*—性质的检查。

规则 4 用于主体 s_i 请求得到对客体 o_j 的 write 访问权。其请求的五元组 $R_k=(\phi, g, s_i, o_j, w)$，系统的当前状态 v=(b, M, f)。

Rule 4：get-write： $\rho_4(R_k, v)\equiv$

if $\sigma_1 \neq \phi$ or $\gamma \neq g$ or $x \neq w$ or $\sigma_2=\phi$ then

$\rho_4(R_k, v)=(?,v)$

if $w \notin M_{ij}$ or $[f_1(s_i)<f_2(o_j)$ or $f_3(s_i) \not\supseteq f_4(o_j)]$ then

$\rho_4(R_k, v)=(no, v)$

if $U_{P_4}=\{o|\ o\in b(s_i: r)$ and $[f_2(oj)<f2(o)$ or $f_4(oj) \not\supseteq f_4(o)]\}$

$\cup \{o|\ o\in b(s_i: a)$ and $[f_2(oj)>f_2(o)$ or $f_4(oj) \not\subseteq f_4(o)]\}$

$\cup \{o|\ o\in b(s_i: w)$ and $[f_2(oj)\neq f_2(o)$ or $f_4(oj)\neq f_4(o)]\}=\phi$ then

$\rho_4(R_k, v)=(yes, (b\cup\{(s_i, o_j, w)\},M, f))$

else

$\rho_4(R_k, v)=(no, v)$

end

规则 4 的安全性检查类似于规则 1，也作以下四项检查：

（1）s_i 的请求是否适用于规则 4；

（2）s_i 是否被自主地授予了对 o_j 的 w 权；

（3）s_i 的安全级是否支配 o_j 的安全级；

（4）*—性质的检查。

但规则 4 的*—性质检查较为复杂，它要求以下三个条件均必须成立：

（1）在 b 中，若 s_i 已对某一客体 o 有读权，则 o_j 的安全级必须支配 o 的安全级；

（2）在 b 中，若 s_i 已对某一客体 o 有添加权，则 o_j 的安全级必须受 o 的安全级支配；

（3）在 b 中，若 s_i 已对某一客体 o 有写权，则 o_j 的安全级必须等于 o 的安全级。

只要其中有一条不成立，则认为*—性质不成立，拒绝执行用户请求。

规则 5 用于主体 s_i 请求释放对客体 o_j 的访问权，包括 r、w、e 和 a 等权。其请求的五元组 $R_k=(\phi, r, s_i, o_j, r)$或$(\phi, r, s_i, o_j, w)$ 或(ϕ, r, s_i, o_j, a) 或(ϕ, r, s_i, o_j, e)，系统的当前状态 v=(b, M, f)。

Rule 5：release-read/write/append/execute： $\rho_5(Rk, v)\equiv$

if $(\sigma 1\neq \phi)$ or $(\gamma \neq r)$ or $(x \neq r, w, a$ and $e)$ or $(\sigma_2=\phi)$ then

$\rho_5(R_k, v)=(?, v)$

else

$\rho_5(R_k, v)=(yes, (b-\{(s_i, o_j, x)\},M, f))$

end

因为访问权的释放不会对系统造成安全威胁，所以不需要作安全性检查，并

可将四种情形用同一条规则来处理。

规则 6 用于主体 s_λ 请求授予主体 s_i 对客体 o_j 的某种访问权。其请求的五元组 $R_k=(s_\lambda, g, s_i, o_j, r)$或$(s_\lambda, g, s_i, o_j, w)$ 或$(s_\lambda, g, s_i, o_j, a)$ 或$(s_\lambda, g, s_i, o_j, e)$，系统的当前状态 v=(b, M, f)。

Rule 6：give-read/write/append/execute $\rho_6(R_k, v)\equiv$

if $(\sigma_1=\phi)$ or $(\gamma\neq g)$ or $(x\neq r, w, a \text{ and } e)$ or $(\sigma_2=\phi)$ then

$\rho_6(R_k, v)=(?,v)$

if $x\notin M_{\lambda j}$ or $c\notin M_{\lambda j}$ then

$\rho_6(R_k, v)=(no, v)$

else

$\rho_6(R_k, v)=(yes, (b, M\oplus[x]_{ij}, f))$

end

规则 6 中 $M\oplus[x]_{ij}$ 表示将 x 加入到访问矩阵 M 的第 i 行第 j 列元素 M_{ij} 中去。即用集合 $M_{ij}\cup\{x\}$替换 M 中 M_{ij}。

由于 s_λ 的请求只涉及自主访问控制中的授权，因此规则 6 除了作请求是否适用于规则 6 的检查外，仅作自主安全性有关的检查。即 s_λ 自身必须同时具有对客体 o_j 的 x 权和控制（c）权，方能对 s_i 进行相应的授权。

要注意的是，这一授权的成功，并不意味着 s_i 已获得对 o_j 的 x 访问权。因为 s_i 还未经过简单安全性和*—性质的检查。规则 6 的处理结果，是仅修改访问控制矩阵 M，使 M_{ij} 项元素也包含进权限 x。此时，访问集 b 中的三元组并无变化。

规则 7 用于主体 s_λ 撤销主体 s_i 对客体 o_j 的某种访问权。其请求的五元组 $R_k=(s_\lambda, r, s_i, o_j, r)$或$(s_\lambda, r, s_i, o_j, w)$ 或$(s_\lambda, r, s_i, o_j, a)$ 或$(s_\lambda, r, s_i, o_j, e)$，系统的当前状态 v=(b, M, f)。

Rule 7：rescind-read/write/append/execute：$\rho_7(R_k, v)\equiv$

if $(\sigma_1=\phi)$ or $(\gamma\neq r)$ or $(x\neq r, w, a \text{ and } e)$ or $(\sigma_2=\phi)$ then

$\rho_7(R_k, v)=(?,v)$

if $x\notin M_{\lambda j}$ or $c\notin M_{\lambda j}$ then

$\rho_7(R_k, v)=(no, v)$

else

$\rho_7(R_k, v)=(yes, b-\{(s_i, o_j, x)\}, M\ominus[x]_{ij}, f))$

end

规则 7 中，$M\ominus[x]_{ij}$ 表示将 x 从访问矩阵 M 的第 i 行第 j 列元素 M_{ij} 中去掉。即用集合 $M_{ij}-\{x\}$替换 M 中 M_{ij}。

类似于规则 6，规则 7 也只涉及自主安全性。系统要求 s_λ 必须对 o_j 同时具有 x 权和 c 权，方能对 s_i 的 x 权予以撤销。

系统执行这一请求时，不仅从访问矩阵 M 的 i 行 j 列的元素 M_{ij} 中将 x 删除掉，而且访问集 b 中也必须删去三元组(s_i, o_j, x)，这意味着主体 s_i 将丧失对 o_j 的 x 访问权，尽管它可能符合简单安全性和*—性质。

规则 8 用于改变静止客体的密级和范畴集。其请求的五元组 $R_k=(\phi, c, \phi, o_j, f^*)$，其中 c 表示 change，系统的当前状态 v=(b, M, f)。

Rule 8：change-f：$\rho_8(R_k, v)\equiv$

if $(\sigma_1\neq\phi)$ or $(\gamma\neq c)$ or $(\sigma_2\neq\phi)$ or $(x\notin F)$ then

$\rho_8(R_k, v)=(?,v)$

if $f_1^* \neq f_1$ or $f_3^* \neq f_3$ or $[f_2^*(o_j) \neq f_2(o_j)$ or $f_4^*(o_j) \neq f_4(o_j)$ for some $j \in A(m)]$ then

$\rho_8(R_k, v)=(no, v)$

else

$\rho_8(R_k, v)=(yes,(b, M, f^*))$

end

所谓静止客体是指被删除了的客体，该客体名可以被系统中的主体重新使用，例如某存储器段或某个文件名。当该客体被重新使用来存放数据时，客体的安全级定义为创建这一客体的主体的安全级。显然主体可以创建客体，但该客体的安全级必须由系统来定义，因此规则 8 中没有主体。

A(m)是活动客体的下标集，即 $A(m)=\{j|\ 1 \leqslant j \leqslant m$ 并且存在 i，使 $M_{ij} \neq \phi\}$。

规则 8 的安全性要求是，新定义的安全级 $f^*=(f_1^*, f_2^*, f_3^*, f_4^*)$，不能改变系统中主体的安全级，也不能改变活动客体的安全级，只能改变静止客体的安全级，在这种情形下，新状态 v^*用 f^*代替原状态 v 中的 f。

规则 9 用于主体 s_i 创建一个客体 o_j。其请求的五元组 $R_k=(\phi, c, s_i, o_j, e)$ 或 $R_k=(\phi, c, s_i, o_j, \phi)$，系统的当前状态 v=(b, M, f)。

Rule 9：create-object：$\rho_9(R_k, v) \equiv$

if $\sigma_1 \neq \phi$ or $\gamma \neq c$ or $\sigma_2=\varphi$ or $(x \neq e$ and $\phi)$ then

$\rho_9(R_k, v)=(?,v)$

if $j \in A(m)$ then

$\rho_9(R_k, v)=(no, v)$

if $x=\phi$ then

$\rho_9(R_k, v)=(yes,(b, M \oplus [r, w, a, c]_{ij}, f))$

else

$\rho_9(R_k, v)=(yes,(b, M \oplus [r, w, a, c, e]_{ij}, f))$

end

规则 9 要求主体 s_i 所创建的客体不能是活动着的客体。

当客体创建成功后，系统便将对 o_j 的所有访问权赋予 s_i，需要区分的是，当 o_j 不是可执行程序时，e 权不赋予 s_i。这里只涉及自主访问控制中的授权。

规则 10 用于主体 s_i 删除客体 o_j。其请求的五元组 $R_k=(\phi, d, s_i, o_j, \phi)$，系统的当前状态 v=(b, M, f)。

Rule 10：delete-object：$\rho_{10}(R_k, v) \equiv$

if $\sigma_1 \neq \phi$ or $\gamma \neq d$ or $\sigma_2=\phi$ or $x \neq \phi$ then

$\rho_{10}(R_k, v)=(?,v)$

if $c \notin M_{ij}$ then

$\rho_{10}(R_k, v)=(no, v)$

else

$\rho_{10}(R_k, v)=(yes, b, \ominus M \quad [r, w, a, c, e]_{ij, 1 \leqslant i \leqslant n}, f))$

end

从规则 10 中可以看出，s_i 必须对 o_j 具有控制权才能删除 o_j（在这里，拥有权和控制权是一致的），o_j 被删除后，o_j 成为静止客体。此时，访问矩阵的第 j 列，即 o_j 所对应的列中各元素必须全部清空，不包含任何权限。

6、系统的定义

（1）. 符号约定

T={1, 2, 3, …, t, …}表示离散时刻的集合。常用作事件的下标，用来标识事件发生的顺序，在 BLP 模型中，它将用作请求序列，判定序列和状态序列的下标。

$X=R^T=\{x|\ x: T\rightarrow R\}$表示请求序列的集合，元素 x 是一请求序列，可表示为 $x=x_1x_2\cdots x_t\cdots$，时刻 t 时发出的请求用 x_t 表示。

$Y=D^T=\{y|\ y: T\rightarrow D\}$表示判定序列的集合，元素 y 是一判定序列，可表示为 $y=y_1y_2\cdots y_t\cdots$，时刻 t 时作出的判定用 y_t 表示。

$Z=V^T\{z|\ z: T\rightarrow V\}$是状态序列的集合，元素 z 是一状态序列，可表示为 $z=z_1z_2\cdots z_t\cdots$，时刻 t 的状态用 z_t 表示。

（2）. 系统的定义

设 $\omega=\{\rho_1, \rho_2, \cdots, \rho_s\}$是一组规则集，关系 $W(\omega)\subseteq R\times D\times V\times V$ 定义为：

⑴$(R_k, ?, v, v)\in W(\omega)$ 当且仅当对每个 i, $1\leqslant i\leqslant s$, $\rho_i(R_k, v)=(?,v)$；

⑵$(R_k, error, v, v)\in W(\omega)$ 当且仅当存在 i_1, i_2, $1\leqslant i_1<i_2\leqslant s$, 使得对于任意的 $v^*, v^*\in V$ 有 $\rho_{i_1}(R_k, v)\neq(?, v^*)$ 且 $\rho_{i_2}(R_k, v)\neq(?, v^*)$。

⑶$(R_k, D_m, v^*, v)\in W(\omega)$，$D_m\neq ?$，$D_m\neq error$，当且仅当存在唯一的 i，$1\leqslant i\leqslant s$，使得对某个 v^*和任意的 $v^{**}\in V$，$\rho_i(R_k, v)\neq(?, v^{**})$，$\rho_i(R_k, v)=(D_m, v^*)$。

上述定义中，（1）意味着请求 R_k 出错，没有一条规则适合于它；（2）意味着系统出错，有多条规则适合于请求 R_k；（3）意味着存在惟一条规则适合于请求 R_k。

$W(\omega)$由满足上述定义的四元组(R_k, D_m, v^*, v)组成，这意味着在状态 v 下，若发出某请求 R_K，必存在一判定 D_m（$D_m\in\{?, error, yes, no\}$），根据 ω 中的规则，将状态 v 转换为 v^*或保持状态 v 不变。

BLP 模型将系统记作$\sum(R, D, W(\omega), z_0)$，定义为：

$\sum(R, D, W(\omega), z_0)\subseteq X\times Y\times Z$，对任意$(x, y, z)\in X\times Y\times Z$，当且仅当对任一 $t\in T$，

$(x_t, y_t, z_t, z_{t-1})\in W(\omega)$时，有$(x, y, z)\in\sum(R, D, W(\omega), z_0)$。

其中，z_0 是系统初始状态，通常表示为 $z_0=(\phi, M, f)$。

由上述定义可知，系统是由 $X\times Y\times Z$ 中的某些三元组(x, y, z)组成，三元组中的

$x=x_1x_2\cdots x_t\cdots$为一请求序列

$y=y_1y_2\cdots y_t\cdots$为一判定序列

$z=z_1z_2\cdots z_t\cdots$为一状态序列

它们满足：$(x_1, y_1, z_1, z_0)\in W(\omega)$

$(x_2, y_2, z_2, z_1)\in W(\omega)$

$(x_3, y_3, z_3, z_2)\in W(\omega)$

……

$(x_t, y_t, z_t, z_{t-1})\in W(\omega)$

如图 4.5 所示，系统从初始状态 z_0 开始，接收用户的一系列请求 $x_1,x_2,\cdots,x_t,\cdots$，根据 ω 中的规则作出一系列相应的判定 $y_1,y_2,\cdots,y_t,\cdots$，系统状态从 z_0 逐步转化为 $z=z_1, z_2, \cdots, z_t, \cdots$

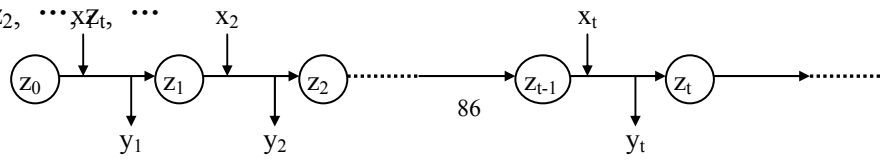


图 4.5　BLP 模型定义的系统

4.2.2 BLP 模型的安全性

1. BLP 模型的安全性证明

为了证明 BLP 模型所构建的系统是安全的，BLP 模型定义了安全状态、安全状态序列、系统的安全出现以及什么样的系统是安全系统，最后证明了 BLP 模型所构建的系统是安全的。

(1)安全状态

一个状态 v=(b, M, f)∈V，若它满足自主安全性，简单安全性和*-性质，那么这个状态就是安全的。

(2)安全状态序列

设 z∈Z 是一状态序列，若对于每一个 t∈T，z_t 都是安全状态，则 z 是安全状态序列。

(3)系统的一次安全出现

(x, y, z)∈∑(R, D, W(ω), z_0)称为系统的一次出现。

若(x, y, z)是系统的一次出现，且 z 是一安全状态序列，则称(x, y, z)是系统∑(R,D,W(ω),z_0)的一次安全出现。

(4)安全系统

若系统∑(R, D, W(ω), z_0)的每次出现都是安全出现，则称该系统是一安全系统。

(5)BLP 模型构建的系统是安全的

为了证明 BLP 模型所构建的系统是安全的，BLP 模型证明了以下两条主要的结论：

① BLP 模型的十条状态转换规则都是安全性保持的。即若 v 是安全状态，则经过这十条规则中任意一条规则转换后的状态 v^*也一定是安全状态。

② 若 z_0 是安全状态，ω 是一组安全性保持的规则，则系统∑(R, D, W(ω), z_0)是安全的。

显然，BLP 模型的初始状态 z_0=（ϕ, M, f）是安全的，BLP 模型又证明了它的十条状态转换规则均是安全性保持的，因此，BLP 模型所描述的系统是一个安全系统。

关于 BLP 模型的安全性证明的详细过程，请参阅文献[14]

2．BLP 模型的安全性和实用性分析

通过前面的介绍，可以看出 BLP 模型的安全目标是保护信息的机密性，对信息机密性的保护是通过安全级来进行控制。它的安全目标和控制策略特别适合于军事部门、政府办公部门和有层次结构的机构，因此，BLP 模型提出后，受到美国国防部的特别推崇，以至于在很长一段时期，人们将多级安全策略等同于强制访问控制，不仅许多操作系统使用它进行访问控制，而且在军事系统和办公系统也得到了广泛的应用。

BLP 模型是一个严格形式化的安全模型，并对其安全性给出了形式化的证明，这一优点是至今许多安全模型所不及的。

安全模型和安全模型的应用是有距离的，也就是说，在应用安全模型进行系统设计时，还必须根据实际系统的安全需求和应用需求来灵活实施访问控制。例如 BLP 模型在主体创造客体时，将其客体的安全级定义为该主体的安全级，从安全的角度来看是合理的。但在实际应用中，上级部门经常要下发若干文件，并指定文件下发到某一级时，要允许系统或代表系统的安全员对该客体的安全级进行降级定义。

又例如，在 BLP 模型中，信息只能由低向高流动，即若信息由客体 o_i 流向客体 o_j 时，必须满足 $f_2(o_i) \leqslant f_2(o_j)$, $f_4(o_i) \subseteq f_4(o_j)$。在实际系统中，客体 o 的范畴集 $f_4(o)$通常代表 o 的部门属性，BLP 模型中的这一要求限制了信息在部门之间的横向流动，显然是过于安全了。对此，在实际系统的控制中必须有相应的措施加以解决。

BLP 模型的三条安全特性集中体现了他的安全策略，说明 BLP 模型所描述的系统既实施自主访问控制也实施强制访问控制，用户对系统资源的访问必须同时通过这两层控制方可执行。他的自主访问控制是通过自主安全性来实现的，强制访问控制是通过简单安全性和*-性质来实现的。

孤立地看简单安全性，似乎有些不太安全。例如当主体 s 对客体 o 进行 append

操作时，BLP 模型不要求 o 的安全级支配 s 的安全级，当 s 对 o 进行 write 操作时，根据 BLP 模型对此项操作的解释，此时信息可以在 s 与 o 之间双向流动，但简单安全性仅要求 s 的安全级支配 o 的安全级。如此看来，简单安全级并不限制安全级的主体 s 将信息传递给低安全级的客体。

然而在简单安全性之后，BLP 模型还用*-性质对用户的操作进行了进一步的控制。如图 4.4 所示，当主体 s 向某客体 o 传送信息时，他的信息必须来自于比 o 安全级低或相等的客体。这可以说是补充了简单安全性的不足。但这只能是基于主体本身不含有信息这样一个观点。事实上，BLP 模型定义主体也可以看作是客体，这意味着主体本身也有可能包含有信息，在应用系统中亦是如此。为了解决这一问题可能带来的不安全性，人们在基于安全级进行了强制访问控制时，常简化为以下两条原则：

（1）不上读　主体只能读取比自身安全级更低或相等的客体。

（2）不下写　主体只能向安全级更高或者相等的客体写入。

BLP 模型是以保护信息的机密性为目标，信息只能由低安全级流向高安全级，但这样就有可能导致低安全级的信息破坏高安全级信息的完整性，也就是说，BLP 模型对信息完整性无法提供保护。

4.3 基于角色的访问控制与 RBAC96 模型族

随着计算机的广泛应用，特别是计算机应用由军事部门走向商业和民用部门，各行各业应用的多样化和安全需求的多样化，使得仅有自主访问控制和基于安全级的强制访问控制难以适应。在这一背景下，基于角色的访问控制成为安全领域的一个研究热点，并受到安全专家特别是访问控制专家的极大关注。著名访问控制专家 raviSandhu 等人，对 NIST 提出的基于角色访问控制的模型和理论进一步研究和完善，于 1996 年完成了 RBAC96 模型族的构造。

基于角色访问控制的核心是引入了角色的概念，它使得操作权限不是直接授予用户而是授予角色，用户通过角色身份来获得相应的操作权限。这种访问控制方法特别适合于同一个职务由多个成员来担任的应用场合。例如一个医院的外科医生、内科医生、儿科医生，…等等都必须有多个才能满足大量患者的看病要求。因此在医疗系统中可以定义外科医生，内科医生，儿科医生…等各种不同的角色，

在外科医生中又可细分为普通外科，胸外科和脑外科等，对此，在基于角色的访问控制中，可用子角色的方式来控制其权限。

又例如，在一个大型的办公系统中，由于来往的文件数量巨大，可能需要有多个文件收发员负责对外的文件收发工作，也需要有多个文秘人员负责文件的处理，如根据文件的来源单位和内容送给相关的部门和领导批阅，也需要有多个档案管理人员对处理完的文件进行归档保存，在这种情形下，便可在系统中设置文件收发员，文秘人员，归档人员…等等各种不同的角色，赋予这些角色中的成员相同的操作权限。

基于角色的授权方法相对于对单个用户授权大大的简化了授权的机制和管理，在用户的工作职务发生变化时，只要转换他的角色身份，而不需要对其重新授权，当机构设置发生变化时，如某个部门撤销，某些部门合并等也可以不修改应用程序，而只要修改角色与用户、角色与操作权限之间的配置关系即可。

下面通过 RBAC96 模型的描述，可以更为清楚地了解这一个访问控制方法的原理。

4.3.1 RBAC96 模型

RBAC96 模型族由 4 个模型组成，如图 4.6 所示。其中，

RBAC0 模型：基本模型。描述了具有 RBAC 安全功能的系统的最小需求。

RBAC1 模型：包含 RBAC0，增加了角色层次的概念。

RBAC2 模型：包含 RBAC0，增加了约束的概念。

RBAC3 模型：包含 RBAC1 和 RBAC2，由传递性，自然也包含了 RBAC0。

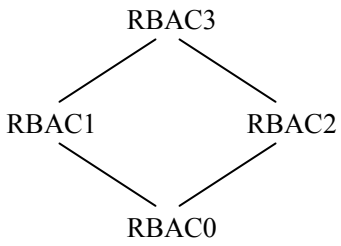


图 4.6　RBAC96 模型族

1、RBAC0 模型

（RBAC0 模型定义如下：）

定义 1：

（1）U 表示用户集，R 表示角色集，P 表示权限集，S 表示会话集；

（2）PA⊆P×R，是权限到角色的多对多指派关系；

（3）UA⊆U×R，是用户到角色的多对多指派关系；

（4）user: S→U，是会话到用户的映射函数，user(s_i)表示创建会话 s_i 的用户；

（5）roles: S→2^R，是会话到角色子集的映射函数，roles(s_i)表示会话 s_i 对应的角色集合；

$$roles(s_i) \subseteq \{r | (user(s_i), r) \in UA\}$$

（6）会话 s_i 具有的权限集 $P_{s_i} = \bigcup_{r \in roles(s_i)} \{p | (p,r) \in PA\}$

任何一个信息系统都会有使用该系统的许多用户，U 表示这些用户的集合。系统实施基于角色访问控制之前，按照实际应用背景，根据系统中不同的工作岗位需设置若干角色，R 表示所有角色的集合，P 表示系统中所有访问权限的集合。直观地讲，会话可解释为系统对用户一次请求的执行，每个会话由一个用户建立。一个用户可以同时打开多个会话，在不同窗口中运行。

PA 和 UA 在系统初始化时由安全管理员进行配置，PA 的元素在 P×R 的范围内指定，PA 的元素一经确定就意味着系统为每一个角色分配了相应的操作权限。UA 的元素在 U × R 的范围内指定。 UA 的元素一经确定，就意味着系统为每一个用户指定了相应的角色身份。

在系统的需求发生变化时，可由安全管理员对 PA 或 UA 的配置进行修改，亦可对 U、R、P 进行修改。

例如，一个医院有许多的医生、护士和药剂师。不妨设 $D_1, D_2, \ldots, D_m$ 是医生，$N_1, N_2, \ldots, N_r$ 是护士，$K_1, K_2, \ldots, K_n$ 是药剂师，医生的职责 DD={诊断病情，开处方，给出治疗方案，填写医生值班记录}，护士的职责 DN={换药，打针，填写护士值班记录}，药剂师的职责 DK={配药，发药}。

任一 D_i 可以尽医生的职责，执行 DD 中的操作，但不能执行 DN 和 DK 中的操作；同样，对护士 N_j (j=1, 2, …, r)和药剂师 K_t(t==1, 2, …, n)的限制也是类似的。但是在这里每一个医生的权限是相同的，每一护士的权限也是相同的，每一药剂师的权限也是相同的。因此医生，护士，药剂师便可以看作医疗系统中三种不同的角色，对用户权限的控制通过用户的角色身份进行，并不要区分该用户是谁。

在系统初始化配置时，安全管理员必须将 DD 中的权限指派给医生这一角

色，将 DN 中的权限指派给护士这一角色，将 DK 中的权限指派给药剂师这一角色，以完成系统中的 PA 指派。

另一方面，安全管理员还必须对医院的每一个职员进行角色指派，例如将张平指派为医生，将王莉指派为护士……等，以完成系统中的 UA 指派。可以为一个用户指派多个角色，例如将张平指派为医生，又指派为护士，那么张平具有两个角色身份，拥有 DD 和 DN 中的所有权限，若张平需给某病人看病，则系统此时根据需要激活张平的医生角色，使其行使医生的权限，也可同时激活他的医生和护士的角色，使其对病人同时行使医生和护士的权限（例如当护士不在时），甚至也可只激活他的护士角色，使其只能行使护士的权限。

函数 $user: S \rightarrow U$ 用来描述每一个会话是由哪个用户创建的。用户建立一个会话时，系统会激活他拥有的角色集的一个子集，用 roles(s_i)表示。会话 s_i 所具有的权限 Ps_i，便是这个子集 roles（s_i）中所有角色的权限的并集。

2、RBAC1 模型

RBAC1 模型包含 RBAC0 所有元素，并引入了角色层次的概念。角色层次反映在一个机构中不同的职务（或角色）不但具有不同的责任和权力，这些角色的权力之间还具有包含关系。职务越高的人责任和权力越大。在 RBAC1 模型中用偏序来描述角色之间的层次关系。在该偏序关系中，高级别的角色继承低级别角色的所有权限。因此一个用户若是某高级别角色的成员，则隐含了他同时也是低级别角色的成员，反之则不然。

（RBAC1 模型定义如下。）

定义 2：

（1）U、R、P、S、PA、UA 和 user: S→U 的定义与 RBAC0 相同；

（2）RH⊆R×R，是集合 R 上的偏序关系，称为角色层次关系；

（3）roles: $S \rightarrow 2^R$，是会话和角色子集的映射函数，但不同于 RBAC0，要求

$$roles(s_i) \subseteq \{r' \mid \exists r' \leq r \text{ 且 } (user(s_i), r) \in UA\}$$

它（他）表示会话 s_i 对应的角色集可以由建立该会话的用户所属的任何角色或其低级角色组成。

（4）会话 s_i 所具有的权限

$$P_{s_i} = \bigcup_{r \in roles(s_i)} \{p \mid \exists r' \le r \text{ 且} p \mid (p, r') \in PA\}$$

它表示会话 s_i 所具有的权限是由 roles(s_i)中的每一角色，以及被这些角色所覆盖的低级角色相对应的权限组成的权限集。

举一简化了的例子来说明 RBAC0 和 RBAC1 模型。

（例）假设某医院有医生四人，用 U={张，王，李，陈}表示。角色集 R={外科医生, 内科医生, 保健医生}，记作 R={r_1, r_2, r_3}，即医院有三种工作职务，代表三个不同的角色。权限集 P={制定治疗方案，做手术，开外用药，开内服药，开保健药}，记作 P={p_1, p_2, p_3, p_4, p_5}，于是

P×R={(p_1, r_1), (p_1, r_2), (p_1, r_3), (p_2, r_1), (p_2, r_2), (p_2, r_3), (p_3, r_1), (p_3, r_2), (p_3, r_3), (p_4, r_1), (p_4, r_2), (p_4, r_3), (p_5, r_1), (p_5, r_2), (p_5, r_3)}

U×R={(张, r_1), (张, r_2), (张, r_3), (王, r_1), (王, r_2), (王, r_3), (李, r_1), (李, r_2), (李, r_3), (陈, r_1),(陈, r_2), (陈, r_3)}

假设系统作以下配置：

PA={(p_1, r_1), (p_2, r_1), (p_3, r_1), (p_1, r_2), (p_4, r_2), (p_5, r_3)}

UA={(张, r_1), (张, r_2), (王, r_1), (李, r_2), (陈, r_3)}

上述配置意味着张和王被指派为外科医生，具有制定治疗方案、作手术和开外用药三种权限。张和李被指派为内科医生，具有制定治疗方案和开内服药两种权限。陈被指派为保健科医生，具有开保健药的权限。但这些权限都必须在相应角色被激活后，才能执行。

若张医生建立了一个会话(给一个病人看病)s_i，即若 user(s_i)=张，则 roles(s_i)$\subseteq$ {r_1, r_2}

若 roles(s_i)= {r_1}，即系统只激活他外科医生这一角色，则这次会话中张医生可具有的权限集 P_{s_i} ={p_1, p_2, p_3}；

若 roles(s_i)= {r_2}，即系统只激活他内科医生这一角色，则这次会话中张医生可具有的权限集 P_{s_i} ={p_1, p_4 }；

若 roles(s_i)={r_1, r_2}，即系统同时激活他外科医生和内科医生的角色，则这次会话中张医生可具有的权限集 P_{s_i} ={p_1, p_2, p_3}$\cup$ {p_1, p_4}={ p_1, p_2, p_3, p_4 }。

以上是 RBAC0 模型的系统控制情形。

如果在一个医院中有这样的一个规定，每一个外科医生和内科医生都必须具有保健医生的知识和技能，或者说都必须具有保健医生的从业资格，那么这就给上述三个角色间规定了一个层次关系，这时必须采用 RBAC1 模型，用 R 上的偏序关系来描述角色间的层次关系。

在此例中

$$R\times R=\{(r_1,r_1),(r_1,r_2),(r_1,r_3),(r_2,r_1),(r_2,r_2),(r_2,r_3),(r_3,r_1),(r_3,r_2),(r_3,r_3)\}$$

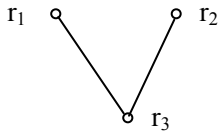


图 4.7　RH 的次序图

三个角色间的层次关系可用 R 上的偏序关系 RH 来表示。

RH={(r_1, r_1), (r_2, r_2), (r_3, r_3) , (r_3, r_1) , (r_3, r_2)}，其次序图如图 4.7 所示。r_1 覆盖 r_3，r_2 也覆盖 r_3，说明外科医生和内科医生不仅分别具有外科医生和内科医生的权限，同时也具有保健科医生的权限。

若张医生建立了一个会话（给一个病人看病）s_i，即若 user(s_i)=张，

则 roles(s_i)$\subseteq$ {r_1, r_2}$\cup$ {r_3}={ r_1, r_2, r_3}

例如，若 roles(s_i)={r_1}，则 P_{s_i} ={p_1, p_2, p_3}$\cup$ {p_5}={p_1, p_2, p_3, p_5}

若 roles(s_i)={r_2}，则 P_{s_i} ={p_1, p_4}$\cup$ {p_5}={p_1, p_4, p_5}

若 roles(s_i)={ r_1, r_2}，则 P_{s_i} ={p_1, p_2, p_3}$\cup$ {p_1, p_4}$\cup$ {p_5}={ p_1, p_2, p_3, p_4, p_5}

利用 RBAC1 模型的角色层次关系可以在系统中实现多级安全控制的要求。

3、RBAC2 模型

RBAC2 除了继承 RBAC0 已有的特征外，引入了一个约束（限制）集。它规定 RBAC0 各部件的操作是否可被接受，只有可被接受的操作才被允许。

约束是 RBAC 的一个重要功能，有时甚至认为它是 RBAC 出现的重要动机。约束是制定高层组织策略的有效机制，它为安全管理员的管理带来便利，特别当

RBAC 的管理是非中心化时，高级安全管理员可将它作为强制需求强加于其它安全管理员。

约束大多作用在 PA 和 UA 上，也可作用在会话的 user 和 roles 函数上。以下列举了一些常见的约束。

（1）互斥限制：同一个用户至多只能指派到相互排斥的角色集合中的一个角色。

例如，一个用户不能同时具有会计和出纳的角色身份，同样一个用户不能同时具有财务经理和采购经理的身份等。这样的角色称为是相互排斥的角色。这一约束有利于支持职责分散原则。

类似的可以作如下约束，同一个权限也至多只能指派到相互排斥的角色集合中的一个角色。

互斥限制有利于对重要权限的管理和分布上的限制，例如，也许角色 A 和角色 B 都可以对某个账号有签名权，但我们要求只能由其中的一个角色拥有这个权限。

（2）基数限制：一个角色的用户成员数目受限，一个用户隶属的角色数目也受限，一个权限能指派的角色数目受限，一个角色对应的权限数目也受限。

（3）先决条件限制：一个用户可以被指派到某角色 A 仅当他已是另一角色 B 的成员；一个权限 p 可以指派给某角色仅当该角色已拥有另一权限 q。

例如，只有那些已是工程中一般角色成员的用户，才能被指派到工程内的测试工程师角色。在许多操作系统中，要求用户先要有读文件所在目录的权力，然后才有读该目录中文件的权力。

（4）运行时的互斥限制：可以允许一个用户同时具有两个互斥的角色的成员资格，但在系统运行时不可同时激活这两个角色。

（5）会话数量的限制：限制用户同时进行会话的数量；限制一个权限活动的会话数量。

（6）时间频度限制：对特定角色权限使用的时间和频度进行限制。

实际上，根据需要还可以定义一些其它的限制。这些限制为了维护系统的安全，通过对系统的配置进行限制来实现对用户行为的约束。因此它体现了系统的强制访问控制。

角色层次也可以看作是一种约束，这种约束可描述为：

指派给低级角色的权限，也必须指派给所有比它高级的角色；指派成高级角色成员的用户，也必须指派成所有比它低级的角色的成员。

因此，从某种意义上来说，RBAC1 是多余的，它可以被包含在 RBAC2 中，但角色层次的存在是较常见的，直接支持角色层次比通过限制间接支持效率更高一些。

4、RBAC3 模型

RBAC3 模型将 RBAC1 和 RBAC2 两者结合，提供了角色层次和约束。

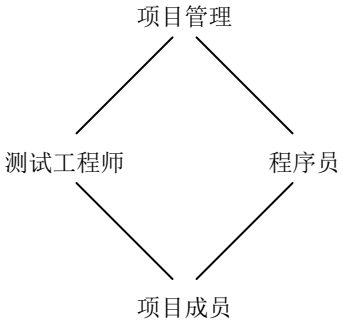


图 4.8 角色层次的例子

约束和层次结构之间存在一些影响。例如，在图 4.8 所示的例子中，假设测试工程师和程序员角色被说明为互斥的。那么项目管理角色就违反了这个互斥。一般情形下，高级角色违反这一约束是可以接受的，而在其他情形下就不行。我们认为模型应该考虑这些可能性，而不应该排斥这种或那种可能。

在基数限制中也有类似的问题。例如，假设一个用户只能指派到最多一个角色中，那么项目管理中的用户指派到他的下级角色就违反了这一约束，这涉及到基数限制是只应用于角色的直接成员关系，还是可以扩展到继承的成员关系。

在应用 RBAC2 模型进行系统设计时，可根据系统的实际需求来制定所需要的约束条件。

RBAC96 模型可用图 4.9 表示。

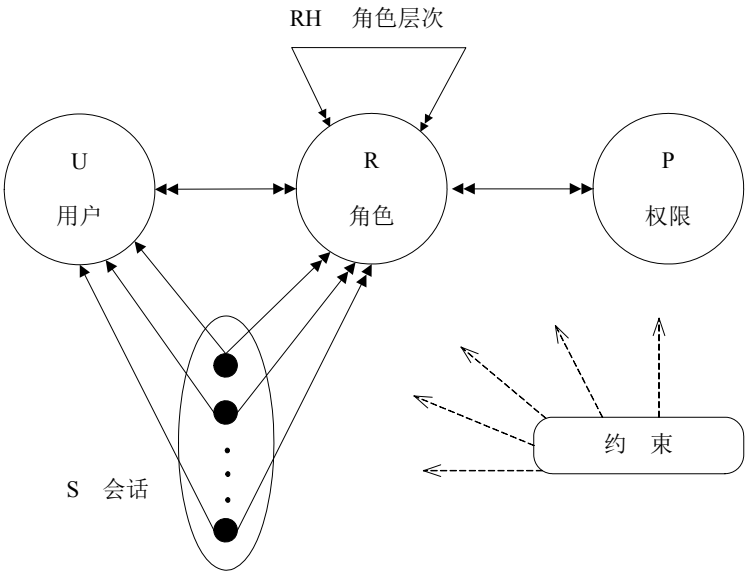


图 4.9 RBAC96 模型簇

4.3.2 基于角色授权模型的基本框架

在基于角色访问控制的系统中，当用户初次进入系统，系统根据用户的职务（或工作职责）为其指派相应的角色，从而让其取得这些角色的访问权限。但在系统的运行过程中，用户可能需要从一个角色转换成另一个角色。例如，职务升迁或工作调动；也可能在某些情况下，用户需要将自己的工作委托给另一个用户来完成，例如出差、生病等。为了适应这一类需求，系统可用授权机制来完成。

这里“授权”是指拥有某角色的用户能将该角色的成员资格授给其它的用户。根据实际需要，授权可以采取各种方式，下面列出授权的各种可能的特征。

1.永久性

永久性是指授权有效期的长短，分为永久与暂时两种类型。

永久性授权：被授权用户永久地取得了被授予的角色成员资格。

暂时授权：被授权用户只在一段时间内得到被授予的角色成员资格，一旦有效期结束，这种授权也就被回收。

2.同一性

同一性是指授权者授权后，自身是否还保持角色的成员资格。

同一性授权：授权后，授权用户仍拥有原角色的成员资格。

非同一性授权：授权后，授权用户丧失了原角色的成员资格，直到授权回收，他再重获原角色的所有权限。

3.全部性

全部性是指是否授出角色的全部权限。

全部性授权：授权用户将角色的所有权限授予被授权用户。

在这种情形下，角色的成员用户便分为两类：原始成员和被授权成员，前者是系统安全员最初分配到角色中的成员，后者是由该角色中的成员通过其授权分配到角色中的成员。

部分授权：被授权用户只被授予角色的部分权限。

4.执行性

执行性是指授权由谁执行。

自主执行：由授权者本人执行。

代理执行：授权者向第三方（某个代理）提出请求，让其完成自己的授权，但代理者不能给自己授权。

5.传递性

传递性是指被授权用户能否将角色成员资格转授出去。

单步授权：被授权者不能将得到的角色再转授给别的用户，这意味着角色中的被授权成员无权将该角色的权限授出去。

多步授权：角色的成员资格可以像接力棒一样，被传来传去。

6.多重性

多重性是指授权者在同一时刻可以对多个用户授权。

7.协议性

协议性是指授权是否要征得被授权者同意。

确认协议授权：授权必须有双方都认可的协议，以确保授权方与被授权方都同意此次授权。

非确认授权：不需要被授权者的认可，一旦授权者发出授权，被授权者必须接受。

关于授权回收可以有以下几种策略。

基于支撑角色的回收：若某个用户是在角色 A 的背景下，取得了角色 B 的授权，则当他角色 A 的授权被回收后，他的角色 B 的授权也被回收。

基于发起角色的回收：若用户甲对用户乙进行了某角色的授权，当用户甲的

角色被回收之后，用户乙也不能拥有该角色。

授权者有关回收：只有授出权限的用户才能回收他所授出的权限，角色中的任何其它成员均无权回收。

授权者无关回收：允许授权方角色中的任何原始成员回收该角色中的任一授权，不论这一授权是该角色中的哪一成员授出去的。

在授权者有关回收方式中，若授权用户不是采用基于发起角色的回收，其自身的角色成员资格被撤销时，那么该授权由谁来回收？另外，当被授权成员有不端行为时，可能会持续很长一段时间而不能发现。授权者无关回收能克服上述缺点，但它可能引起角色成员之间的冲突。

以上只是给出了授权与回收的一个框架模型，它们还可能有一些其它的特征，这可根据应用系统的实际需求来进行设计。

上述授权机制体现了用户的自主性，它与传统的自主访问控制相比，其授权不是基于权限，而是基于角色，简化了系统对授权的管理，也更适合于实际应用系统的需求。

在大型系统中，角色数可能是成百上千，而用户数则可能是成千上万，对这些角色和用户的管理是一项非常复杂的工作，它不可能由一个系统安全员去完成。下放 RBAC 的管理权，而又不失广义上的集中式控制的 ARBAC97 模型给出了如何基于 RBAC 来实现对 RBAC 的管理。

ARBAC97 模型包括三个部分：

URA97　　用户到角色指派管理；

PRA97　　权限到角色指派管理；

RRA97　　角色与角色的继承关系管理。

对 ARBAC97 模型的进一步讨论（超出了本书的范围），读者可参阅文献[23]。

4.3.3　RBAC96 模型安全性和实用性分析

由前面的介绍可以看出，RBAC 是中性的策略。它实际上是提供了一种描述安全策略的方法或框架。通过对 RBAC 各个部件的配置，以及不同配件之间如何进行交互，可以在很大的范围内使需要的安全策略得以实现。

例如，通过基于角色的授权方式可以实现自主访问控制，而且比传统的自主

访问控制更为灵活和方便。通过角色—用户的配置及角色—权限的配置可以实现系统所需要的各种强制访问控制策略，若采用 RBAC1 模型建立角色层次关系，则可实现多级安全的控制策略。RBAC2 的约束机制为实现强制访问控制提供了更为丰富的手段。RBAC 的角色---用户配置中最极端的情形一个角色仅一个用户时，便可等同的实现传统的自主访问控制和多级安全控制策略。

为适应系统需求的变化而改变其策略的能力也是 RBAC 的一个重要的优点。当应用系统增加新的应用或新的子系统时，RBAC 可以赋以角色新的访问权限，可以为用户重新分配一个新的角色，同时也可以根据需要回收用户的角色身份或回收角色的权限。

RBAC 支持如下三条安全原则：

1．最小特权原则

RBAC 可以使分配给角色的权限不超过具有该角色身份的用户完成其工作任务所必须的权限。用户访问某资源时，如果其操作不在用户当前被激活角色的授权范围之内，则访问也将被拒绝。

2．职责分散原则

RBAC 可从对互斥角色的用户进行限制，使得没有一个用户同时是互斥角色中的成员，并通过激活相互制约的角色共同完成一些敏感的任务，以减少完成任务过程中的欺诈。

3．数据抽象原则

在 RBAC 中不仅可以将访问权限定义为操作系统中（或数据库中）的读或写，也可以在应用层的定义权限，如，存款和贷款等抽象权限。支持数据抽象的程度将由实施细节决定。

RBAC 引入了角色的概念，通过角色与用户、角色与权限的配置为系统实现其安全控制提供了灵活且强有力的保护。这种保护可适用于多种不同的安全需求，而不仅限于多级安全。在计算机应用日益广泛，各种应用的安全需求呈现多样化的形势下，特别是对于金融，商业，和大型企业的安全控制，RBAC 提供了一种理想和实用的模式，因此自 1996 年 RBAC96 模型簇提出后，受到信息安全特别是访问控制专家的广泛关注和热烈讨论，并进一步探讨应用于多域访问控制之中。

习题四

4.1 在访问矩阵模型中，基于客体的授权表和基于主体的能力表各有什么优缺点？

4.2 授权的集中式管理模式和分散式管理模式各有什么优缺点？

4.3 BLP 模型的三条安全特性，反映了一种什么样的安全策略？

4.4 当用户对资源发出访问请求时，BLP 模型要进行一些什么样的安全检查？

4.5 RBAC96 模型簇由哪几个模型组成？各个模型的主要安全功能和区别是什么？

4.6 RBAC 是怎样实现强制访问控制的？

4.7 相对于 BLP 模型，RBAC 模型实现强制访问控制是否更加灵活？为什么？

4.8 如何在 RBAC 模型中实现自主访问控制？

4.9 根据实际应用的具体情况，在 RBAC 中基于角色的授权和回收还可能需要具有一些什么样的特征？

第 5 章　访问控制实例

在操作系统、数据库管理系统和应用系统中大量采用了访问控制技术来保证系统安全，通过本章，我们可以了解不同访问控制技术的安全保护目标以及在不同环境中是如何实现的。

5.1　操作系统访问控制技术

操作系统是方便计算机用户管理和控制计算机软硬件资源的系统程序，通常意义上的操作系统，除了基本的操作系统功能外，一般还包括很多工具程序，例如 Windows 操作系统为用户提供的桌面管理器、系统管理工具和附件，以及 Linux 操作系统提供的 Shell、基本工具集、Xwindows 图形界面系统以及 gnome、KDE 等桌面管理程序。安全操作系统是指符合安全评估标准、达到一定安全等级的操作系统。任何操作系统都必须具备一定的安全性，但是并非所有的操作系统都可以称为安全操作系统，一般认为，达到 TCSEC 的 B1 级或以上级别要求的操作系统才是安全操作系统。TCSEC 依据的安全策略模型是 BLP 模型，该模型所制定的最重要的安全准则——严禁上读、下写——针对的是信息的保密性要求，其主要的技术手段是访问控制机制。在我国，B1 级或以上的安全操作系统品牌已有 10 余种，这些操作系统绝大多数都是对 Linux 内核进行了改造。

操作系统访问控制是操作系统安全的重要部分，其目的是为了保护计算机软硬件资源的合法使用，根据安全保护目标的不同，操作系统中可能采用自主访问控制、强制访问控制和基于角色的访问控制等不同的访问控制机制。下面我们分别以 Windows2000/XP，Linux，SElinux 和红旗 Asianux Server 3 为例，来了解操作系统的访问控制基本技术。

5.1.1 Windows2000/XP 系统的访问控制技术

微软公司开发的 Windows 系列中，不同版本的操作系统安全性有一定的差异，并且还在不断地增强。早期 16 位的 Windows 系统安全性较弱，而现在常用的 Windows2000/XP 操作系统经安全评测达已达到了 TCSEC 的 C2 级标准。

Windows 要求所有的实体在使用操作系统中的各种服务、资源之前必须首先

验证其身份，然后根据身份进行授权和访问控制。

1、Windows2000/XP 身份认证过程

Windows 中计算机可以选择两种方式接入网络，一种是工作组方式，一种是域方式。工作组方式基本上是自由管理模式，每个计算机上的资源由机器的使用者决定是否共享给组内其他人使用；域是 Windows 为了统一管理多台计算机资源，实现资源安全共享，而采取的一种管理方法，域通过活动目录来统一管理用户帐户，以及网络打印机、Web 服务器和电子邮件服务器等网络资源。

Windows2000/XP 中的用户分为本地用户和域用户，多个用户可以形成一个用户组，用户组也分为本地组和域上的组。本地用户通过本地安全认证系统 LSA 在本地计算机进行身份认证，域用户通过网络认证系统 Net-Logon 在域控制服务器上进行身份认证。

Windows2000/XP 的身份认证过程如图 5.1 所示。

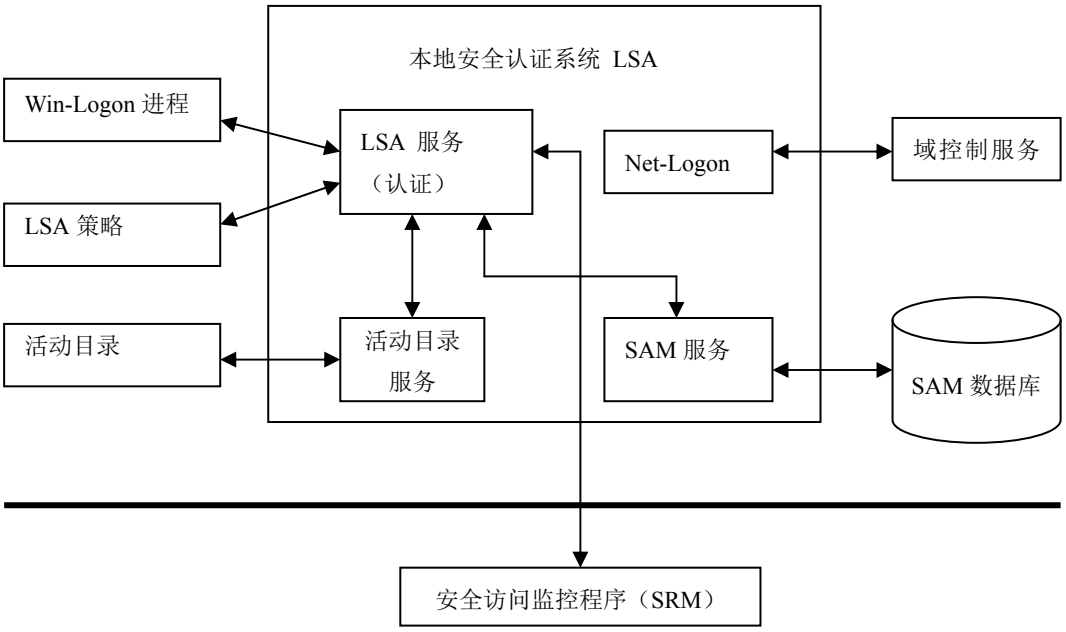


图 5.1 Windows 用户认证与访问控制系统

（1）Win-Logon 进程

Win-Logon 是一个运行 WINLOGON.EXE 的用户态进程，负责获取用户名和密码，发送给 LSA 验证。当用户按下“Ctr+Alt+Del”时，Win-Logon 会调用图形化识别和验证组件 GINA（Graphical Identification and Authentication），显示登

录窗口，切换到安全桌面并提示输入用户名称和密码。Win-Logon 为该用户创建唯一的本地组，将桌面的该实例（键盘、鼠标和屏幕）分配给该用户。为阻止特洛伊木马，登录窗口仅可由 Win-Logon 访问。

（2）LSA 服务

本地安全认证系统 LSA（Local Security Authority）是 Windows 系统中一个重要的服务，是一个运行映像 LSASS.EXE 的用户态进程，负责本地安全规则、用户身份验证及发送安全审计消息。它从 WINLOGON.EXE 中获取用户的账号和密码，然后调用身份认证包（默认是 MSV1-0.DLL），与存储在账号数据库中的信息进行对比，如果对比的结果匹配，LSA 就认为用户的身份有效，允许用户登录计算机。

LSA 策略数据库包含了系统安全性规则设置，主要内容是哪些用户可以访问系统、谁被赋予了哪些权限等。保存在注册表中的 HKEY-LOCAL-MACHINE\security 下。

如果是域用户，那么，用户的域认证请求会通过 Net-Logon 发送到域控制服务器上的 LSA 服务进行认证。

（3）SAM 数据库

安全账号管理器 SAM（Security Account Manager）通过存储在计算机注册表中的安全账号来管理用户和用户组的信息。我们可以把 SAM 看成一个账号数据库。对于没有加入到域的计算机来说，它存储在本地，而对于加入到域的计算机，它存储在域控制器上。SAM 数据库包含用户的密码和属性，保存在 HKEY-LOCAL-MACHINE\SAM 下的注册表中。

（4）活动目录 Active Directory

活动目录既是一个包含网络资源（如计算机、用户和打印机）的数据库，也是一个目录服务系统，它使得数据库中的信息通过目录服务提供给用户和程序使用。目录服务启动后，Windows 可以在整个活动目录范围内对用户的登录进行验证。

（5）安全访问监控程序 SRM（Security Reference Monitor）

SRM 负责访问控制和审计策略，由 LSA 支持。SRM 提供客体（文件、目录等）的访问权限，检查主体（用户帐户等）的权限，产生必要的审计信息。因此，SRM 是联系身份认证和访问控制的纽带。

2、Windows 访问控制基本要素

Windows2000/XP 操作系统为维护系统资源的安全性，采用了基于访问控制列表的自主访问控制机制，对用户/用户组实施访问控制。为了实现访问控制机制，操作系统必须明确定义主体、客体和权限的表示方法，以及权限的授予方法，这些内容构成了操作系统访问控制的基本要素。

（1）主体和安全标识

在 Windows 操作系统中，主体包括用户、用户组以及进程、线程等主动实体。Windows 系统为每个用户和用户组分配一个安全标识符 SID（Security Identifiers），组的安全标识符有时也称作 GSID。安全标识符在同一个域中是唯一的，系统中的 SID 包括标识符标记、版本号、颁发机构号、子颁发机构号以及为了解决重复问题的 RID（Relative ID）。例如一个 SID 可能为：

S-1-5-21-123625317-425641126-188346712-2895

其中第一项 S 表示该字符串是 SID；第二项是 SID 的版本号，对于 Windows2000 来说，这个值为 1；第三项表示标识符的颁发机构，本例中 5 表示 NT 颁发机构，其他还例如 9 表示 Exchange 2007 颁发机构等；第四到七项用来表示域或者本机的标识，即子颁发机构；最后一项是 RID，RID 对系统内缺省用户是固定的，例如 500 表示 Administrator，501 表示 Guest，由系统管理员创建的帐户的 RID 通常大于或等于 1000。

Windows 系统除了负责维护一些新创建的用户和用户组的 SID，还维护着一些缺省的、固定的 SID，称作知名 SID（well-known SIDs），它们可能不包含 SID 的所有子项，例如 S-1-1-0 表示 EveryOne 组，S-1-5-18 表示 Local System 帐户等。

一个进程或者线程的安全标识符包含在其访问令牌中，缺省情况下和进程或者线程的发起者的 SID 相同。

（2）客体和安全描述符

Windows 系统中的客体包括一切 Windows 可识别的访问对象，包括文件对象、注册表键、网络共享、互斥量、信号灯、事件、进程、线程、令牌、硬件、服务和驱动等。Windows 系统中的安全对象是指具有安全描述符 SD（SECURITY_DESCRIPTOR）的对象。Windows 中绝大多数对象都是安全对象，少数如 FAT 文件系统下的文件、目录等不是安全对象。 由于安全描述符中可以包含结构复杂的访问控制所需的权限信息，因此，系统对安全对象可以实施细粒

度的访问控制，而一般对象，仅可实施粗粒度的访问控制或者根本不作任何控制。

安全描述符主要包括所有者安全标识符，所属主组的安全标识符，系统访问控制表 SACL（System ACL）、自主访问控制表 DACL(Discretionary ACL)和屏蔽标志。

在 Windows API 中，一个安全描述符可描述如下：

```
typedef struct _SECURITY_DESCRIPTOR
{
 BYTE Revision; /* currently SECURITY_DESCRIPTOR_REVISION */
 BYTE Sbz1;        /* 0 */
 SECURITY_DESCRIPTOR_CONTROL Control;
                          /* The type of security descriptor */
 PSID Owner;       /* points to a SID (the owner SID) */
 PSID Group;       /* points to a SID (the primary group) */
 PACL Sacl;        /* An array of discretionary accesses */
 PACL Dacl;         /* the auditing accesses */
} SECURITY_DESCRIPTOR, *PISECURITY_DESCRIPTOR;
```

我们可以看到，去掉两个版本控制成员 Revision 和 Sbz1，一个安全描述符由五个成员组成：

自主访问控制列表（Dacl）：保存着对象的许可（允许谁访问对象，而拒绝谁）。

系统访问控制列表（Sacl）：指定要对对象执行的审核的类型。如果发生了审核事件，会被存储到审核事件的日志中。

指定对象的所有者（Owner）：一个 SID，对象的所有者总是可以更改安全描述符，而不管其他人对访问的锁定。

指定对象的主组（Group）：一个 SID，Windows 通常忽略此参数（这是为了 POSIX 兼容性，现在已经退化了）。

Control：表现为一个 16 位整数的一组标志。可以为零或以下标志：

SE_DACL_PRESENT：Dacl 成员有效。

SE_SACL_PRESENT：Sacl 成员有效。

SE_DACL_AUTO_INHERITED：DACL 从其包含的父（例如父进程、上级文件夹等）那里自动继承并得到其中各项。

SE_SACL_AUTO_INHERITED：和 SE_DACL_AUTO_INHERITED 一样，只不过应用于 SACL。

SE_DACL_PROTECTED：如果父的 ACL 与这里定义的任何 ACL 冲突，覆盖父的 ACL。用于覆盖继承。

SE_SACL_PROTECTED：和 SE_DACL_PROTECTED 一样，只不过用于 SACL。

SE_DACL_DEFAULTED： Dacl 成员等同于此类对象的缺省 DACL。

SE_SACL_DEFAULTED： Sacl 成员等同于此类对象的缺省 SACL。

SE_GROUP_DEFAULTED：Group 成员等同于此类对象的缺省组。

SE_OWNER_DEFAULTED：Owner 为缺省的。

SE_SELF_RELATIVE：表示此安全描述符是自相关的。

此结构的后四项都是指针，指向 ACL 等所在的缓冲区。这些指针可以指向内存中的任意合法地址，而且并不需要在一个连续块中。由于安全描述符不是连续的，所以将安全描述符写到磁盘或者令之跨越进程将会是一件相当复杂的事情。微软通过引入一个称为自相关安全描述符的结构来解决这一问题。

Windows 系统中安全对象的安全描述符管理比较复杂，而且也不是集中存储和管理的。例如 NTFS 文件系统中每个文件夹和文件都有安全描述符，为了减少安全描述符的存储量，NTFS 文件系统将具有相同安全描述符的文件、文件夹合并成一项，存储在 MetaData 文件中，同时，通过通过建立索引的方法来进行快速查找。而系统中文件系统以外的一些安全对象的安全描述符可能存在注册表或者其他文件中。

（3）访问控制列表

Windows 系统中访问控制列表有两种，即系统访问控制列表 SACL 和自主访问控制列表 DACL。系统访问控制列表是为审计系统服务的，包含了对象被访问的时间和访问结果等内容；自主访问控制列表表示针对某一对象的所有访问权限。

一个访问控制列表由多个访问控制项（Access Control Entries）组成。访问控制项 ACE 可以描述为：

```
struct ACE
{
    BYTE AceType;        /* can be allow/deny/audit/custom */
    BYTE AceFlags;       /* Inherited? */
    ...
    ACCESS_MASK Mask; /* Specifies the action */
    ...
    SID Sid;             /* associate with this user/group/... */
} ;
```

其中，AceType 表示该访问控制项是允许、拒绝、审计（仅用于系统访问控制列表）还是其他定制的类型；AceFlags 表示该访问控制项是否是从父继承过来的；Mask 是访问屏蔽位，是一个 32 比特的整数，可以包括读、写、创建、删除、修改等功能，在 Windows 中，无论什么时候，当用户要对一个对象执行操作（比如说读）时，要执行的操作会被编码为一个 32 位的整数（称为 ACCESS_MASK），例如，如果要读文件，那么 ACCESS_MASK 就应该是 FILE_GENERIC_READ，该整数和访问屏蔽位进行与操作后结果不变，说明该操作与访问控制项中的操作是相匹配的；Sid 表示该访问控制项的主体。

自主访问控制列表包含了用户和组以及在对象上的访问权限（例如只读或者拒绝访问等），拒绝访问的级别高于允许访问的级别，Windows 系统在应用层会自动将一个访问控制列表内的访问控制项进行排序，拒绝访问控制项在允许访问控制项的前面，在进行权限比对的时候，就只需依次从前往后进行判断。

同样，当用户执行了一个操作后，操作系统会在该操作的安全对象的系统访问列表中进行查找，如果发现了相匹配的项，就将该操作事件写入审计日志。

（4）访问令牌（Access Tokens）

用户通过登录进程的验证后，登录进程会给用户生成一个访问令牌，该令牌相当于用户访问系统资源的票证，当用户试图访问系统资源时，代表用户进行访问的进程将获得该访问令牌的一个拷贝。当进程访问操作系统中的一个安全对象或者执行系统任务而需要相应权限的时候，进程中的线程会将访问令牌提供给操作系统进行检查，操作系统会从访问控制令牌中获取用户和用户组的信息，并与对象的 DACL 中的访问控制项 ACE 进行匹配，如果请求操作被允许，那么访问

就被允许，否则进程将会得到一个拒绝访问的错误状态。

一个访问令牌包含如下内容：

- 用户 SID，表示线程以哪个用户身分运行；
- 用户所属用户组的完整列表，用多个 SID 表示；
- 用户登录认证的信息；
- 用户和用户组的特权列表；
- 线程创建新的安全对象时，其安全描述符使用的缺省的拥有者 SID；
- 线程创建新的安全对象时，其安全描述符使用的缺省的主组的 SID；
- 当用户创建一个安全对象而未指定安全描述符 SD 时，线程使用的缺省 DACL；
- 访问令牌的来源，例如 Session Manager, LAN Manager 和 RPC Server 等；
- 令牌是主令牌还是模拟令牌；
- 受限 SID 的列表，表示不允许模拟的 SID 的列表；
- 当前模拟的级别，包括四种，分别为不能识别用户身份，只能以匿名方式模拟；可以识别用户和用户权限，但不能以用户权限模拟；可以在本地模拟；可以远程模拟。

Windows 中的令牌分为两类，即主令牌和模拟令牌。Windows 是一个多线程的操作系统，一个单独的进程可以拥有同时运行的多个线程。模拟意味着把其他用户的令牌设置到当前线程上。通常，访问检查是根据线程令牌执行的，但当线程在模拟一个用户时，用户令牌被分配给了该线程，该线程的访问检查就根据线程拥有的令牌执行，而不是线程本身的令牌，访问完成后，线程会还原，意味着用户令牌被该线程丢弃。

为什么要使用模拟呢？假设操作系统中有一个服务程序需要访问一个名为 sec.txt 的文件，服务进程的创建者帐户 A 拥有读、写、删除和更新该文件的权限。如果没有模拟，任何用户都能潜在地和 A 一样读、写、删除和更新该文件，有了扮演，用户就只能以自己本身的权限来访问该文件，对 sec.txt 的访问就会受到控制。

（5）权限与授权

用户的权限是一个用户能在操作系统上执行的所有操作的总称。在 Windows 系统中，用户在被管理员创建的时候，就拥有该用户所在用户组的权限，以后，可能通过授权等操作发生改变。权限由在访问控制列表 ACL 中定义的许可访问权限和一系列的特权（privilege）组成。特权是指一些操作系统定义的安全策略，比如关闭系统或是安装驱动程序，又如 SeBackupPrivilege 特权可以允许用户绕

过 NTFS 文件权限的限制，读（备份）任何文件和文件夹等。这些特权可以通过组策略（用户权利指派）来配置，并包含到用户的线程令牌中。特权在缺省状态下是被禁用了的，因此在使用之前首先要启用它。

组策略（gpedit.msc）是管理员为用户和计算机定义并控制程序、网络资源及操作系统行为的主要工具。组策略对本地计算机可以进行两个方面的设置：本地计算机配置和本地用户配置。所有策略的设置都将保存到注册表的相关项目中。对计算机策略的设置保存到注册表的 HKEY_LOCAL_MACHINE 的相关项中，对用户的策略设置将保存到 HKEY_CURRENT_USER 相关项中。

Windows 2000/XP 采用自主访问控制机制，一个用户如果是对象的拥有者，或者被授予授权传递的能力，那么，他可以将自己的权限授予其他的用户。

例如在一个 NTFS 格式的分区上，用鼠标选定某一文件或文件夹，通过鼠标右键查看该文件或文件夹的属性，就可以在安全一栏中看到其安全属性。图 5.1 列出了所有可访问文件 network.log 的用户和用户组（如果一个用户的权限仅从其用户组继承过来，而没有附加的权限，那么不再单独列出），以及这些用户和用户组所具有的访问权限，这些权限就是从文件的安全描述符 SD 中读取出来的。文件的拥有者可以修改这些权限，如果文件的拥有者将“完全访问”的权限授予其他用户，那么这些用户就也可以修改文件的权限，并将自身的这些权限转授给他人。

图 5.2 NTFS 文件系统权限

在 Windows 系统中，Administrators 组的用户具有任意访问权限和所有特权，因此除非必要，日常工作中，普通用户尽量不要使用 Administrators 组的用户来操作计算机，以免感染病毒或者误操作，而造成对计算机重要受保护文件的损坏。

3、Windows 访问控制流程

用户经过认证后，LSA 会产生一个访问令牌， SRM 根据调用者的访问令牌查询访问对象的安全描述符进行授权检查。客体的安全属性由访问控制项（ACE）来描述，客体所有的 ACE 组成访问控制列表（ACL）。没有 ACL 的客体意味着任何主体都可以访问。而有 ACL 的客体则由 SRM 检查其中每一项 ACE，从而决定主体的访问是否被允许。

Windows 访问控制流程如图 5.3 所示。

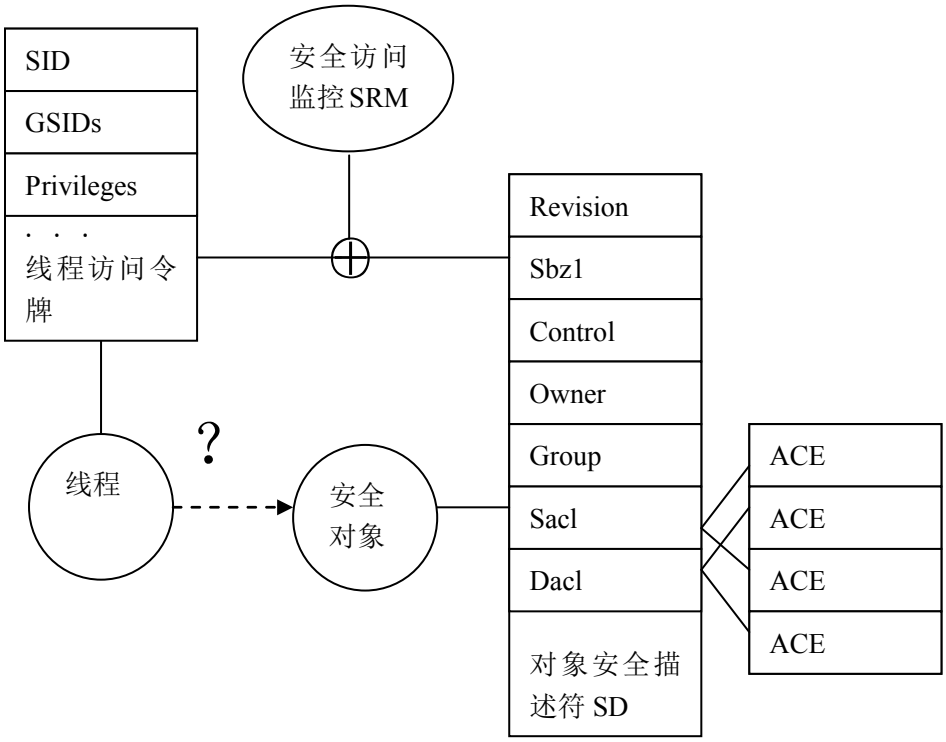


图 5.3 访问控制流程

线程代表认证后的用户执行对象的访问，一个线程是否能够访问一个安全对象，通过下面的步骤来进行检查：

(1) 安全访问监控程序获取线程令牌，取得用户和组 SID；
(2) 安全访问监控程序获取对象的安全描述符，从安全描述符中得到 DACL 和 SACL；
(3) 对 ACL 中的每个 ACE，都找到其相关的 用户和组 SID，并与线程令牌中的 SID 进行比较；
(4) 如果找到了，看 ACE 是拒绝 ACE 还是允许 ACE；
(5) 若是一个允许 ACE，将其 ACCESS_MASK 与线程的 ACCESS_MASK 进行比较；
(6) 如果所有的访问方式都允许，那线程被就允许访问；
(7) 如果有错误发生，那线程就被拒绝访问；
(8) 如果 ACE 是审计 ACE，那么就将此次操作写入审计日志。

4、小结

Windows2000/XP 以自主访问控制为主来实现操作系统的访问控制，操作系统初装时，有许多预定义的安全策略，以后管理员可以通过组策略工具进行配置，这些策略在操作系统中以特权的方式存在。Windows2000/XP 操作系统中对象（例如文件）的创建者具有对对象的拥有权，他可以将自己对对象的权限任意授给其

他用户，并决定该用户是否具有转授这些权限的权力，授权一般通过 Windows 所提供的工具，例如文件属性浏览工具，网络共享工具等。

Windows2000/XP 操作系统以能力表和授权表两种方式来存储操作系统中的访问控制矩阵，并根据系统中权限分配的改变自动更新，例如，Windows2000/XP 中特权是以能力表的方式存储的，而安全对象的安全描述符是以授权表的方式存储的。根据实际情况灵活采用不同的存储方式，可以减少访问控制矩阵的存储体积，提高访问的速度。

从 Windows Vista 开始，Windows 操作系统中还包含了完整性控制等新的访问控制技术以不断弥补其在安全性方面的不足。

5.1.2 Linux 操作系统的访问控制技术

Linux 是一种类 UNIX 的操作系统，当前 Linux 操作系统主要发行版本的安全性大致处于 TCSEC 的 C2 级，采用基于访问控制矩阵的自主访问控制机制。

1、 主体标识

在 Linux 操作系统中，主体仍然是用户/用户组，用户继承所处用户组的所有权限，用户的帐号基本信息存放在/etc/passwd 中，每个用户的信息在此文件中占一行，例如：

root:x:0:0:root:/root:/bin/bash

可以看出，passwd 文件中每条用户记录由七个域组成，相邻域之间用冒号隔开，其基本形式为：

登录帐号：密码域：UID：GID：用户信息：主目录：用户 shell

①登录帐号：用户登录时的帐号名，即用户名。

②密码域：用户加密后的密码，如果是“x”，则表示密码并不保存在 passwd 文件中，而是映射到/etc/shadow 文件中。

③UID：用户标识，在系统中每个用户的 UID 是唯一的，系统管理员 root 的 UID 为 0。

④GID：用户组标识，GID 为 0 的组对应于 root 用户组。

⑤用户信息：创建用户时对用户的一些解释说明，可选。

⑥主目录：用户登录时的默认目录。

⑦用户 shell：用户所使用的 shell 类型，可以设为/bin/bash，/bin/csh 等类型。

2、 客体标识和权限

Linux 操作系统中所有的计算机软硬件资源都被映射成了文件，因此，Linux 操作系统中的任何客体都可以用文件名来标识。

Linux 操作系统中，文件的权限通常用权限字表示，文件的拥有者可以把文件的访问属性设成三种不同的模式：读(r)，写(w)和运行(x)和三个不同的用户级别：文件拥有者(u)，所属的用户组(g)，系统里的其他用户(o)。以下命令可以查看文件的访问权限：

ls －l filename

如果文件对于三种不同的用户都提供三种文件访问模式，输出结果看起来应该是：

-rwxrwxrwx

第一个字符“-”显示文件的类型，一般而言，“-” 表示普通文件，“ d” 表示目录文件，“ l” 表示链接文件，“c”表示字符设备，“b”表示块设备，“p”表示命名管道，比如 FIFO 文件（First In First Out, 先进先出），“ f”表示堆栈文件，比如 LIFO 文件（Last In First Out，后进先出）。

第一个字符之后的第一个三位字符组表示文件拥有者对该文件的权限，第二个三位字符组表示文件用户组对该文件的权限，第三个三位字符组表示系统其他用户对该文件的权限。如果没有权限，一般显示“-”字符。

以下是一个显示一个属于 root 的文件的用户权限：该文件的拥有者 root 拥有所有权限，但是用户组和其他用户只能阅读和执行。

drwxr-xr-x 2 root root 21504 Apr 24 19:27 dev

第一个字符 d 显示该文件是一个目录（对目录而言，执行权限通常是指可搜索权限）。

3、授权

（1）文件从属

每个文件（或者目录）从属于一个文件拥有者（一般是一个用户名）和一个用户组。文件拥有者一般来说就是生成（或者拷贝）这个文件的用户。一个文件只能被文件拥有者删除，或者是文件所属的用户组里的其他用户，或者是 root 用户。对于其他用户，如果被赋于适当的权限，也有可能修改或者删除该文件。例如要查看文件 junk 的拥有者：

ls －l junk

屏幕输出如下：

-rwx------ 1 yogin inca 27 Apr 24 14:12 junk

该文件属于拥有者 yogin 和用户组 inca。

文件的拥有者可以通过命令 chown（修改文件拥有者）和 chgrp（修改用户组）来修改，

chown peter junk

chgrp peter junk

ls －l junk

执行以上三条命令后，命令 ls －l junk 输出如下：

-rwx------ 1 peter peter 27 Apr 25 20:27 junk

说明文件 junk 的拥有者变成了用户 peter 和用户组 peter。文件的拥有者发生改变后，文件原来的拥有者将不再拥有该文件，文件从属关系只能由文件的拥有者或者 root 用户进行改变。

（2）权限的改变

文件的拥有者通过改变文件的权限对其他用户进行授权。

可以使用 chmod 命令来修改文件的访问权限。例如，以下命令将把文件 junk 给所有用户增加“只读”权限。

chmod a+r junk

在以上的命令，除了用“a”表示所有用户(all)，还可以用“u”表示用户(user)，“g”表示用户组(group)，“o”表示其他用户(other users)。除了加号“+”增加权限，还可以使用减号 “-”删除权限，等于号“=”设置权限。除了“r”表示只读权限(read)，还可以用“w”表示写权限(write)，“x”表示执行权限(execute)。

又如，以下命令将删除其他用户对 junk 文件的执行权限：

chmod o-x junk

除了字符，也可以使用数字来设置权限：

execute=1

write=2

read=4

权限的组合只需把权限按位进行或运算即可，例如“只读和执行”权限“r-x”可以用数字表示为 5（1+4）。

要给三个不同的用户级别设置访问权限，只需要把三个数字粘在一起就可以了。举例：

chmod 770 junk

将给文件拥有者和所属用户组所有权限（读，写和执行），而对于其他用户没有任何权限。

chmod 666 junk

将给所有用户（文件拥有者，所属用户组，其他用户）读写权限，但是没有执行权限。

chmod 411 junk

目录的访问权限和一般文件的访问权限是不同的。对于目录来说：

r =允许列出该目录下的文件和子目录

w =允许生成和删除该目录下的文件

x =允许访问该目录

（3）缺省文件属性

Linux 使用 umask 设置缺省文件属性，当一个文件生成时，系统给以文件缺省的文件权限。如果系统的缺省权限是“-rw-r—r--”，意味着由该用户生成的文件能被该用户读和写，而用户组和其他用户只能读。可以使用以下命令检查我们刚生成的文件的缺省权限：

umask -S

可选项－S 代表“符号”，告诉 umask 按容易阅读的格式显示文件权限，而不是缺省的数字格式。可以修改新生成文件的缺省权限：

umask u=rw,g=,o=

对于新生成的文件，以上命令将给文件拥有者以读和写的权限，而用户组和其他用户将没有任何访问权限。

在 umask 命令里使用数值来设置文件的缺省属性比较麻烦。因为数值显示的是从用户那里去除掉的权限（刚好和 chmod 相反），比如：

umask 000

表示对于新生成的文件，将给所有人所有的权限。

umask 177

表示给文件拥有者以读和写的权限，而其他用户没有任何权限。

为了让设置在系统启动时即生效，Linux 管理员通常在文件/etc/profile 或.bash_profile 里修改对应的行。

4、 POSIX ACL

传统的基于权限字的访问控制只能对文件设定属主、组和其他人的权限，如果想为一个文件交叉定义若干个不同组的用户访问权限，比如 Tom，Mary，Tony，Tod 分别属于不同的组，某一文件想让 Mary 和 Tony 只读，Tom 和 Tod 可写，其他用户不可访问，这种要求用传统的基于权限字的访问控制是无法实现的。为了给不同的用户或用户组定义不同的使用权限，在 Linux 2.4 内核中，访问控制列表 ACL 就作为一个系统补丁存在，而在 2.6 内核中，它已经是标准内核的一部分了。Linux 和大多数 UNIX 一样遵循 POSIX ACL 标准，支持 ACL 需要内核和文件系统的支持，现在 2.6 内核配合 EXT2/EXT3, JFS, XFS, ReiserFS 等文件系统都可以支持 ACL。

在 Linux 中，一个访问控制表 ACL 由许多访问控制项 ACE 组成，每个访问控制项包括三个域：关键字域 Entry tag type，用于标识项类型；访问者域 qualifier (optional)，包括一个组或者一个用户的名字；权限域 permission。其中，Entry tag type 域有以下几个类型，

ACL_USER_OBJ： 相当于 Linux 里文件拥有者的权限。

ACL_USER： 定义了额外的用户可以对此文件拥有的权限。

ACL_GROUP_OBJ： 相当于 Linux 里文件所属组的权限。

ACL_GROUP： 定义了额外的组可以对此文件拥有的权限。

ACL_MASK： 定义了 ACL_USER, ACL_GROUP_OBJ 和 ACL_GROUP 的最大权限 。ACL_OTHER： 相当于 Linux 里其他用户的权限。

例如我们用命令 getfacl file1 来查看文件 file1 的访问控制列表，系统显示：

```
# file: file1
# owner: root
# group: root
user::rw-
user:testu1:rwx #effective:r--
group::r--
group:testg1:r--
```

mask::r--

other::r—

前面三个以#开头的定义了文件名，文件拥有者和文件所属组。

user::rw- 定义了 ACL_USER_OBJ， 说明文件拥有者拥有读和写的权限。

user:testu1:rwx #effective:r-- 定义了 ACL_USER，说明用户 testu1 拥有了对文件的读写和执行权限，但实际上，由于 mask 的限制，该用户的实际权限为只读，这一情况通过#effective:r—表示出来了。

group::r-- 定义了 ACL_GROUP_OBJ，说明文件所属组拥有只读权限。

group:testg1:r-- 定义了 ACL_GROUP，使得 testg1 组拥有了对文件的读权限。

mask::r-- 定义了 ACL_MASK 的权限为只读。

other::r-- 定义了 ACL_OTHER 的权限为只读。

从文件 file1 的 ACL 我们可以看出，只有 user 和 group 类型的 ACE 才有访问者域，通过访问者域可以指定特定用户和组的权限。

如果文件 file1 所在的文件系统启用了 ACL，那么命令 ls -l file1 会显示结果

-rw-r--r--+ 1 root root 7 Dec 11 00:28 file1

和前面的 ls 命令比较，在权限字部分多了一个“+”号，以表示文件拥有额外的访问控制项。

5、 小结

Linux 系统中主体就是用户和用户组，客体就是文件，Linux 操作系统通过在文件系统中存储授权表来实现访问控制矩阵。Linux 中文件的拥有者可以将自身对文件的权限（包括拥有者权限）任意授予其他用户。root 用户拥有对系统所有文件的任意权限。

Linux 操作系统一般通过 POSIX ACL 替代传统的权限字来实现细粒度的访问控制，在这种方式下，文件的拥有者可以决定将文件的权限授予某些指定的用户和用户组，而不授予其他的用户和用户组。

5.1.3 SELinux 和红旗 Asianux Server 3 安全技术

1、 SELinux 安全操作系统简介

SELinux（Security-Enhanced Linux）是由美国国家安全局 NSA 于 2000 年底发布的 Linux 操作系统内核的安全增强，它采用 Flask/Fluke 安全体系结构，可

支持基于角色的访问控制和强制访问控制。目前，大多数安全操作系统都是在SELinux 的基础上进行扩充的。

SELinux 提供了比传统的 Linux 权限更好的访问控制。例如，管理员可以只允许一个应用程序添加记录到一个日志文件但不允许其重写或者删除该日志文件的内容。虽然 ext2 和 ext3 文件系统有一个 append-only 标签（使用 chattr 设置），但是这个属性不区分某一个进程（不能在为一个访问 append-only 的同时，又允许另一个进程据有完全可写的权限）；另一方面，一个应用程序可以被允许在一个文件夹中建立文件和向其写入数据，但不能删除文件：这种特性是没有SELinux 的普通的 Linux 内核所不能做到的。

SELinux 的设计遵循了安全操作系统的设计原则：分配给每个主体对客体的访问特权是完成其工作所必需的特权的最小集合，有效地防止了特权的转移；MAC 和 RBAC 机制简单明了，并经过严格验证；MAC、RBAC 和 LSM（Linux Security Modules，Linux 安全模块）保护机制是公开的；每个主体对每一个客体的每一次访问都必须经过检查，以确认是否已经得到授权，并且采用明确的授权访问方式；通过域和角色来实现权限分离；使用 LSM 可以使安全增强模块以可加载内核模块的方式加入内核，在实际的 SELinux 发布中，用户可根据自己的需要来选择加载 SELinux 安全模块来启动安全增强子系统，方便用户使用。

SELinux 采用域-类型模型，通过域和类型以及建立在它们之上的规则来决定主体是否有对客体的访问权限。

在使用了 SELinux 的系统中，每一个进程的上下文都包含三个组成部分：一个 ID（identity），一个角色（role）和一个域（domain）。ID 是指这个进程的所有者，就是账户，但前提是这个账户必须被预先编译到 SELinux 策略中去使 SELinux 认识这个账户，不然的话 SELinux 默认地将那些未知的系统进程 ID 记为 system_u ，将那些未知的用户进程 ID 记为 user_u；角色用来判断某个处于此角色的 ID 可以进入哪些域，还用来防止某个处于此角色的 ID 进入其它不该进入的域。比如， user_r 角色就不允许进入 sysadm_t （重要的系统管理域）。

SELinux 安全域中运行着的每一个进程和每一个资源（一般文件、目录文件和套接字等）都有一个与之相联系的“类型”（type）。系统管理员在这基础之上建立了一系列规则，这些规则列出了某个域可以在每一个类型上执行的所有动

作。 域-类型模型的一个优点就是我们可以对策略进行分析，从而判断出哪些信息有可能外溢，这在一般的 Linux 系统上是无法实现的。

2、 红旗 Asianux Server 3 安全技术

红旗 Asianux Server 3 以 SELinux 为基础，遵照国家标准 GB17859-1999《计算机信息系统安全保护等级划分准则》所规定的内容，通过引入安全等级、安全分组等安全标记，以 BLP 模型作为强制访问控制框架，实现了全部安全标记保护级(第 3 级)和部分结构化保护级(第 4 级)所要求的等级保护功能于 08 年 4 月通过 CC EAL4（相当于 TCSEC 的 B1 级）认证，我们仅对该操作系统所使用的访问控制技术进行简单的介绍，详细内容参考红旗 Asianux Server 3 的相关手册。

（1）特权分离

普通 Linux/Unix 的用户特权划分只有两级，超级用户和普通用户。超级用户具有所有的特权，普通用户没有特权。这种做法不符合安全系统的“最小特权”原则。攻击者只要获得超级用户身份，便得到了对整个系统的完全控制。

红旗 Asianux Server 3 根据“最小特权”原则对超级用户的特权进行了化分，在日常系统管理员角色之外单设了安全管理员角色。系统管理员负责系统的安装、管理和日常维护，如安装软件、增添用户帐号、数据备份等。安全管理员负责安全策略的制定和执行，安全属性的设定、安全审计等职责。传统的超级用户(Root)与其它标准用户一样，操作行为将受到安全管理员部署的所有安全策略的制约。即使由于人为管理疏忽使得非法用户获得 Root 权限，其操作行为仍受限于安全策略限制并被实时审计。

（2）保密性和完整性强制访问控制

红旗 Asianux Server 3 通过改进的 BLP 模型实现了保密性访问控制和完整性访问控制，以阻止信息的非法访问，和防止涉密信息被非法篡改。红旗 Asianux Server 3 开发了易用的配置管理工具和配置模板，增加强制访问控制的易管理性，并简化了实现，提高了部署的实用性。

（3）基于角色的访问控制

RBAC 是红旗 Asianux Server 3 的重要功能，红旗 Asianux Server 3 实现了基于角色的访问控制并提供了集成化的配置管理工具，管理员可以通过配置管理工具设置合适的角色及权限，按‘最小特权’原则分配和限制用户权限，从而灵活有效的控制主体对客体的访问。

（4）基于 ACL 的自主访问控制

红旗 Asianux Server 3 通过访问控制列表（ACL）机制细化了对系统资源的访问控制粒度， 可以实现系统中任一用户（组、角色）对各种系统资源（目录，文件，特殊文件等）的控制访问。其自主访问控制的结构和流程如图 5.4 所示。

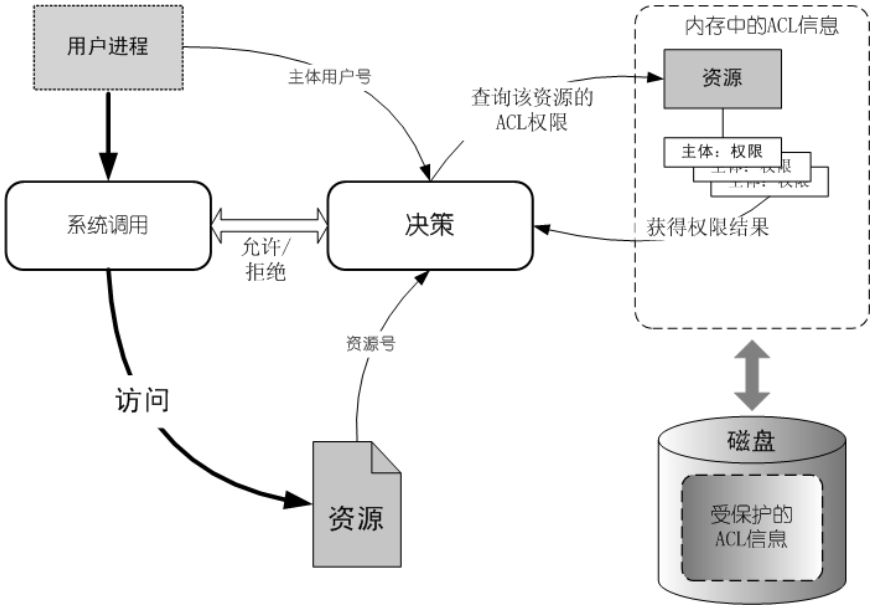


图 5.4 自主访问控制结构和流程

除了提供上述标准客体的 ACL 保护外，红旗 Asianux Server 3 中的自主访问控制还提供了对于 Linux 安全敏感资源的 ACL 控制，如限制命令和 Setuid 程序的执行，以及阻止进程被 Kill 等等。

5.2 数据库访问控制技术

美国国家计算机安全中心（NCSC）颁布了 TCSEC 后，又于 1994 年 4 月继续颁布了《可信计算机系统评估准则的可信数据库管理系统解释》(Trusted Database Management System Interpretation of the Trusted Computer Evaluation Criteria)，简称 TDI。它将 TCSEC 扩展到数据库管理系统。

在美国的大型 DBMS 中，多数产品都有达到 B1 或相当于 B1 级别的版本，我国安全 DBMS 的研究与开发正处于起步和发展阶段，有一些单位开发了达到 B1 级安全标准的原型产品。

为了实现数据库的安全保护，用户首先向数据库提供身分识别信息并证明身份识别信息的有效性，数据库对用户的身份进行识别和认证，如果身份正确，数

据库将会在基于身份识别信息的基础上决定用户所拥有的权限以及所得到的授权。

5.2.1 Oracle 数据库中的身份认证

数据库主要的认证机制是用户名和口令，因为用户名能被猜测到，而且在某些情况下也是众所周知的，所以对用户的认证非常关键。

Oracle 数据库的主要认证手段有三种：即密码认证、外部认证或强认证和代理认证。

（1）密码认证。主要是指数据库中的用户名、口令认证方式。

（2）外部认证或强认证。Oracle 支持操作系统级的外部认证，即委托操作系统来创建和管理数据库用户。Oracle 还支持强认证，包括 PKI 数字证书、Kerberos、DCE 和 RADIUS。RADIUS 的标准扩展了认证能力，可以包括标识卡、生物识别和智能卡。

（3）代理认证。Oracle 支持针对数据库连接和数据库会话的细粒度管理，从而把物理连接和数据库会话相互分离开来。代理认证是为了解决类似 Web 应用等应用模式中，多个应用共享一个数据库用户访问数据库，从而带来的对访问控制的不利影响。在代理认证中，应用程序首先通过代理帐号建立与数据库相连接的连接池，这是物理连接，代理帐号通常没有访问数据库中数据的能力，应用程序在访问数据库时，必须将会话中的实际数据库用户与代理连接关联起来，使得不同用户访问数据库的会话得以区分开来。

Oracle 还支持通过 Oracle 企业用户安全(Enterprise User Security,EUS)对数据库用户进行集中化管理、身份验证和授权。Oracle 同时支持数据库和部署在 Oracle 应用服务器上的 Web 应用的单点登录，提供了单点登录所需要的可扩展性、安全性和高可用性，例如，数据库对强认证的支持也包含了对单点登录技术——例如 PKI 数字证书、Kerberos、DCE——的支持。

5.2.2 Oracle 数据库访问控制技术

1、 权限

Oracle 数据库权限分为三级：系统权限、对象权限和对象内部权限。

系统权限是指作用于整个数据库中的所有对象的权限，或者是对数据库本身

状态进行更改的权限。例如：EXECUTE ANY PROCEDURE 权限允许被授权的用户可以执行定义在数据库任何模式(除 SYS 模式外)之中的任何过程，SELECT ANY TABLE 权限允许被授权的用户可以查询数据库任何模式下的表，ALTER DATABASE 权限允许被授权的用户修改数据库参数等。

视图 system_privilege_map 列出了所有数据库系统权限，管理员可以通过语句

SQL>SELECT name FROM system_privilege_map;

来查看系统中的权限。查看谁拥有什么系统权限，需要查询 dba_sys_privs，例如想了解谁拥有 SELECT ANY TABLE 权限，那么可以进行如下查询，查询结果将显示拥有该权限的用户和数据库角色：

SQL>SELECT grantee FROM dba_sys_privs

WHERE privilege=’SELECT ANY TABLE’;

Oracle 数据库中的对象权限是指对具体对象的操作权限，例如对某个模式下的一个或多个表的查询、修改权限，对某个模式下视图或过程的访问权限等。用户通过视图 USER_OBJECT_PRIVS 可以查询其所拥有的对象权限。例如 SQL 语句

SQL>SELECT privilege,object FROM user_object_privs

WHERE owner=’scott’;

可以查看用户所拥有的、对 scott 用户的对象的所有对象权限。

Oracle 对象内部权限是指对对象的部分访问权限，例如对表中指定行或者列的访问权限。对象内部权限所针对的对象在数据库中通常没有具体的命名，它们是通过规则定义而逻辑存在的。对象内部权限是实施细粒度访问控制的前提。

2、 授权

在 Oracle 数据库中，可以把权限直接授予用户，也可以先把权限授予角色，然后再把角色授予用户。

（1）直接授权

数据库管理员通过执行语句

SQL>GRANT SELECT ANY TABLE TO scott;

可以将 SELECT ANY TABLE 的系统权限直接授予用户 scott。

如果数据库用户 USER1 拥有表 account_table，那么他执行语句

SQL>GRANT SELECT ON account_table TO scott;

可以将对表 account_table 的查询权限授予用户 scott。系统管理员也可以通过执行语句

SQL>GRANT SELECT ON USER1.account_table TO scott;

达到同样的效果。

数据库通过 REVOKE 语句，将授出的权限撤销，例如 USER1 执行语句

SQL>REVOKE SELECT ON account_table FROM scott;

可以将他授予 scott 的对表 account_table 的查询权限撤销。

在 Oracle 数据库中，如果把某个权限授予某个拥有管理权的用户，那么该用户就能把这个权限授予其他用户。如果拥有管理权的用户被系统删除了，那么他授予其他用户的所有对象权限也全部失效，但是，任何系统权限不会失效！

（2）数据库角色

数据库角色是一种给用户间接授权的方式。采用角色间接授权可以简化管理，而且可以进行灵活配置。如果想给一组用户添加或删除权限，只需要写一条简单的语句为角色添加或删除该权限即可，系统将为被授予该角色的每位用户添加或删除该权限。如果有 100 个用户，直接为他们授予权限，则需要写 100 条给这些用户添加或删除权限的独立语句。

Oracle 数据库支持角色层次关系。执行以下语句

SQL>CREATE ROLE a;

SQL>GRANT SELECT ON table1 to a；

SQL>CREATE ROLE b;

SQL>GRANT SELECT ON table2 to b；

SQL>GRANT a TO b;

SQL>GRANT b TO scott;

则在数据库中创建了角色 a 和 b，角色 a 拥有对 table1 的查询权限，角色 b 则同时拥有对 table1 和 table2 的查询权限，用户 scott 通过角色 b 来获得 b 所拥有的对数据库的访问权限。

角色的层次关系为角色的管理提供了灵活性，视图 dba_role_privs 中保存了用户/角色直接拥有的角色，而视图 user_object_privs 和 system_privilege_map 中保存了用户/角色拥有的对象权限和系统权限，以及权限的授予者，因此结合这

些视图，可以精确追踪到用户角色层次，以及每个权限是通过哪个角色获取的。

在 Oracle 数据库中，角色不属于任何模式，当一个用户被删除后，该用户创建的角色并不会被删除，角色中的所有系统权限仍然保留，但是对象权限都被删除了。

Oracle 数据库允许在应用系统运行过程中激活角色，从而达到动态授权的目的，为此 Oracle 提出了安全角色的概念。安全角色主要包括基于口令保护的角色和安全应用角色。

(a) 基于口令保护的角色

基于口令保护的角色是指除非提供口令，否则不能启用利用口令保护的角色。对于一般角色，用户可以通过 SET ROLE 命令恢复未被激活的角色。但是如果是基于密码的角色用户就必须在激活时提供该角色的密码，而不是简单的 SET ROLE 或者运行 DBMS_SESSION.SET_ROLE 存储过程所能完成的。以下语句创建了一个基于口令保护的角色 pass_protected_role，启用该角色的口令为 secpass，该角色被指派给了用户 scott。

SQL> CREATE ROLE pass_protected_role IDENTIFIED BY secpass;

SQL> GRANT pass_protected_role TO scott;

SQL> ALTER USER scott DEFAULT ROLE ALL EXCEPT pass_protected_role;

最后一个 SQL 语句表示默认情况下，scott 登录数据库后，不会激活角色 pass_protected_role。而且，由于 scott 不知道角色 pass_protected_role 的口令，所以他也不能通过任何工具来激活这个角色。

如果 pass_protected_role 的口令只有应用程序知道，那么，在应用程序执行过程中，可以通过执行

SET ROLE pass_protected_role IDENTIFIED BY secpass;

或者调用 dbms_session.set_role 存储过程来动态激活该角色，这样，可以避免用户 scott 绕过应用系统的访问控制而直接访问数据库数据。

（b）安全应用角色

安全应用角色也称为包验证角色，就是用 Oracle 中的内置函数或存储过程来验证角色，使用安全应用角色的好处是由数据库最终决定是否启用或者禁用角

色。这是在 oracle 9i 后引入的。安全应用角色使得角色的使用必须通过两道保护：首先，即使用户或应用程序有机会使用角色，它们也必须拥有 PL/SQL 程序的执行权限；其次，PL/SQL 程序本身可以采取一系列的验证措施。

为了创建安全应用角色，必须指定将要作为角色的 PL/SQL 程序。下面将创建名为 sec_app_role 的角色，并利用 sec_mgr 模式的 priv_mgr 程序判断是否启用或禁用该角色。以 sec_mgr 的身份登录，执行：

```
SQL>CREATE ROLE sec_app_role IDENTIFIED USING sec_mgr.priv_mgr;
SQL>GRANT sec_app_role TO scott;
SQL>ALTER USER scott DEFAULT ROLE CONNECT,RESOURCE;
```

程序 priv_mgr 的创建过程为：

```
SQL>CREATE OR REPLACE PROCEDURE priv_mgr
2 AUTHID CURRENT_USER
3 AS
4 BEGIN
5 IF (SYS_CONTEXT(‘userenv’,’ip_address’)=’127.0.0.1’
6 THEN
7 dbms_session.set_role(‘sec_app_role’);
8 END IF;
9 /
```

用户 scott 要启用角色 sec_app_role，首先必须要被授予执行 priv_mgr 程序的权限，此外他还必须在数据库服务器所在的计算机上才能启用，而不能远程启用。

3、 Oracle 数据库的强制访问控制技术

支持强制访问控制的 Oracle 产品一般被叫作可信 Oracle（Trusted Oracle），是 Oracle 数据库中单独的一个版本，并需要运行在特定的操作系统之上。利用可信的 Oracle，数据库的任何一个部分都贴上安全标签（包括数据库进程 SMON、PMON、监听器等，数据库文件，数据库用户，数据库对象等等）。由于可信操作系统和可信数据库的可用性逐渐降低，Oracle 在发布完最后一个可信数据库版本——Trusted Oracle 7.2.3 后就不再更新了。

但是强制访问控制在很多情况下是符合用户安全需求的，因此用户希望数据库能实现独立于严格可信操作系统的基本标签功能。

Oracle 标签安全(Oracle Label Security，OLS)允许用户通过给数据记录贴上安全标签来实施强制访问控制。其中的标签标识了用户必须具有什么样的权限才可以读或写数据，标签保存在给表增加的针对每条记录的特殊列中。数据库用户也被贴上了标签，指明他们能够访问数据记录的权限。当用户访问表中的记录时，数据库的 OLS 引擎会比较用户的标签和行的标签，从而判断用户是否拥有访问该记录的权限。OLS 只应用于需要行级安全的表。

实施标签安全的步骤如下：

a)创建 OLS 策略，策略是标签、用户权限和受保护的数据库对象的容器。

b)定义 OLS 标签组件，每一个标签都包含可以互换的三个组件：级、格、组。

c)创建我们希望使用的真正的 OLS 标签。必须为 OLS 定义有效的标签。根据应用的安全策略，利用第二步定义的组件，创建一组有效的标签。这一步是可选的，但一般都需要完成，以确保标签数据的完整性；

d)为表或模式应用 OLS 安全策略。为表增加一列标签列，并根据标签的结果增加支持行级安全所需要的一些设置。这一步同时也定义了策略的安全实施行为。

e)给用户或应用程序分配将要使用的标签权限。这一步确定了谁最终能访问哪些数据，标签被分配给用户，而用户可以是单个用户，也可以是一组用户，通过比较用户的标签以及数据记录的标签，OLS 确定用户是否可以访问数据。

4、 行级和列级访问控制技术

早期的 Oracle 数据库是通过视图的方式来实现行级或列级的访问控制，虽然视图是实施安全保护的流行方法，但是在不同的 DML 语句中混合安全保护策略将增加其复杂性，在 Oracle8.1.5 版本中引入了虚拟私有数据库（Virtual Private Database,VPD）的技术，可以简单实现行级访问控制，并在 10g 中还引入了一个被称作列敏感策略的新特性，可以实现列级的访问控制。

以行级虚拟数据库为例，VPD 通过创建策略函数透明地更改对数据的请求，基于一系列定义的标准向用户提供表的局部视图。在运行策略时，所有查询都附

加了谓词，以便筛选出允许用户看到的行。例如，如果只允许用户查看帐户管理员 scott 的帐户，则 VPD 可通过设置，自动地将查询：

SELECT * FROM accounts;

重写为：

SELECT * FROM accounts WHERE name = 'scott';

为了实现这一功能，DBA 在表 accounts 上设置了一项安全策略。该策略具有一个相关函数，称为 policy function，它返回一个用作谓词的字符串 where name = 'scott'，附加在对表 accounts 的操作上。

5、 小结

Oralce 数据库管理系统是一个典型的融合了自主访问控制、强制访问控制和基于角色的访问控制等多种访问控制机制于一体的安全产品。首先，通过授权，对象的拥有者可以将对该对象的权限任意授予其他用户，体现了自主访问控制的灵活性；第二，Oracle 主流产品提供了独立于严格可信操作系统的基本标签功能，可以满足用户强制访问控制的要求；第三，Oracle 通过角色给用户间接授权，大大简化了权限的管理。此外，Oracle 还通过虚拟私有数据库 VPD、视图等机制来尽量满足访问控制应遵循的“最小特权”原则，提高数据库管理系统和应用的安全性。

5.3 应用系统访问控制实例

5.3.1 网络防火墙访问控制实例

随着互联网的飞速发展和互联网应用的普及，人们越来越关注网络安全问题。网络安全性是一个涉及面很广泛的问题，最基本的网络安全主要是确保无关人员不能访问，更不能修改传送给其他接收者的信息。当越来越多的企业、单位和个人享受互联网给人们带来的便利的同时，如何防范由此带来的非法入侵、篡改和破坏等安全问题也成为人们需要解决的当务之急。在众多的网络安全解决方案中，防火墙扮演了非常重要的角色。

防火墙是指设置在不同网络(如可信任的企业内部网和不可信的公共网)或网络安全域之间的一系列报文控制部件的组合，可对通过防火墙的网络通信数据进行监测、限制和更改，从而实现网络的安全保护。防火墙可以通过软件或硬件的方式来实现，不论何种方式，其核心功能都是通过特定的访问控制策略，设置

防火墙的网络数据包的穿行规则，从而阻止或允许特定的数据流通过防火墙。

防火墙是典型的访问控制设备，被访问主机可认为是资源客体，而访问者则是访问主体。通过防火墙，可以灵活根据资源的开放程度或是保护需求，制定主体对资源的访问规则，这些规则限制并约束了主体的访问行为。从访问控制的策略来看防火墙实施的是强制访问控制。

要设置防火墙访问规则，首先必须对数据流的来源和目的进行标识，即确定通信的双方主机。目前广泛使用的互联网传输协议通过 IP 地址唯一地标识一台互联网上的主机。主机之间的通信数据通过路由协议在一个或多个网关（或称路由）设备之间传递和转发，最终达到目标主机，如图 5.3 所示。

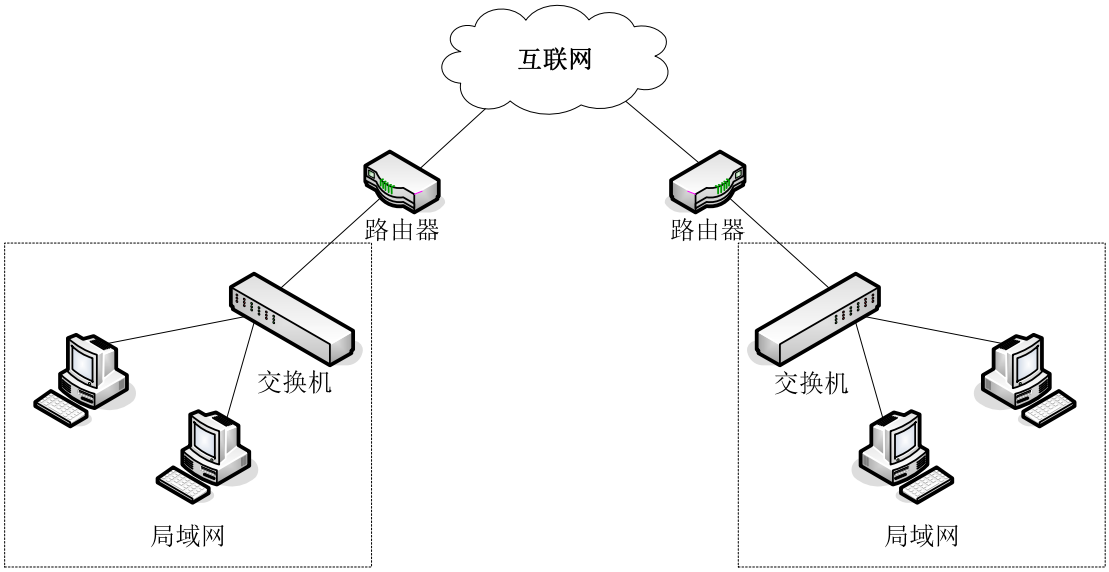


图 5.3 互联网主机通信示意图

由通信示意图可知，路由设备在网络通信中具有关键作用，如果在路由设备上增加是否转发数据的规则判断，则可实现对双方的通信控制，这就是防火墙的基本原理。

当前，防火墙的种类从实现方式上可以分为以下几类：

(1) 包过滤（packet filter）：

对每个数据包按照用户所定义的进行过滤，如比较数据包的源地址、目的地址是否符合规则等。包过滤不管会话的状态，也不分析数据。如用户规定允许端口是 21 或者大于等于 1024 的数据包通过，则只要端口符合该条件，数据包便可以通过此防火墙。如果配置的规则比较符合实际应用的话，比如：针对安全级别设置不同的 VLAN(虚拟内部局域网)，在这一层能够过滤掉很多有安全隐患的数

据包。

(2) 应用网关（application gateway）：

检验通过此网关的所有数据包中的应用层的数据；经常是由经过修改的应用程序组成运行在防火墙上。如 FTP 应用网关，对于连接的 client 端来说是一个 FTP server，对于 server 端来说是一个 FTP client。连接中传输的所有 ftp 数据包都必须经过此 FTP 应用网关。

(3)电路级网关（circuit-level gateway）：

此电路指虚电路。在 TCP 或 UDP 发起（open）一个连接或电路之前，验证该会话的可靠性。只有在握手被验证为合法且握手完成之后，才允许数据包的传输。一个会话建立后，此会话的信息被写入防火墙维护的有效连接表中。数据包只有在它所含的会话信息符合该有效连接表中的某一入口（entry）时，才被允许通过。会话结束时，该会话在表中的入口被删掉。电路级网关只对连接在会话层进行验证。一旦验证通过，在该连接上可以运行任何一个应用程序。以 FTP 为例，电路层网关只在一个 FTP 会话开始时，在 tcp 层对此会话进行验证。如果验证通过，则所有的数据都可以通过此连接进行传输，直至会话结束。

(4)代理（proxy）：

通常情况下指的是地址代理，一般位于一台代理服务器或路由器上。它的机制是将网内主机的 IP 地址和端口替换为服务器或路由器的 IP 地址和端口。举例来说，一个公司内部网络的地址是 129.0.0.0 网段，而公司对外的正式 IP 地址是 202.138.160.2~202.138.160.6，则内部的主机 129.9.10.100 以 WWW 方式访问网外的某一台服务器时，在通过代理服务器后，IP 地址和端口可能为 202.138.160.2:6080。在代理服务器中维护着一张地址对应表。当外部网络的 WWW 服务器返回结果时，代理服务器会将此 IP 地址及端口转化为内部网络的 IP 地址和端口 80。使用代理服务器可以让所有的外部网络的主机与内部网络之间的访问都必须通过它来实现。这样可以控制对内部网络带有重要资源的机器的访问。

由上述防火墙分类可知，防火墙可以从区域、VLAN、地址、用户、连接、时间等多个层面对数据报文进行判别和匹配，访问控制规则的源和目的既可以是已经定义好的 VLAN 或区域，也可以细化到一个或多个地址对象以及用户组对

象。与报文阻断策略相同，访问控制规则也是顺序匹配的，系统首先检查是否与报文阻断策略匹配，如果匹配到报文阻断策略后将停止访问控制规则检查。但与报文阻断策略不同的是访问控制规则没有默认规则。也就是说，如果没有在访问控制规则列表的末尾添加一条全部拒绝的规则的话，系统将根据目的接口所在区域的缺省属性（允许访问或禁止访问）处理该报文。

下面以某企业采用天融信网络卫士防火墙进行网络安全防护为例，说明防火墙的访问控制原理。整个网络示意图如图 5.4 所示。从图中可看出，该网络有三个安全域，分别对应防火墙的 Eht0、Eth1 与 Eth2。

Eth0 口属于内网（area_eth0）安全域，为交换 trunk 接口，同时属于 VLAN.0001 和 VLAN.0002，vlan.0001 IP 地址为 10.10.10.1，连接研发部门文档组所在的内网（10.10.10.0/24）；vlan.0002 IP 地址为 10.10.11.1，连接研发部门项目组所在的内网（10.10.11.0/24）。

Eth1 口 IP 地址为 192.168.100.140，属于外网 area_eth1 安全域，公司通过与防火墙 Eth1 口相连的路由器连接外网。

Eth2 口属于 area_eth2 区域，为路由接口，其 IP 地址为 172.16.1.1，为信息管理部所在区域，有多台服务器，其中 Web 服务器的 IP 地址：172.16.1.3。

针对该三个网络安全域，用户的访问控制要求如下：

- 内网文档组的机器可以上网，允许项目组领导上网，禁止项目组普通员工上网。
- 外网和 area_eth2 区域的机器不能访问研发部门内网；
- 内外网用户均可以访问 area_eth2 区域的 WEB 服务器。

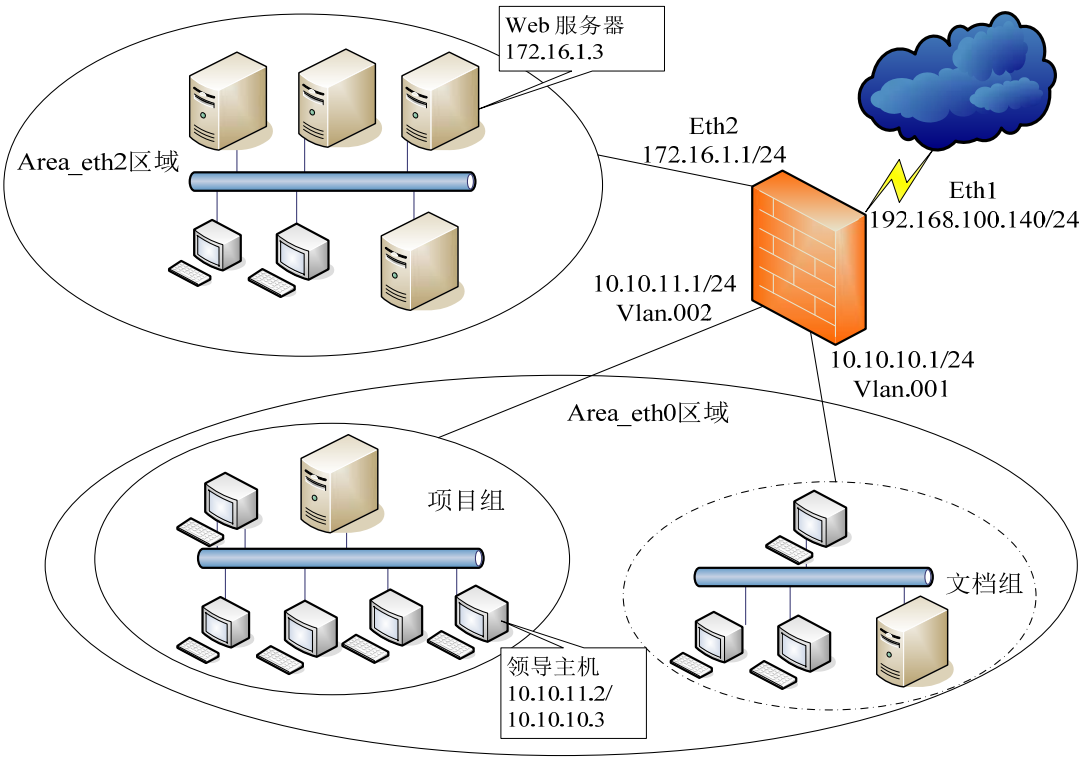


图 5.4 某企业网络配置示意图

上述访问控制需求体现了防火墙既能够防止外部的非法访问，又能够对内部用户的上网行为进行约束。具有以下三条配置要点：

- 设置区域对象的缺省访问权限：area_eth0、area_eth2 为禁止访问，area_eth1 为允许访问。
- 定义源地址转换规则，保证内网用户能够访问外网；定义目的地址转换规则，使得外网用户可以访问 area_eth2 区域的 WEB 服务器。
- 定义访问控制规则，禁止项目组除领导外的普通员工上网，允许内网和外网用户访问 area_eth2 区域的 WEB 服务器。

为实现上述访问控制需求与策略，配置命令具体如下：

1）设定物理接口 eth1 和 eth2 的 IP 地址。

#network interface eth1 ip add 192.168.100.140 mask 255.255.255.0

#network interface eth2 ip add 172.16.1.1 mask 255.255.255.0

2）添加 VLAN 虚接口，设定 VLAN 的 IP 地址，再选择相应的物理接口加入到已添加的 VLAN 中。

#network vlan add range 1,2

#network interface vlan.0001 ip add 10.10.10.1 mask 255.255.255.0

#network interface vlan.0002 ip add 10.10.11.1 mask 255.255.255.0

#network interface eth0 switchport trunk allowed-vlan 1,2 native-vlan 1 encapsulationdotlq

3）定义主机、子网地址对象。

#define host add name 172.16.1.3 ipaddr 172.16.1.3

#define host add name 192.168.100.143 ipaddr 192.168.100.143

#define host add name doc_server ipaddr 10.10.10.3

#define subnet add name rd_group ipaddr 10.10.11.0 mask 255.255.255.0 except ‘10.10.11.2 10.10.11.3’

4）设置区域对象的缺省访问权限：area_eth0、area_eth2 为禁止访问，area_eth1 为允许访问（缺省权限，无需再设定）。

#define area modify name area_eth0 access off（不允许访问内网）

#define area modify name area_eth2 access off（不允许访问内网）

5）定义地址转换规则。

定义源地址转换规则，使得内网用户能够访问外网。

#nat policy add dstarea area_eth1 trans_src 192.168.100.140

定义目的地址转换规则，使得内网文档组以及外网用户都可以访问 area_eth2 区域的 WEB 服务器。

#nat policy add orig_dst 192.168.100.143 orig_service HTTP trans_dst 172.16.1.3

6）定义访问控制规则。

允许内网和外网用户均可以访问 WEB 服务器

#firewall policy add action accept dstarea area_eth2 dst 172.16.1.3 service HTTP

允许项目组领导访问外网，禁止项目组普通员工访问外网

#firewall policy add action deny srcarea area_eth0 srcvlan vlan.0002 src rd_group dstarea area_eth0 service HTTP

上述配置过程有几点需要注意：

1）目的地址需要选择 WEB 服务器的真实 IP 地址，因为防火墙要先对数据包进行目的地址转换处理，当内网用户利用 http://192.168.100.143 访问 Web 服

务器时，由于符合 NAT 目的地址转换规则，所以数据包的目的地址将被转换为172.16.1.3。然后才进行访问规则查询，此时只有设定为WEB 服务器的真实IP 地址才能达到内网用户访问 WEB 服务器的目的。

2）定义目的地址转换规则时，不能选择目的区域与目的 VLAN。

该实例体现了防火墙在不同网络安全域之间的访问控制作用，这些网络安全域表现为不同的网络子网，由不同权限的人员使用。而子网的权限是根据企业工作需求制定，通过防火墙实施这些权限的控制与约束，达到安全防护的目的。

5.3.2 电子政务系统访问控制实例

电子政务是指政府在其管理和服务职能中运用现代信息和通信技术，实现政府组织结构和工作流程的重组优化，提高政务处理效率的一种重要手段。电子政务的特殊性，决定了此类应用在信息安全方面比其他应用要求更为严格。

公文处理是电子政务的核心功能之一，对于那些敏感的政府部门来说，保证公文的安全是电子政务系统的首要目标。所采用的强制访问控制策略基本上是遵循 BLP 模型的框架和原则。例如，通常对系统中的主客体，即电子政务系统中的政府职员和公文，附加安全级的标记。安全级（或称安全标记）由密级和部门两部分构成。在实际应用中，密级常定义为“公开、秘密、机密、绝密”四个不同的级别，以此来标识各种公文的不同保密程度，每位政府职员可以阅读的公文的密级具有严格的规定，如秘密级的职员绝不可阅读机密和绝密公文，即使该公文被有意或无意放在职员的个人文件夹中，也不可违反上述规则。这种安全需求常用强制访问控制策略来表达和实现。

然而，电子政务系统是一个实用性的系统，它服务于政府的办公需求。公文的处理包括起草、审核、修改、再审核、再修改直至上级领导核准签发等一系列环节，整个过程中公文需要在不同职务、不同级别的工作人员之间进行复杂地流转。因此简单的一个“不能上读，不能下写”的原则是不能满足其访问控制需求的。它往往需要采取基于任务的访问控制策略，即根据每一个主体的工作职责或工作任务来分配和控制其对客体的访问权限。但是一旦公文签发了，或者从外单位收到了公文，那么系统中哪些主体能阅读，哪些主体不能阅读，则需根据主、客体的安全级来决定，对这种读权限的控制，电子政务系统基本上是遵循 BLP 模型的安全策略，即仅当主体的部门和客体一致，且主体密级不低于客体密级时，

方进一步判断主体对客体是否具有访问权限。表 5.1 和 5.2 给出了主体和客体的数据库表设计的实例。其中密级用整数 1、2、3、4 分别表示公开、秘密、机密、绝密四个级别。

表 5.1 职员表

序号	字段名	类型	备注
1	ID	Integer	职员编号，主关键字
2	Name	Char(10)	职员姓名
3	DeptID	Integer	所属部门编号
4	SecLevel	Integer	密级，1-4 分别表示公开到绝密四个级别
5	Title	Char(10)	职务
…	…	…	…

表 5.2 公文表

序号	字段名	类型	备注
1	ID	Integer	公文编号，主关键字
2	Name	Char(50)	公文名称
3	DeptID	Integer	所属部门编号
4	SecLevel	Integer	密级，1-4 分别表示公开到绝密四个级别
5	DocURL	VarChar(200)	公文正文链接
…	…	…	…

对应的数据库 DDL 语句如下：

```
CREATE TABLE EMPLOYEE
    (ID  int  PRIMARY  KEY,
    Name  char(10),
    DeptID  int,
    SecLevel  int,
    Title  char(20),
    …)
```

```
CREATE TABLE DOCUMENT
    (ID  int  PRIMARY  KEY,
    Name  char(50),
    DeptID  int,
    SecLevel  int,
    DocURL  varchar(200),
    …)
```

例如张三是信访部门的副处级干部，他可以阅读本部门机密及以下级别的文件。假设信访部门编号为 16，用 3 表示机密级别，则可使用以下 DML 语句进行设置：

```
INSERT  INTO  EMPLOYEE(ID, Name, DeptID, SecLevel, Title)
    VALUES(1, '张三', 16, 3, '副处级')
```

假设信访部门有一名为“举报人资料”的绝密公文，则可使用以下 DML 语句进行设置：

```
INSERT  INTO  DOCUMENT(ID, Name, DeptID, SecLevel)
    VALUES(2, '举报人资料', 16, 4)
```

因为 3＜4,根据强制访问控制规则，张三无法阅读此绝密公文。

假设信访部门有一名为“财务预算”的机密公文，则可使用以下 DML 语句进行设置：

```
INSERT  INTO  DOCUMENT(ID, Name, DeptID, SecLevel)
    VALUES(3, '财务预算', 16, 3)
```

根据强制访问控制规则，张三的安全级高于此公文的安全级，因此张三可读访问该公文。若系统中还有其他访问控制机制（如自主访问控制或基于角色的访问控制），则须根据应用中的其他访问控制策略（如自主或基于角色的访问控制策略）进行进一步的判断。只有通过了系统中所有的访问控制检查，张三才能访问该公文。

5.3.3 医院管理信息系统访问控制实例

医院管理信息系统是指利用计算机软硬件技术、网络通信技术等现代化手

段，对医院及其所属各部门的人流、物流、财流进行综合管理，对在医疗活动各阶段产生的数据进行采集、储存、处理、提取、传输、汇总、加工生成各种信息，从而为医院的整体运行提供全面的、自动化的管理及各种服务的信息系统。

处方管理是医院管理信息系统的重要功能之一，根据我国《医疗机构药事管理暂行规定》和处方管理条例的有关规定,医师和药学人员在药物临床应用中必须遵循安全、有效、经济方便的原则，对临床使用药品进行权限划分，对医师使用特殊药物进行权限管理，不同级别的医师具有不同的处方权限。例如常有如下具体规定：

住院医师处方权限一线用药，通常为非限制使用的抗菌类药物；主治医师处方权限除了一线用药外，还包括二线用药，通常为限制性使用的抗菌类药物；副主任医师、主任医师处方权限为一、二线用药和三线用药（特殊使用抗菌药物）。

上述访问控制需求比较适合使用基于角色的访问控制来实现，根据我国医疗单位的实际情况，可将医生按职称划分为实习医师、住院医师、主治医师、副主任医师、主任医师等不同角色，并根据药品分级管理办法，为每种角色设置不同的药品处方权限。

为了实现医院管理信息系统中的处方控制，可定义医师表、角色表、处方药品表分别如表 5.3，5.4，5.5 所示(省略部分可根据应用需求添加)：

表 5.3 医师表

序号	字段名	类型	备注
1	ID	Char(6)	医师编号，主关键字
2	Name	Varchar(20)	医师姓名
3	RoleID	Integer	医师角色编号
…	…	…	…

表 5.4 处方药品表

序号	字段名	类型	备注
1	ID	Char(6)	药品编号，主关键字
2	Name	Varchar(255)	药品名称
3	Alias	Varchar(100)	别名

4	Type	Integer	类别值，用 1，2，4 分别表示 1-3 线药品
4	ProdID	Integer	生产商编号
…	…	…	…

表 5.5 角色表

序号	字段名	类型	备注
1	ID	Char(4)	角色编号，主关键字
2	Name	Varchar(20)	角色名称
3	Types	Integer	可开具处方药类别值之和(掩码)，如 3=1+2，可开具 1，2 类药品
…	…	…	…

对应的数据库 DDL 语句如下：

```
CREATE  TABLE  PHYSICIAN
    (ID  char(6)  PRIMARY  KEY,
    Name  varchar(20),
    RoleID  int,
    …)
CREATE  TABLE  MEDICINE
    (ID  char(6)  PRIMARY  KEY,
    Name  varchar(255),
    Alias  varchar(100),
    Type  int,
    ProdID  int,
    …)
CREATE  TABLE  ROLE
    (ID  char(4)  PRIMARY  KEY,
    Name  varchar(50),
    Types  int,
```

…)

如“注射用阿莫西林钠克拉维酸钾（又名强力阿莫仙）”属于二线用药，“注射用头孢曲松钠/配舒巴坦钠（又名新君必治）”属于三线用药，可在处方药品表中通过以下 DML 语句设置：

```
INSERT  INTO  MEDICINE(ID, Name, Alias, Type)
    VALUES(‘M00201’, ‘注射用阿莫西林钠克拉维酸钾’, ’强力阿莫仙’, 2)
INSERT  INTO  MEDICINE(ID, Name, Alias, Type)
    VALUES(‘M00202’, ‘注射用头孢曲松钠/配舒巴坦钠’, ’新君必治’, 4)
```

角色“主治医师”可开具 1、2 线药品处方，角色“副主任医生”可开具 1、2、3 线处方，可在医师表中通过以下 DML 语句设置：

```
INSERT  INTO  ROLE(ID, Name, Types)
    VALUES(‘R002’, ‘主治医师’, 3)
INSERT  INTO  ROLE(ID, Name, Types)
    VALUES(‘R003’, ‘副主任医师’, 7)
```

假设医生张三职称为主治医师，则其记录可通过以下 DML 语句设置：

```
INSERT  INTO  PHYSICIAN(ID, Name, RoleID)
    VALUES(‘P00100’, ‘张三’, ‘R002’)
```

医院管理信息系统中访问控制策略的决策和实施应该通过专门的模块来实现，并应在技术上保证模块不可被篡改或绕过。决策模块接受主体对客体的请求，根据访问控制策略库判断请求是否允许，将判断结果传递给实施模块完成访问的具体控制。可见，决策模块是访问控制系统中的安全核心组件，其决策判定的处理流程如图 5.9 所示：

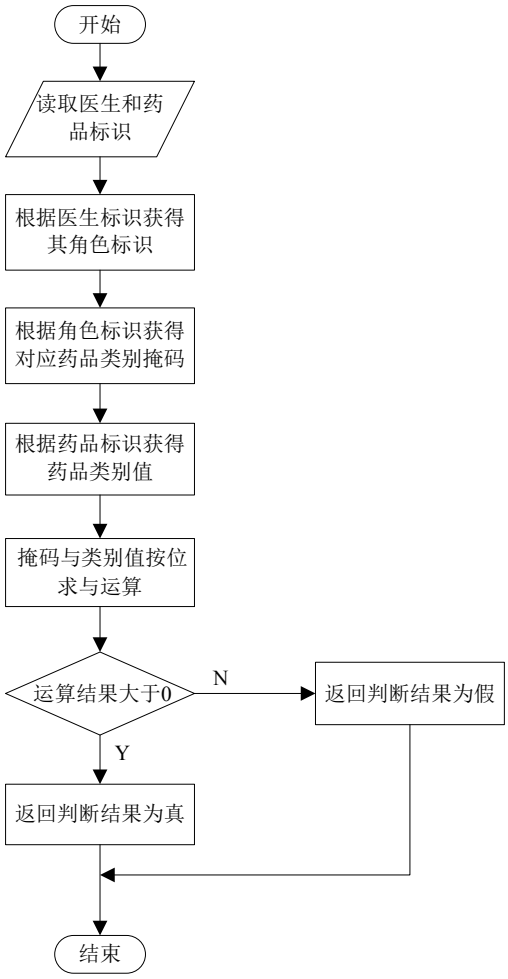


图 5.9 访问决策模块处理流程

假设主治医师张三请求开具含一线药品息斯敏的处方。根据以上处理流程，系统首先查询医师表，得知张三的角色为主治医师；然后查询角色表，得知主治医师对应的药品类别掩码为 3；接下来查询处方药品表，可知药品息斯敏的类别为 1，3 和 1 按位与运算的结果为 1＞0，说明可以开具此处方。假设张三请求开具含药品新君必治的处方。查询处方药品表可知新君必治的类别为 4，掩码 3 和类别 4 按位与运算的结果为 0，说明不可开具此处方。

习题五

1. 如果将网络中的所有资源都看做网络管理员所有，那么防火墙实施的访问控制策略属于自主访问控制策略还是强制访问控制策略？

2. 某人在互联网上申请了个人主页空间，希望只允许好友访问他的个人主页，在访问控制方面应采用何种访问控制策略？

3. 根据 5.x 访问控制决策流程，分析副主任医师李四能否开据含有强力阿莫仙的处方。

第 6 章 多域访问控制技术

近年来，随着Internet和一些分布式系统支撑技术的飞速发展和普遍应用，人们开发了越来越多大规模的分布式系统。在国防军事联盟合作、医疗卫生保健系统、电子商务、电子政务和其它领域，已经或正在开发一些这样的系统。在这些系统中，往往存在着大量的跨越组织边界的资源共享和信息交换。对比单个组织内的访问控制问题，多域之间授权和权限控制问题的复杂性和严重性大大增强。如何解决这些问题，在很大程度上决定了这些系统最终能否获得成功。

基于角色映射的多域安全互操作技术，它针对两个都实施基于角色访问控制的管理域情况，通过建立外域和本域间的角色映射关系，将本域角色指派给外域角色实现跨域的安全共享访问。

对跨越多个管理域的系统而言，动态结盟环境下基于角色访问控制技术通过发布角色委托，将角色传递委托给不同信任域内的其他角色，从而实现结盟组织内部之间的资源共享。

多虚拟组织结盟的访问控制技术从软件体系结构的模式出发，利用现有的中间件技术，设计和实现了一个跨域的安全互操作结盟基础设施。它支持多个组织共享对象同时又保持各组织对本地资源的自治权。

在大规模的 Internet 和跨越多域边界的应用中，结合 PKI 的跨域基于角色访问控制技术利用用户角色证书和支持角色层次的证书链，结合 X.509 证书扩展项，实现基于角色和支持角色层次的授权管理。

6.1 基于角色映射的多域安全互操作

6.1.1 多域安全互操作的应用背景

我们首先来看如下的应用实例：A公司和B研究所为了某个研究项目建立起合作关系。为了该科研项目的顺利实施，A公司和B研究所之间需要进行相应的共享协作。A公司和B研究所是不同的单位，他们各自拥有由多台主机、路由设备和网络组成的信息管理系统，并且有各自独立的资源保护安全策略和相应的安全管理员。对此，我们称A公司和B研究所是两个不同的“安全管理域”，下文一般简称为“安全域”或“域”。

假设在上例中，A公司和B研究所在各自的域内均采用基于角色访问控制（RBAC）实施对用户的授权管理。即系统根据不同的职责、功能或者岗位设置相应的角色，对每一个角色分配不同的操作权限，用户通过所分配的角色获得相应的权限，实现对信息资源的访问。设A公司用户“张三”具有 “部门经理”的角色，且要求在B中查询有关合作项目的进展情况，于是出现了跨越安全域的访问。在此情形下，我们称B研究所的安全域为“本域”，而对应的A公司的安全域为“外域”。当然“本域”和“外域”是相对的概念，若B研究所用户提出对A公司域内资源的访问，则称B研究所为“外域”，而A公司就称为“本域”。

当“张三”提出对B研究所域内资源的访问时，B的安全策略不能识别A域中的用户“张三”，访问不能够进行。因此，两个域之间在安全策略上须达成某种共识，以便在两者之间建立安全性会话。其核心问题就是在本域中，对外域角色做出一个适当的评价。一种最基本的情况是在两个域之间建立一种缺省的安全策略，以此提供基本的安全性。比如，可以将外域的角色统一作为本域中最底层的角色来看待，给外域角色提供最基本的访问权限。但是，这样的方式缺乏灵活性。

6.1.2 角色映射技术

角色映射技术是指在两个域之间定义角色映射关系，使外域角色能够转换成本域角色，从而赋予外域角色对本域资源的访问权限。为了说明角色映射的含义，看下面的一个例子。

用H_0表示本域中的角色集合，H_1表示外域中的角色集合。用H_1H_0表示从H_1到H_0的角色映射关系。用离散数学术语来表达，从H_1到H_0的角色映射关系是笛卡儿积$H_1\times H_0$的子集，即表示为$H_1H_0\subseteq H_1\times H_0$。其中映射关系$H_1H_0$中的任意一个序偶$(r_1，r_0)$，称为从外域角色$r_1$到本域角色$r_0$的一个角色映射，也称为角色$r_1$关联到本域角色$r_0$。它的含义就是将本域中的角色$r_0$分配给外域中角色$r_1$，使得外域中具有角色$r_1$的用户在本域中具有角色$r_0$的权限，从而可以实现跨域的安全访问。对此，记作$r_1 \rightarrow r_0$。

如图6.1所示，A公司和B研究所分别代表两个域，其中定义了从A公司到B研究所的角色映射关系：部门经理$_A\rightarrow$研究员$_B$，员工$_A\rightarrow$职员$_B$，项目经理$_A\rightarrow_{NT}$工程师$_B$。

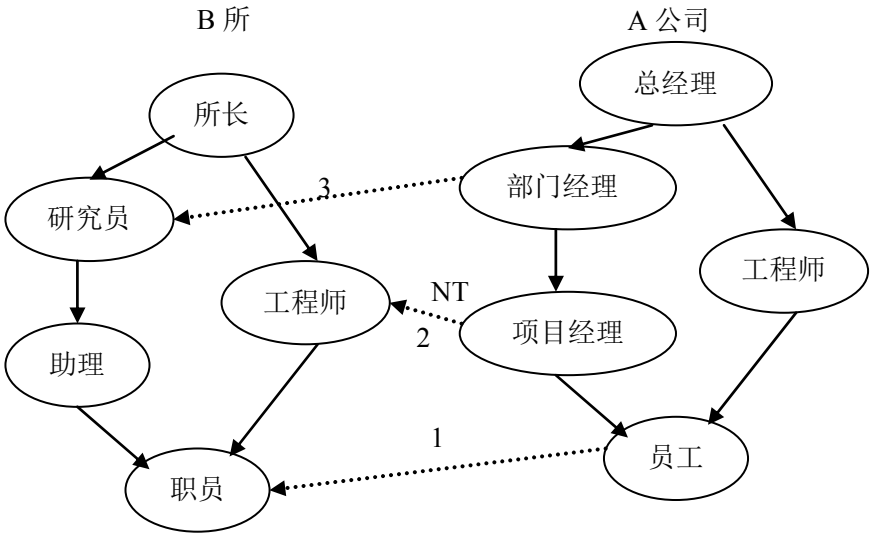


图6.1　域间角色转换图

图6. 1角色映射关系中定义了两种不同的关联:可传递关联和非传递关联，分别用符号$\rightarrow$ 和$\rightarrow_{NT}$表示。

所谓可传递关联，是指如果外域的角色x关联到本域角色y，则角色x以及在外域内角色x的所有上级角色都可以通过这个关联映射到本域角色y。如图6.1所示，部门经理$_A\rightarrow$研究员$_B$标记为可传递关联，则A中部门经理任意的上级角色，比如总经理角色，在B中同样能够映射到研究员角色。在B中，助理和职员是研

究员的下级角色，所以A中的总经理和部门经理角色也将同时具有助理和职员角色的权限。

所谓非传递关联，是指如果外域的角色x关联到本域角色y，则角色x的所有上级角色都不能通过这个关联映射到本域角色y。如图6.1所示，项目经理$_A \rightarrow_{NT}$工程师$_B$标记为非传递关联，在A中项目经理的所有上层角色，比如总经理角色和部门经理角色，在B中都不能通过该关联映射到B中工程师角色。在A中只有项目经理角色才能获得B中工程师角色。当然，职员是工程师的下层角色，所以A中项目经理角色也将同时获得职员角色的权限。

要注意的是，不论是可传递关联$r_1 \rightarrow r_0$ 还是非传递关联$r_1 \rightarrow_{NT} r_0$，实际上都是为外域角色r_1 在本域中指定了一个角色子集与之对应，该子集中包含角色r_0 及r_0 的所有下层角色。只不过前者可以将该对应关系传递给r_1 的所有上级角色，而后者则不允许。

定义非传递关联是为了满足以下情形的需求，即如果本域管理员想赋予某一个外域角色特定的权限，而且希望仅有该外域角色可以拥有这个权限，他的上级角色不能继承这一权限。这样本域管理员在没有权力更改外域角色层次关系的情形下，按照自己的需要对外域角色的权限进行了控制。

6.1.3 建立角色映射的安全策略

在两个域之间，通过角色映射技术，使得外域角色关联到本域的某些角色，从而可以实现跨越安全域的访问操作。如何建立外域角色到本域角色的关联，则需要本域管理员制定相应的安全策略。总体上来讲，制定的安全策略可以分成以下三类。

第一种是缺省策略，这种策略在外域角色和本域角色之间建立最小数目的关联。在这一策略下，所有的外域角色都被映射到同一个本域角色。

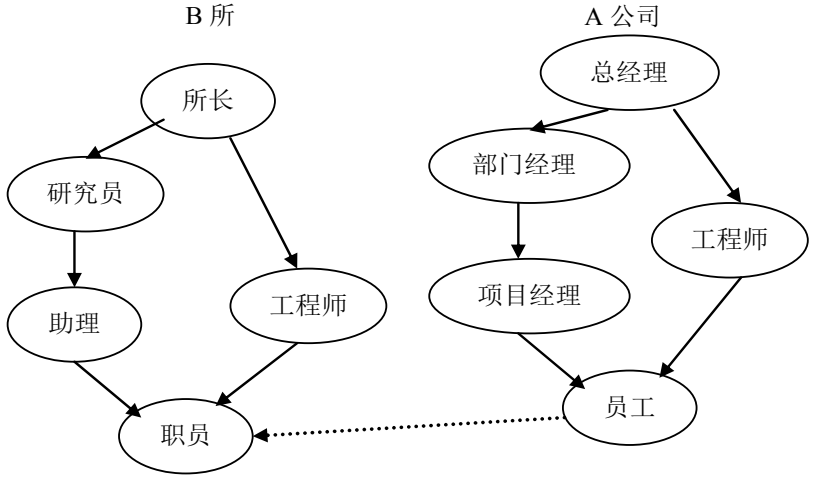


图6.2　缺省策略示意图

如图6.2所示，员工$_A \rightarrow$职员$_B$标记为传递关联，在A角色层次中员工的所有上层角色，比如总经理角色、部门经理角色、项目经理角色和工程师角色都可以通过该关联获得B中的职员角色。这体现了一种缺省原则。

这种方案最易于建立，但也最不灵活。这是因为所有外域角色都被当作同一个本域角色看待。然而这种方案也很重要，因为它提供了最基本的互操作性，并且常常和下文的原则一起协同工作。

第二种是明确策略，这种策略是指本域管理员明确地将每一个外域角色直接映射到本域的一个角色。实现该策略的有效途径就是用非传递关联为外域中每个角色指定本域中一个角色子集与之对应。用数学的语言表达就是定义了一个从外域角色集到本域角色集的幂集的函数，即f：$H_1 \rightarrow 2^{H0}$。

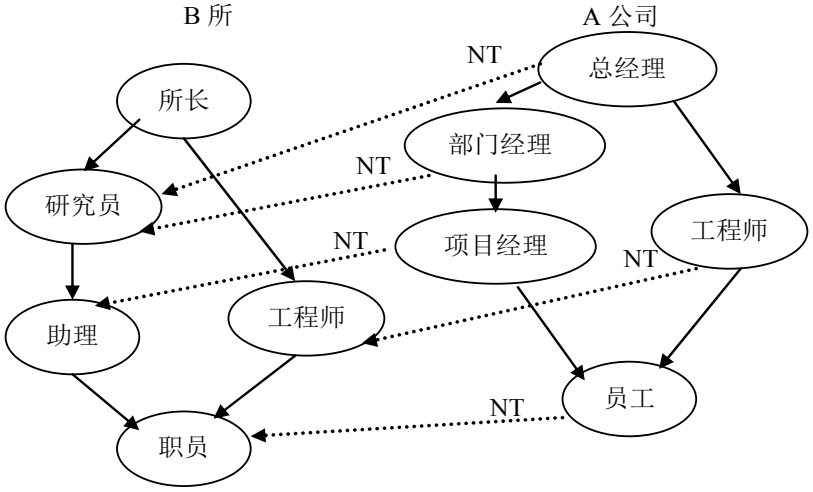


图6.3　明确策略示意图

如图6.3所示，在A角色层次中，使用非传递关联为每一个角色明确的指定一个角色映射关系。这种方案最灵活，但是也最难建立。因为它强制要求管理员为外域中每一个角色都建立一个关联，增加了创建和维护关联的复杂性。

第三种是部分明确策略，这种策略是指本域管理员为特定的外域角色在本域层次中指明转换角色，外域中的剩余角色可以通过传递关联，建立到本域角色层次的映射关系。如图6.1所示，角色关联“员工$_A$→职员$_B$”体现了缺省策略，角色关联“项目经理$_A$→$_{NT}$工程师$_B$”体现了明确策略；其他的角色都可以通过关联“员工$_A$→职员$_B$”和“部门经理$_A$→研究员$_B$”建立到本域角色的映射关系。

这种方案体现了角色映射技术真正意义上的灵活性，也有利于管理员进行管理。

总体上而言，上述的安全策略是一般性的原则，在具体应用时还需要根据各种特殊情况进行相应的处理。比如，若本域内没有合适的角色指派给外域角色，本域管理员可以为其增加一个新的角色，来供其使用；若外域角色无层次关系，则无法使用部分明确策略，只能为每一个需要进行跨域访问的外域角色创建到本域角色的关联；若本域角色无层次关系，则可能根据实际需要，为外域角色创建到本域角色的多个关联。

6.1.4 角色映射的维护

在建立关联以后，外域角色集和本域角色集之间就具有了相应的映射关系。但当外域或本域角色层次发生改变时，就可能会影响到已经建立的映射关系。下面从角色的增加和删除两方面对现存关联的影响进行分析。

第一种情况是角色的增加。本域角色的增加不会影响到已经建立起来的外域到本域角色的映射关系。例如，假设在图6.1中，在B的助理角色之下，职员角色之上，增加一个新的资深职员角色。则角色映射关系“部门经理$_A$→研究员$_B$”使得A的部门经理角色除可以获得B的研究员角色之外，通过B的研究员角色还可继承得到助理角色、资深职员角色和职员角色。这种继承关系在本域角色层次发

生变化时自动完成。外域的角色层次添加新的角色也不会影响到现存的关联关系。外域新添加的角色也可以通过传递关联或缺省策略映射到本域角色或由本域管理员为其增加新的关联。

第二种情况是角色的删除。当在本域角色层次中，删除一个角色，如果不存在到达该角色的关联时，对现有的关联不会发生影响。如果存在到达本角色的关联时，则应该对关联进行维护。首先删除每一个到达该角色的关联，然后添加一系列新的关联，这些关联到达被删除角色的所有下一级角色。可传递关联被一系列可传递关联替代，非传递关联被一系列非传递关联替代。

例如，设外域角色层次和本域角色层次原有的关联如图6.4所示。

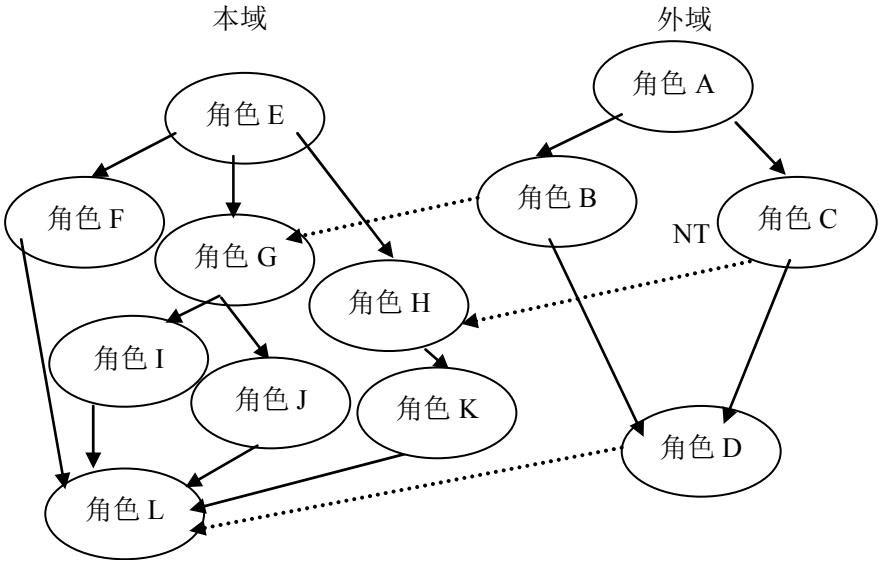


图6.4　角色删除前域间角色转换图

如果在本域中删除角色G和角色H以后，则现存的外域角色到角色G和角色H的关联都需要发生改变。改变后的结果如图6.5所示。

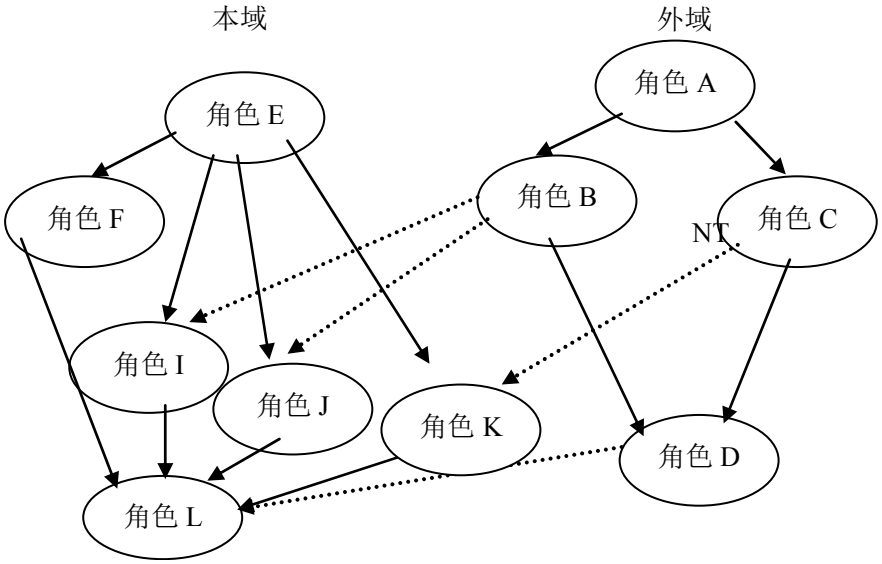


图6.5　本域角色删除后域间角色转换图

其中删除外域角色到角色G和角色H的关联，然后添加一系列新的关联，并且分别到达角色G和角色H的下一级角色，其中原有的可传递关联被可传递关联所代替，非传递关联被非传递关联所替代。

对于外域的角色删除也需要区分情况进行不同的处理。若开始于被删除角色的关联是可传递关联，则删除开始于该角色的这个可传递关联，添加一系列新的可传递关联，新的关联开始于被删除角色的上一级角色，并指向被删除角色到达的角色。例如，设外域角色层次和本域角色层次原有的关联如图6.5所示，如果在外域中删除角色B以后，则现存的外域角色B到本域角色I和角色J的关联都需要发生改变。改变后的结果如图6.6所示。

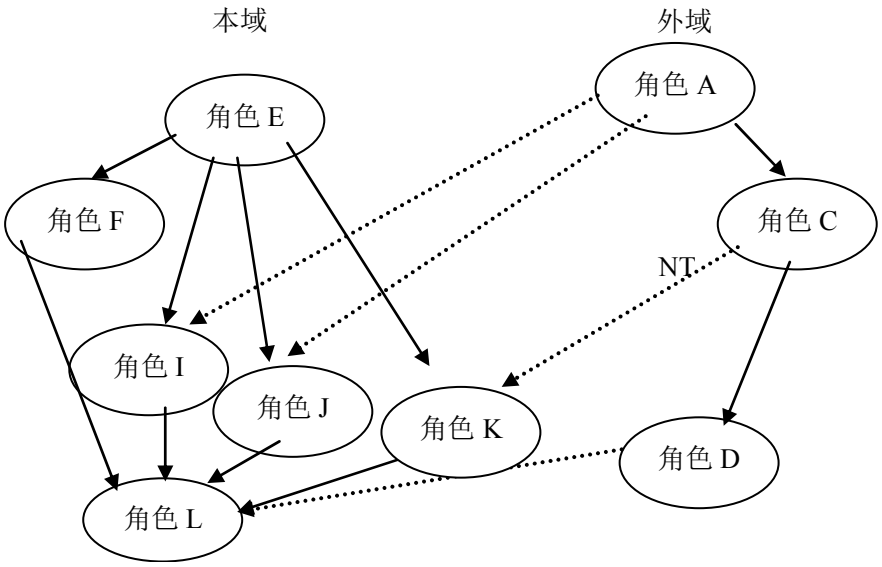


图6.6　外域角色删除后域间角色转换图

若开始于该角色的关联是非传递关联，则直接删除该关联即可。

6.1.5 角色映射的安全性分析

通过外域到本域的角色映射方法，能够实现外域对本域进行访问的安全目标。然而角色映射也打破了本域的管理界限，允许外域角色对本域的资源进行访问，从而带来了一定的安全风险。下文对角色映射带来的安全风险进行分析。

1.　多域穿梭问题

当一个用户试图访问另一个域中的资源时，它必须穿过域边界，这就是域穿梭。如果一个主体能够进行多次的域穿梭，将会产生安全隐患，因为它可能带来“渗透”和“隐蔽提升”。

所谓渗透就是指，一个主体通过借助于另一个域，企图间接地访问其他的域。比如，若 D_2 和域 D_1 建立了色映射，域 D_1 和 D_0 也建立了角色映射，然而此时并不意味 D_2 和 D_0 也需要建立角色映射。但是，来自于 D_2 的用户可以首先进入 D_1，然后通过 D_1 和 D_0 的角色映射关系，渗透进入 D_0。因此，角色映射应该防止渗透。

所谓隐蔽提升是指主体能够通过多次穿越域边界，以比起始角色更高级的角色返回到其起始域。也就是说，一个用户通过多域穿梭隐蔽地提升它在角色层次中的级别。

为了避免“渗透”和“隐蔽提升”，角色映射可限定只对单次的域穿梭有效。

每次提出访问请求的主体，必须在其证书中加入起始域和角色名称。在主体进行域穿梭时，系统必须进行严格验证，保证主体最多只能够进行一次跨越域的角色映射。

2. 关联冲突问题

关联冲突问题是指可能外域中一个下级角色关联到本域的角色x，而它的上级角色关联到本域的角色y，而x是比y更高的角色。

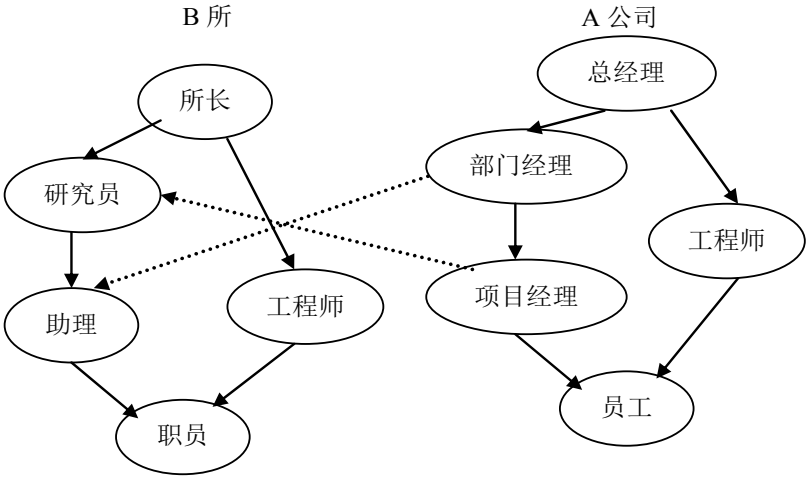


图6.7 关联冲突示意图

如图6.7所示，部门经理$_A$→助理$_B$标记为可传递关联，如果添加新的关联项目经理$_A$→研究员$_B$，则项目经理$_A$将被映射到研究员$_B$。那么这两个关联相互冲突了，因为在A中部门经理角色要比项目经理角色高，而映射到B以后，助理角色要比研究员角色低。系统安全策略是不允许建立这样关联的。有一种解决冲突的方法就是给予外域角色在本域角色层次中所允许的最高转换角色，因此，如图6.8所示，部门经理角色也将会转换成研究员角色。

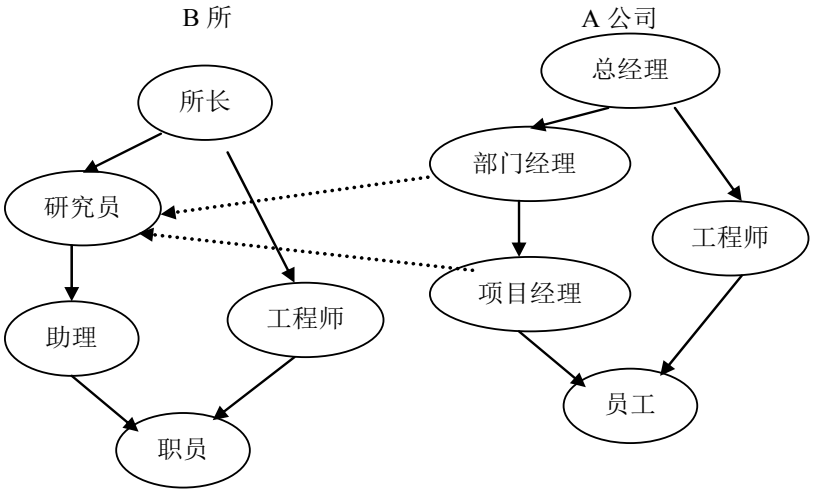


图6.8 解决冲突关联示意图

3. 角色层次的安全披露问题

一般情况下，角色都是根据组织内不同的职责、功能或者岗位而设置的。因此，系统内角色层次从很大程度上反映了系统整体安全策略，它至少表明了系统

设置有哪些角色和角色之间的支配关系。如果要在外域和本域间建立关联，则必须将外域中所有的角色层次信息披露给本域的安全管理员，那么这意味着角色层次所代表的安全信息暴露给了本域，这将是一个值得关注的问题。

域间的角色映射为不同管理域之间的互访、互操作提供了一种安全的实现方法。该方法特别适合于组织机构和业务内容相近，需长期合作且信息共享需求较大的机构，如高等院校与高等院校之间，金融机构与金融机构之间、医院与医院之间等。

6.2 动态结盟环境下基于角色访问控制

6.2.1 动态结盟环境下的应用背景

若干个组织或机构为了共同的利益或目标，需要相互协作共同去完成某项任务或做某方面的工作，在执行这项任务或工作期间，他们需要资源共享为其他方提供相关的服务。这种合作关系多是临时的、动态的，随时可以结盟，也随时可以解除。

对于这种应用场景，采用角色映射的方法就显得过于复杂，且不够灵活。本节介绍的动态结盟环境下基于角色的访问控制（dRBAC）可较好地解决这一问题。这一方法的基本思想是由被访问方的实体对访问方的实体进行授权，这种授权在两方之间主要是基于角色来进行。而在内部，角色与用户之间的指派，则由各方的资源管理者授予。因此，一项访问被允许之前，往往要经过一系列的授权组成的授权链。

例如设由A、B和C三个国家的军队组成一个军事联盟，正在举行联合军事演习。A国方面的视频设备获得一些视频信息，A国军队指定史密斯将军(Smith General)既可以观察视频信息又可以将这个权力委托给参加军事演习的B国和C国军队，实现动态结盟环境下的资源共享。因为这些军队通过不安全的网络进行连接，所以对拥有视频信息主机的访问必须经过A国的授权。

在共享资源的授权过程中，有时需要体现对数值属性的支持，使得授权对象不同而访问级别也不相同。比如，A国军官收到的视频数据比其他国家的军官可能具有更高的图像分辨率和更短的时间延迟。

针对这些安全需求，动态结盟环境下的基于角色访问控制通过引入角色委托、第三方委托和值属性等访问控制机制，实现这些安全目标。

6.2.2 dRBAC 基本组件

1. 实体

在基于角色的访问控制中，通常将系统保护的资源定义为客体，而将提出访问请求的用户或者用户进程称为主体。在 dRBAC 中，所有的资源和主体都被看作“实体”。它既可以代表个体的集合，也可以是具体的个体，例如，“A 国军队”可以是一个实体，它代表所有军官和士兵所组成的群体；而“史密斯将军”也是一个实体，它代表一个具体的军官；并且“视频设备”也是一个实体，它代表捕获视频信息的主机。

在动态结盟环境下，通过公/私钥对来唯一的标识一个实体。根据应用的需求，该公/私钥对在全局范围内是唯一的。

2. 角色

dRBAC 的核心组件是角色。系统将受保护资源的访问权限都指派给相应的角色，角色代表了对系统资源的访问权限。从表达形式上看，角色是由左边字符串、“.”连接符、右边字符串所组成的一个字符串。其中左边字符串表示角色所在的实体名字，右边字符串表示角色名字。比如“Camera.View”就表示在视频设备实体（Camera）内定义的一个角色“View”，它代表了对视频信息的访问权限。每一个提出视频信息访问请求的用户都必须证明其具有“Camera.View”角色。

实体和角色是一对多的关系；一个实体内部可以定义多个角色，一个角色只能属于某一个实体。比如角色“Camera.View”只属于“Camera”实体，“Camera”实体就称为角色“Camera.View”的宿主实体。“Camera”实体内还可以定义其他的角色，比如“Camera.Init”、“Camera.Update”等角色。

dRBAC 通过确定用户是否拥有对资源访问所必需的角色做出访问控制决策。dRBAC 试图回答的关键问题是：“用户 P 拥有角色 R 吗”？

3. 委托

角色通过委托（Delegation）授权给实体。从一般意义上而言，委托具有托付别人做某些事情的含义。例如“Alice 委托给 Bob 一些权利”意思就是：Alice 以自己的名义授予 Bob 行使某些任务的权限。

在 dRBAC 中，委托的一般形式是：

[主体 → 客体] 发布者

其中，主体可以是一个角色或者实体，客体是一个角色，发布者是一个实体，“→”读作是“拥有角色”。发布者通过创建委托，以自己的名义将客体角色授予给主体。

dRBAC 的委托具体可以分为下列三种类型：

1) 客体委托

客体委托（Object Delegation）的客体必须是一个角色，而不能是实体。因为一个实体不能将自己的身份委托给别人。主体既可以是一个角色也可以是一个实体。

如果主体是角色，那么委托的形式就是：

[A.role1 → B.role2]B

它的含义就是实体 B 将角色 B.role2 指派给 A.role1，使得具有角色 A.role1 的用户获得了角色 B.role2 的权限。这个委托类似于 6.1.2 节描述的角色映射技术，可以看作 A.role1 被关联到 B.role2。

因为任何实体都可以分发自己内部定义的角色，因此主体 A 还可以通过将 A.role1 委托给其他的主体，而将 B.role2 的角色继续向下传递委托。

如果主体是实体，那么委托的形式就是：

[A → $B.role_2$]B

它的含义是实体 B 将角色指派给主体，使得主体 A 具有角色 $B.role_2$ 的权限。此种委托类似于 RBAC 中的用户角色指派，即将角色 $B.role_2$ 指派给主体 A。

在这种情况下，主体 A 不能再进一步将角色 $B.role_2$ 委托给别的主体。

2) 分配委托

能够将某个角色授予给相应的主体或角色的能力称为该角色的“分配权”。在分配委托（Assignment Delegation）中，发布者实体将某个角色的分配权指派给某个主体。在客体后面加一个“’”表示主体对客体的分配权，分配委托的形式如下：

[A → B.role$_2$’]B

它的含义是实体 B 将角色 B.role$_2$ 分配权指派给主体 A，使得主体 A 可以将角色 B.role$_2$ 指派给别的主体。基于自主访问控制的策略，主体 A 只能拥有自身内角色的分配权，并没有实体 B 内角色的分配权。但是主体 B 通过发布分配委托，授予主体 A 对角色 B.role$_2$ 的分配权。

可以把主体 A 看作是角色 B.role$_2$ 的管理角色，主体 A 能够将角色 B.role$_2$ 指派给其他的主体或角色。

3) 第三方委托

第三方委托(Third-Party Delegation)是在分配委托的基础上进行的，主体通过分配委托获得某客体角色的分配权后，可以将该客体角色委托给别的主体或角色。

下面举例说明以上这些概念：

（1） [D → B.b ’] B

该委托是一个分配委托，该委托表明了主体 B 将角色 B.b 的分配权指派给 D。主体 D 获得了角色 B.b 的分配权，可以将角色指派给其他的主体或角色。

（2） [C → B.b ’] D

这是一个第三方委托，也是一个分配委托，主体 D 将通过（1）步获得的角色 B.b 的分配权指派给主体 C。并使主体 C 也获得了角色 B.b 的分配权。

（3） [A → B.b] C

这也是一个第三方委托，但不是一个分配委托，它是一个客体委托，主体 C 只是将角色 B.b 指派给主体 A，A 并未获得角色 B.b 的分配权。

通常，客体的宿主实体和发布者是同一个实体，这种情况下的委托就称作是“自我证明”的(“self-certifying” delegation)。任何实体都有权将自己内部定义的角色指派给别的主体，所以“自我证明”委托都是有效的。这体现了一种自主访问控制的安全策略。上述例子中的委托（1）是一个自我证明的委托。

与之相对应的是第三方委托，在第三方委托中，客体的宿主实体和发布者不是同一个实体。第三方委托的发布者主体必须证明自己获得了客体的分配权，如果没有获得客体的分配权，第三方委托是无效的。上述例子中的委托（2）、（3）都是第三方委托。委托（2）的发布者实体是 D，而客体是角色 B.b，因为在委托（1）中，主体 B 将角色 B.b 的分配权指派给 D，所以委托（2）才是有效的。正是因为有了委托（1）的证明，所以委托（2）和（3）才是有效的。否则只有委托（2）和（3）是无效的。

上述例子中的委托（1）是一个自我证明的委托，将委托（1）（2）（3）组合起来就构成了一个授予实体 A 角色 B.b 有效的委托链，实体 A 具有了角色 B.b 的访问权限。委托链都必须由自我证明的委托所发起的，否则是无效的。

通过这种方式，每个实体就变成自己的认证中心（CA）。自我证明的委托就避免了对外部第三方的依赖，直接由发布委托的主体对委托链进行自我证明。有关认证中心 CA 的内容，请参阅下一章 PKI 技术的相关内容。

4. 具体实例分析

为了说明 dRBAC 基础组件在实际系统中的应用，我们来分析 6.2.1 节描述的应用实例。

例 1：由 A、B 和 C 三个国家的军队正在举行联合军事演习。A 国方面的视频设备获得一些视频信息，A 国军队指定史密斯将军(Smith General)既可以观察视频信息又可以将这个权力委托给参加军事演习的 B 国和 C 国军队。Alice、Joe

分别是 A 国和 B 国的军士，Bob 和 Carle 分别是 B 国和 C 国的将军。

下面从该实例中所体现的实体、角色、委托和最终的访问控制结果进行相关分析。

1) 实例中包含的实体

在该实例中体现出的实体包括有 A 国军队、B 国军队、C 国军队、视频设备、Smith 将军、Bob 将军、Carle 将军、Alice 军士、Joe 军士，这些实体分别简写为：A、B、C、Camera、Smith、Bob、Carle、Alice、Joe。

2) 实例中包含的角色

在该实例中体现出的角色包括有 A 国将军、B 国将军、C 国将军、A 国军士、B 国军士和视频设备的查看角色，这些角色分别简写为：A.General、B. General、C. General、A. Soldier、B. Soldier、Camera. View。视频信息的访问权限被指派给 Camera. View 角色，具有该角色的用户才能够查看视频信息。

3) 实例中包含的委托

在该实例中发布的委托信息如下。

(1) 实体 Camera 将角色 Camera. View 分配权授予给角色 A. General：

[A. General→Camera. View’] Camera （6.1）

这是一个分配委托，也是一个自我证明的委托。

(2) 实体 A 将角色 A. General 指派给 Smith，使得 Smith 获得角色 Camera.View 的分配权：

[Smith→A. General] A （6.2）

这是一个客体委托，也是一个自我证明的委托。

(3) 实体 Smith 发布第三方委托，将角色 Camera. View 指派给实体 Alice、角色 B. General、角色 C. General：

[Alice→Camera. View] Smith （6.3）

[B. General→Camera. View] Smith （6.4）

[C. General→Camera. View] Smith （6.5）

以上均为第三方委托，也是客体委托。

(4) 实体 B 将角色 B. General 和 B. Soldier 分别指派给实体 Bob 和 Joe：

[Bob→B. General] B （6.6）

[Joe→B. Soldier] B （6.7）

以上均为客体委托，也是自我证明的委托。

(5) 实体 C 将角色 C. General 指派给实体 Carle：

[Carle→C. General] C （6.8）

这是一个客体委托，也是一个自我证明的委托。

4) 实例中访问控制结果

根据上述的实体、角色和发布的委托，该实例最终的访问控制结果体现为“哪些用户拥有角色 Camera. View”？

通过（6.1）、（6.2）、（6.3）的委托链可以看出 Alice 拥有该角色。

通过（6.1）、（6.2）、（6.4）、（6.6）的委托链可以看出 Bob 拥有该角色。

通过（6.1）、（6.2）、（6.5）、（6.8）的委托链可以看出 Carle 拥有该角色。

所有的这些委托链都是以委托（6.1）这个自我证明的委托发起的，所以是有效的。

通过（6.7）的委托可以看出 Joe 不具有该角色。

下面再来分析另外一个应用实例。

例 2：现有 AirNet 公司和 BigISP 公司是市场合作伙伴，BigISP 的员工可以使用 AirNet 的服务。Sheila 是 AirNet 的市场部管理人员，负责管理 BigISP 的员工访问该业务。Maria 是一个 BigISP 员工，需要用她的员工身份访问 AirNet 公司提供的服务。

下面从该实例中所体现的实体、角色、委托和最终的访问控制结果进行相关分析。

1) 实例中包含的实体

在该实例中体现出的实体有 AirNet 公司、BigISP 公司、管理人员 Sheila、BigISP 公司成员 Maria，这些实体分别简写为：AirNet、BigISP、Sheila、Maria。

2) 实例中包含的角色

在该实例中体现出的角色有 AirNet 公司市场部管理员、BigISP 公司的员工和 AirNet 公司的服务等，这些角色分别简写为：AirNet. Mktg、BigISP. Member、AirNet. Access。AirNet 公司服务的访问权限被指派给 AirNet. Access 角色，具有该角色的用户才能访问 AirNet 公司的服务。

3) 实例中包含的委托

在该实例中发布的委托信息如下。

(1) 实体 AirNet 将角色 AirNet. Access 的分配权指派给角色 AirNet.Mktg：

[AirNet.Mktg→AirNet. Access’] AirNet （6.9）

这是一个分配委托，也是一个自我证明的委托。

(2) 实体 AirNet 将角色 AirNet.Mktg 指派给实体 Sheila，使得 Sheila 获得角色 AirNet. Access 的分配权：

[Sheila→AirNet. Mktg] AirNet （6.10）

这是一个客体委托，也是一个自我证明的委托。

(3) 实体 Sheila 发布第三方委托，将角色 AirNet. Access 指派给角色 BigISP. Member，使得 BigISP 的员工可以访问 AirNet 公司的服务：

[BigISP. Member → AirNet. Access] Sheila （6.11）

这是一个第三方委托，也是客体委托。

(4) 实体 BigISP 将角色 BigISP. Member 指派给实体 Maria：

[Maria→BigISP. Member] BigISP （6.12）

这是一个客体委托，也是一个自我证明的委托。

4) 实例中访问控制结果

根据上述的实体、角色和发布的委托，该实例最终的访问控制结果体现为“哪些用户拥有角色 AirNet. Access”？

通过（6.9）、（6.10）、（6.11）、（6.12）的委托链可以看出 Maria 拥有该角色 AirNet. Access，从而可以获得 AirNet 公司的服务。该委托链是以委托（6.9）这个自我证明的委托发起的，所以是有效的。

6.2.3 基本组件的扩展

dRBAC 采用实体、角色和委托等安全机制，能够实现动态结盟环境下的共享访问。通常，被系统保护的资源允许在不同级别上访问，系统也需要根据授权对象的不同而调整相应的访问级别。比如，A 国军官收到的视频数据比其他国家的军官具有更高的图像分辨率和更短的时间延迟。下面介绍的值属性将能够实现这样的安全目标。

1. 值属性

值属性在实体内定义，是能被设置成数值以便调整资源访问级别的数值参数。值属性本身不是一个访问权限，它必须和具体角色结合，才能调节该角色访问资源的安全级别，实现对受保护资源的细粒度的访问控制。dRBAC 使用值属性来增强表达访问控制安全策略的能力。

例如，对于 A 国获取的视频信息，A 国的 Smith 将军应该尽可能地实时收到数据。可是，出于军事秘密的考虑，可能需要延迟 5 个小时才能授权 A 国信任的新闻报道员（A.Reporter）访问视频信息。为了表达这样的安全策略，可以发布下列带值属性的委托：

[A. General → Camera. View with Camera. Delay=0] Camera

[A. Reporter → Camera. View with Camera. Delay=5] Camera

与基本 dRBAC 的客体委托语法相比，扩展的客体委托的语法，通过“with Camera.Delay=0”来达到设置值属性的效果。

上文是单个值属性的例子，Camera 视频资源可能具有多个独立的值属性。例如：

Camera.Delay：表示视频信息延迟多长时间允许访问；

Camera.Rez：表示视频的图像分辨率的问题，分辨率的变化范围从 0 到 1，其中 0 表示视频分辨率最低，1 表示分辨率最高。

为了能够独立地调整这些参数，dRBAC 可以在客体委托时指定多个值属性。例如，

[A.General →Camera.View with Camera.Delay=0
and Camera.Rez =1] Camera

[A.Reporter →Camera.View with Camera.Delay=5
and Camera.Rez =0.5] Camera

2. 值属性的分配委托

如同 dRBAC 中角色分配委托一样，可以将值属性的分配权指派给其他主体，使得其他主体具有调节值属性的权限。下面是一个具体的例子。

[B →Camera.Rez *= ’] Camera

[B.b →Camera.View] Camera

[B.a →B.b with Camera.Rez *= 0.5] B

通过上述的委托，Camera 授予实体 B 对值属性 Camera.Rez 的分配权，并且将角色 Camera.View 指派给角色 B.b。实体 B 将角色 B.b 指派给角色 B.a 时，可以将 Camera.Rez 设置为原有基本值的 0.5 倍。

3. 值属性叠加问题

在基本 dRBAC 中，角色会通过委托链的形式在各个实体之间进行传递。那么，值属性也会在委托链中具有叠加的问题。

例如，对于 A 国获取的视频信息，需要延迟 5 个小时才能授权给 A 国信任的新闻报道员（A.Reporter）访问视频信息，一个外国新闻部的报道员（Abroad.Reporter）可能在更长的延迟时间后接收同样的视频信息。

[A. General → Camera. View with Camera. Delay=0] Camera

[A. Reporter → Camera. View with Camera. Delay=5] Camera

[A → Camera. Delay += ’] Camera

[Abroad.Reporter→A. Reporter with Camera.Delay+=24]A

通过上述的委托， 则 Abroad.Reporter 最终收到的信息将会比原始视频信息

延迟了 29 个小时。

值得关注的是，每一个实体都必须在被授权的范围内发布委托，实体不可能授权给别的主体比其拥有的访问级别更高的访问级别。在 dRBAC 中通过设置适当的操作符和指定特定值来实现值属性的相关操作。在上述的委托的中，Camera 实体将值属性 Camera. Delay 的分配权指派给实体 A，并且定义 Camera.Delay 值只能进行累加“+=”的操作，说明实体 A 在进一步授权给别的实体时，只能使得其它实体收到视频信息的延迟时间更长。

4. 具体实例分析

为了说明值属性在实际系统中的应用，我们对例 1 和例 2 中的应用需求进行扩展，并进行相关分析。

例 3：在例 1 应用背景需求的基础上，添加访问级别的安全需求，要求 A 国军官收到的视频数据比其他国家的军官具有更高的图像分辨率和更短的时间延迟。

该实例中所涉及的实体、角色都和例 1 中完全相同，下文在例 1 的基础上，添加有关值属性、值属性委托和访问控制结果的相关分析。

1) 实例中定义值属性和操作符

根据应用需求，我们首先定义如下的值属性：

Camera.Delay：表示视频信息延迟多长时间允许访问，初始值定义为 0；

Camera.Rez：表示视频的图像分辨率的高低，分辨率的变化范围定义为从 0 到 1，其中 0 表示视频分辨率最低，1 表示分辨率最高，初始值为 1。

然后定义如下操作符和参数数值：

+：对数值属性加一个正值，值越高就表明延迟时间越长，基值是 0。

*：对数值属性乘以一个 0 到 1 之间的正值。值越大就表明图像分辨率越高，最大的值是不大于 1，表明图像分辨率不可能比发布者主体自己所拥有的分辨率更高。

2) 实例中对有关值属性的委托

在例 1 所发布委托的基础上，添加委托中对值属性的支持如下。

(1) 在例 1 中，（6.1）委托的含义是实体 Camera 将角色 Camera. View 分配权授予给角色 A. General。在本例中，实体 Camera 还需要将调整值属性的分配权也指派给角色 A. General，所以将（6.1）委托修改为如下三条委托：

[A. General→Camera. View’] Camera （6.13）

[A. General→Camera. Delay +=’] Camera （6.14）

[A. General→Camera. Rez *=’] Camera （6.15）

(2) 在例 1 中，（6.3）、（6.4）、（6.5）委托的含义是实体 Smith 发布第三方委托，将角色 Camera. View 指派给实体 Alice、角色 B. General、角色 C. General。在本例中，主体 Smith 发布委托时，根据授权对象不同，需要对值属性进行相应的调整，所以将委托（6.3）、（6.4）、（6.5）分别修改为（6.16）、（6.17）、（6.18）：

[Alice→Camera. View with Camera.Delay += 0
and Camera.Rez *= 1] Smith（6.16）

[B. General→Camera. View with Camera.Delay += 1
and Camera.Rez *= 0.5] Smith（6.17）

[C. General→Camera. View with Camera.Delay += 1

and Camera.Rez *= 0.5] Smith（6.18）

例 1 中其他的委托不需要修改。

3) 实例中最终访问控制结果

根据上述的实体、角色和发布的委托以及值属性设置，可以看出 Alice、Bob 和 Carle 具有对视频信息的访问权限。

但是根据委托 (6.16) 、(6.17) 、(6.18)中值属性的设置不同，可以看出 Alice 具有最短的延迟时间，延迟时间为 0；具有最高的图像分辨率，分辨率值为 1。Bob 和 Carle 的延迟时间为 1；图像分辨率为 0.5。

下面对例 2 中的应用进行相关扩展。

例 4：AirNet 公司和 BigISP 公司是市场合作伙伴，BigISP 的成员可以以一种受限制的方式使用 AirNet 的服务。比如，更小的带宽，更少的存储空间，更少的每月在线时间。Sheila 是 AirNet 的市场部管理人员，管理这个业务。Maria 是一个 BigISP 成员，想要用她的成员资格登录到 AirNet。提出的问题是：如何控制访问的级别或者服务的质量？

该实例中所涉及的实体、角色都和例 2 中完全相同，下文在例 2 的基础上，添加有关值属性、值属性委托和访问控制结果的相关分析。

1) 实例中定义值属性和操作符

根据应用需求，我们首先定义如下的值属性：

AirNet. Bandwidth：表示用户可以使用的带宽，初始值为 100KB/s；

AirNet. Storage：表示用户可以使用的存储空间，初始值为 50MB；

AirNet. Monthlyhrs:表示用户每月可以使用的在线时间，初始值为 60 小时。

然后定义如下操作符和参数数值：

-=：对数值属性减少一个正值，值越低就表明服务级别越低；

<=：收集证明链中所有值的最小值，初始值定义为 100KB/s。

2) 实例中对有关值属性的委托

在例 2 所发布委托的基础上，添加委托中对值属性的支持如下。

(1) 在例 2 中，（6.9）委托的含义是实体 AirNet 将角色 AirNet. Access 的分配权指派给角色 AirNet.Mktg，在此例中，实体 AirNet 还需要将调整值属性的分配权也指派给角色 AirNet.Mktg，所以将委托（6.9）修改为四条委托：

[AirNet.Mktg→AirNet. Access’] AirNet （6.19）

[AirNet.Mktg→AirNet. Bandwidth<=’] AirNet （6.20）

[AirNet.Mktg→AirNet. Storage -=’] AirNet （6.21）

[AirNet.Mktg→AirNet. Monthlyhrs -=’] AirNet（6.22）

(2) 在例 2 中，（6.11）委托的含义是实体 Sheila 发布第三方委托，将角色 AirNet. Access 指派给角色 BigISP. Member，使得 BigISP 的员工可以访问 AirNet 公司的服务。在此例中，实体 Sheila 发布委托时，将对值属性进行相应的调整，所以将委托（6.11）修改如下：

[BigISP. Member → AirNet. Access with AirNet.Bandwidth <= 100
and AirNet.Storage -= 20
and AirNet.Monthlyhrs -= 10]Sheila (6.23）

例 2 中其他的委托不需要修改。

3) 实例中最终访问控制结果

根据上述的实体、角色和发布的委托以及值属性设置，可以看出 Maria 利用

其成员身份获得 AirNet 公司服务的访问。

但是根据委托(6.23)中值属性的设置，可以看出 Maria 获得的网络带宽最大只有 100KB/s；获得的存储容量是 30MB；每月在线时间是 50 小时。

通过值属性的设置，使得 BigISP 中角色在共享 AirNet 服务时，对其提供更加细致的控制粒度。

dRBAC 通过发布角色委托，并且由多个角色委托构成委托链来实现角色在各个实体之间的传递，它提供了一个非常复杂的结构。如何在动态结盟环境下部署和实施该 dRBAC 是一个艰巨的任务。具体实施 dRBAC 的相关机制和方法超出了本书的范围，读者可以参考相关文献[**]。

6.2.4 dRBAC 安全性分析

针对跨域的动态结盟环境，dRBAC 提供了一个灵活的、基于角色委托的访问控制机制。dRBAC 定义了实体和实体内的角色，并且通过角色的委托来进行权限的传递，从而实施跨域的安全共享访问。并且在访问过程中，dRBAC 还通过值属性结合角色来调整系统资源所提供的访问级别。与其他的多域安全互操作访问控制技术相比，dRBAC 具有以下优势。

1. dRBAC 采用公私钥对来代表一个实体。

根据公钥密码学的原理，一个公私密钥对唯一地代表一个实体，这也是进行数据加密和数字签名的前提条件。所以每一个实体通过公私钥对都获得了全局唯一的标识符。而且，每一个实体都可以自主的签发证书，将角色委托给其他实体，并不需要第三方的认证中心（CA），这也使得系统管理更为简便。

2. dRBAC 通过委托链将角色在不同的实体之间进行传递。

dRBAC 通过发布角色委托来进行权限的传递。比如“Alice 委托给 Bob 某个角色”，它的含义就是 Alice 以自己的名义授予 Bob 行使某些角色的权限。Bob 可以继续将角色委托给 Charlie，那么角色就传递给了 Charlie。dRBAC 通过委托和传递，构建了更加具体的跨域访问机制。

相对于 6.1 节的角色映射技术而言，dRBAC 只是将实体的角色委托给另外一个实体内的某个角色，而并不需要了解另外实体内的角色层次如何设置，从而就避免了角色层次的披露问题。

但是从安全的角度来看，dRBAC 也存在一些值得探讨的问题。

1. dRBAC 缺乏对强制安全访问控制策略的支持。

dRBAC 主要依赖于自主安全控制，缺乏对强制安全访问控制策略的支持。在 dRBAC 中，每一个实体都可以发布“自我证明”的委托，将角色自主的委托给其他主体，而对于接收委托的主体缺乏必要的条件约束，完全依赖于发布主体对接收主体的“信任”。然而，在现实生活中，信任也必然会包含一定的安全风险。在一个安全保护级别要求很高的情形下，如何在 dRBAC 主体发布委托的过程中，实施强制访问控制的安全策略是一个值得研究的问题。

2. dRBAC 缺乏对委托链的传递控制。

dRBAC 在发布角色委托的过程中，缺乏对委托链的深度控制和广度控制。比如，主体 A 将角色授予到 B 内某个角色以后，主体 B 就有权继续向下进一步进行角色的委托。比如，B 可能对主体 C、主体 D，乃至主体 N 等发布委托；而且主体 B 将角色委托给 C 以后，主体 C 还可以进一步向下传递委托角色，这样角色的委托可能就失去了控制。如何对角色委托的过程实施深度和广度控制问题也是值得研究的问题。

6.3 安全虚拟组织结盟的访问控制

6.3.1 安全虚拟组织结盟的应用背景

在社会生产和生活中，为了共同完成某项任务，通常需要建立多组织的合作关系。比如 A 公司与 B 公司为了同一个商业目标，需要建立项目研究小组，进行有效的合作。A 与 B 都具有各自独立的网络设施和信息管理系统，他们之间为了共同目标需要对相关的资源信息进行共享，而各个组织又要保持对本地资源的自治权，并且这种共享关系多是临时的、动态的，每个组织可以根据自己的需要随时加入和退出该结盟关系。

该应用场景与 6.2 节的应用背景非常类似，若采用角色映射的方法就显得过于复杂，且不够灵活。6.2 节介绍的 dRBAC 采用以角色委托为中心的授权链技术来解决该应用背景下访问控制问题。但 dRBAC 概念比较抽象，实施过程也比较复杂，在分布式环境下如何发现和验证授权证书链是一个比较困难的问题。

本节介绍的安全虚拟组织结盟访问控制技术（Secure Virtual Enclaves，即 SVE）基于中间件技术，在不改变底层的操作系统、开放的网络协议标准和信息管理系统的基础上，设计和开发一套使用分布式应用技术的软件体系结构，支持结盟节点之间跨域的共享访问。

SVE 技术提供了一套软件基础设施，它支持多个组织共享对象同时又保持各组织对本地资源自治。这种基础设施对应用是透明的，无需改变每一个结盟节点现有的应用系统。下文将加入 SVE 结盟环境的每一个组织简称为“虚拟节点”，或简称“节点”。

6.3.2 SVE 体系结构和基本组件

SVE 结盟基础设施主要由虚拟节点管理、安全策略管理、安全策略交换控制器和访问监控器等组件组成。其中虚拟节点管理组件负责创建虚拟节点、批准其他节点加入和进行控制信息交换；安全策略管理组件负责创建节点内实施访问控制的安全策略；安全策略交换控制器组件负责在节点之间发布和交换安全策略；访问监控器组件负责共享资源的安全策略具体执行和实施。

SVE 结盟基础设施的体系结构和各个组件之间的关系如图 6.9 所示。

1. 安全策略管理

针对节点中的共享资源，系统需要制定相应的安全保护策略。管理员使用安全策略管理组件，创建和维护本地的安全策略。下面针对某个节点，来分析其安全策略的表示方法。

在节点内，采用域和类型实施语言（Domain and Type Enforcement ，即 DTE）来描述系统的安全策略。在该描述语言中，主体被分配相应的角色，称为主体角色；客体被分为不同的安全类别，称为客体类别；系统针对主体角色和客体类别设置授权规则和访问约束条件，授予主体角色对客体类别相应访问权限。总体而言，系统安全策略可以划分为四种不同类型，分别描述如下：

- 主体角色：为主体指定不同的角色；
- 客体类别：将共享的信息资源划分为不同的安全类别；
- 授权规则：针对不同的主体角色，对客体类别授予相应的访问权限；
- 访问约束：对授权规则的补充，针对不同的客体类别设定细粒度的访

问限制条件。

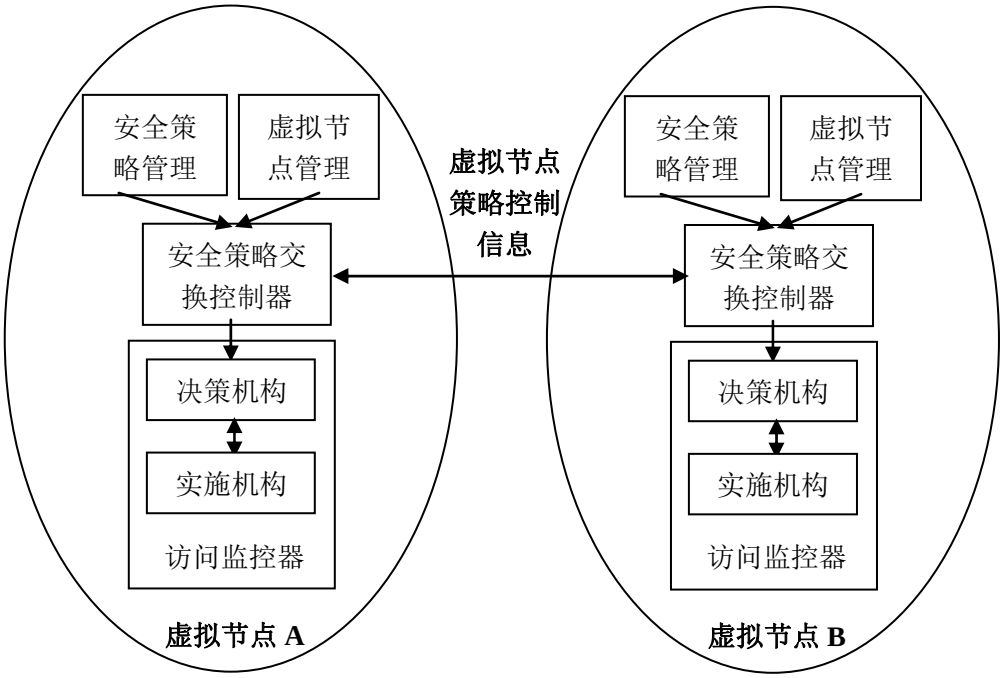


图 6.9　SVE 结盟基础设施的体系结构

下面分析一个具体实例，来说明这些规则的含义。

例1，在节点B中，分别定义四种不同类型的访问规则。

(1) 主体角色

Alice@eweb.com = engineer_d

Bob @ eweb.com = accountant_d

(2) 客体类别

specifications_t = https://eweb.com/specs/*;

source_code_t = https://eweb.com/source/*;

financials_t = https://eweb.com/finan/*;

(3) 授权规则

engineer_d = specifications_t，　source_code_t

(4) 访问约束

accountant_d = financials_t+TimeInterval!900!1600!M!F

从该实例可以看出：

规则（1）描述了主体的角色信息，主体Alice@eweb.com指派为工程师（engineer_d）角色，主体Bob@eweb.com指定为会计师（accountant_d ）角色；

规则（2）中定义了需要保护的客体资源的安全类别，分别是文档资料（specifications_t）类、源代码（source_code_t ）类和财务信息（financials _t ）类；

规则（3）和（4）均为授权规则，规则（3）为工程师角色授权，使得工程师角色可以访问文档资料类和源代码类的客体；

规则（4）为会计师角色授权，使得会计师角色可以访问财务信息类的客体；在规则（4）中“TimeInterval!900!1600!M!F”为访问约束条件，其含义是会计师角色必须在从周一到周五的上午9：00至下午4：00的时间范围内，才能执行对财

务信息类客体的访问。

2. 安全策略交换控制器

安全策略交换控制器就是在各个节点之间的发布、交换和共享相关安全策略的组件。

从上节的描述可知，在某个节点中，完整的安全策略包括主体角色、客体类别、授权规则和访问约束等四个部分。节点之间要实现跨域的安全访问操作，必须在他们之间共享某些安全策略。针对哪些策略需要共享和如何共享的问题，我们来具体分析一个的应用实例。

假如节点A中的某个主体Sub_A对节点B中某个客体提出访问请求，则节点B至少需要获得下列的信息，才能决定是否允许其访问。首先，节点B必须能够识别节点A中Sub_A的主体角色，然后将A中的主体角色与B节点的授权规则关联起来，获得A中的主体角色在B节点的访问权限和访问约束。

因此，在节点A在加入SVE时，需要把A中所有的主体角色信息都发给B。在B中，这些A中的主体角色信息被用来识别A中的主体，所以在B中把这些信息称之为“主体识别规则”。在节点B中，具有包括主体角色、客体类别、授权规则和访问约束等在内的本地安全策略信息。针对节点A中的主体对B中客体类别的访问，节点B需要对A中的主体角色进行另外的授权，添加A中的主体角色对本节点中客体类别实施访问的授权规则和访问约束。

从上述的安全策略共享过程可知，安全策略交换控制器主要完成下列的功能。

1) 负责和其他节点的安全策略交换控制器组件进行安全通信，将“主体识别规则”的信息在SVE的各个节点之间共享。比如节点A的主体角色信息要发布给B，同时B的主体角色信息也要发布给A。
2) 在本节点内部，负责对外节点的主体角色进行另外的授权，根据节点之间共享协议的约定，添加外节点对本地客体类别的授权规则，授予外节点的主体角色对本地客体类别的访问权限。
3) 在本节点内部，负责将外节点的“主体识别规则”信息和外节点对本地客体的授权规则，以及本地的安全策略进行汇总，然后统一交给本地的访问监控器，作为访问监控器实施访问控制决策过程的判定依据。

由上述的流程可知，A中的主体角色在B内能够访问的客体类别和约束条件，均由节点B来设定，从而体现了节点B对本地资源的自治权限。

3. 访问监控器

访问监控器组件负责对主体提出的访问请求实施检查，依据安全策略作出访问控制决策，最终决定允许或中止主体的访问请求。

它从安全策略交换控制器组件接收外节点的“主体识别规则”信息和外节点对本地客体的授权规则，以及本节点的安全策略，然后保存到本节点的数据库服务器中，作为访问控制决策的依据。

访问监控器内部分为访问控制实施机构和决策机构两部分。访问控制实施组件主要负责捕获针对系统受保护对象的访问控制请求，从访问请求中分析提出访问的主体信息、客体信息、访问权限和具体的系统参数，并把这些参数交给访问控制决策结构组件。决策结构做出是否允许访问的判断，并将判断结果返回给实施机构，实施机构根据判断结果允许或中止访问请求。

决策机构接收到实施机构发送的主体信息、客体信息、访问权限和具体系统参数以后，实施访问控制的决策过程。主要包括下列的步骤：

1) 首先区分是该主体是属于外节点还是本节点，从而根据“主体识别规则”或本节点的主体角色信息来获得主体的角色；
2) 然后根据主体的角色信息在授权规则中获取有关该主体角色的授权信息和约束条件；
3) 根据客体信息，在客体类别规则中，获得客体所属于的客体类别信息；
4) 检查主体的角色对应的授权规则中是否包含客体所在的客体类别信息；如果主体角色对客体类别具有访问请求所涉及的访问权限，则进行下一步判定，否则返回拒绝访问的信息；
5) 执行授权规则检查以后，还要检查具体的系统参数是否满足访问约束的限制，若满足约束条件，则返回允许访问的信息，否则返回拒绝访问的信息。

决策机构将判定结果返回给实施机构，由实施机构根据判定结果进行相应的处理，允许或禁止该访问请求。

4. 虚拟节点管理

虚拟节点管理组件主要负责管理节点一级相互之间的控制关系，比如创建新的节点、加入 SVE、退出 SVE 和撤销 SVE 等功能。

SVE 结盟环境启动的一般流程如下。

1) 首先由某一个节点的管理员开始使用虚拟节点管理组件，命名和创建一个SVE结盟环境，创建者成为该结盟环境的唯一成员。该节点创建内部的安全策略，对本地的访问主体进行授权。
2) 然后其他的节点初始化虚拟节点管理组件，开始申请加入该结盟环境，在申请被接纳以后，该申请节点和原有的节点建立结盟关系，完成节点之间控制信息的交换，各个节点内部完成与外节点的“主体识别规则”交换和对外节点主体授权的处理。
3) 节点内主体可以使用SVE结盟环境，提出对外节点资源对象的访问请求，外节点的访问监控器根据相应的安全策略完成权限检查，允许或中止节点之间的安全访问操作。
4) 在完成安全访问操作以后，节点可以选择退出SVE结盟环境。剩余最后的一个节点，负责撤销整个SVE结盟环境。

这些创建、加入和退出以及撤销整个 SVE 结盟环境的功能都由虚拟节点管理组件负责实施。

6.3.3 应用实例分析

为了说明整个 SVE 结盟环境的运作流程，下面具体分析一个应用实例。

例 2，现有节点 A 和节点 B，节点 A 中主体 Carle@eit.com 提出对节点 B 内资源的访问。其中节点 B 中的本地安全策略如例 1 所示。节点 A 中主体 Carle@eit.com 的角色信息如下定义：

Carle@eit.com=programmer_d

即为主体 Carle@eit.com 指定程序员（programmer_d）的角色。

下面分析节点 A 和节点 B 建立 SVE 结盟环境实现安全共享访问的整个过程。

1) 由节点 B 的管理员使用虚拟节点管理组件，创建SVE结盟环境。管理员使用安全策略管理组件，完成本节点内的安全策略创建和管理。
2) 节点 A 初始化虚拟节点管理组件，加入节点 B 建立的SVE结盟环境，节

点A把主体Carle@eit.com的主体角色信息传递给节点B，节点B接收A的“主体识别规则”，并且为其进行另外授权。在B中添加如下规则：

(1) A的主体识别规则

Carle@eit.com = programmer_d

(2) A的主体角色对B的客体类别的授权规则

programmer_d = specifications_t

即A中的主体Carle@eit.com具有A的程序员角色，而且在节点B中，授予A中的程序员角色对文档类信息的访问权限。

3) 节点A中主体Carle@eit.com提出对B中文档类资源 https://eweb.com/specs/index.html 的访问。节点B中的访问监控器截获该次访问请求，然后检查A的主体角色和A的主体角色在节点B中的授权规则，根据检查结果允许主体进行访问。

4) 如果节点A中主体Carle@eit.com提出对B中源代码类资源 https://eweb.com/source/index.html的访问。节点B中的访问监控器截获该次访问请求，然后检查A的主体角色和A的主体角色在节点B中的授权规则，会拒绝该主体的访问。

5) 访问结束后，节点A退出该SVE结盟环境，节点B最后撤销该SVE结盟环境。则整个过程结束。

需要注意的是，因为本例中只是描述A中主体访问B中资源的情形，所以在步骤（2）中，B中主体角色在节点A中的主体识别规则和节点A对B中主体角色的授权规则都被省略。

6.3.4 SVE 的安全性分析

SVE 结盟基础设施基于分布式应用的中间件技术，设计和实现了一个跨域的安全互操作体系框架，能够对共享的安全对象实施安全策略的保护，并且支持节点对本地安全对象的自治管理。从安全的角度来分析，SVE 结盟基础设施也存在一些问题。

首先，每一个加入虚拟组织结盟环境的节点，都必须将本地制定的主体角色规则在整个 SVE 范围内进行发布，如同 6.1 节的角色映射技术一样，它会存在外域角色层次的信息披露问题。

其次，在一个节点内部，外节点的主体识别规则发布到本节点以后，本节点没有将外节点的主体角色映射到本地主体角色，而是另外授予外节点的主体角色对本地客体类别的访问权限。这与6.1节的角色映射技术是有区别的。角色映射技术通过建立外域角色到本地角色的关联，将外域角色当成了本地角色来对待，不需要为外域角色重新授权。

参与SVE结盟环境的节点在两两节点之间都必须进行互相授权，不能提供授权的传递性。这与6.2节的dRBAC技术是不同的。dRBAC技术能够提供基于角色委托的授权链，可以在实体之间进行权限的传递。比如，实体A可以将角色A.role1委托给实体B的角色B.role2，而实体B可以将角色B.role2进一步委托给实体C的角色C.role3。则实体C的角色C.role3可以通过角色B.role2传递获得角色A.role1具有的权限。而这种权限的传递功能是SVE结盟环境所不具备的。

因此，对比 dRBAC 技术而言，SVE 技术缺乏必要的灵活性，必须在两两节点之间进行互相授权。因此，SVE 技术适合于少量节点之间建立较为简单的动态结盟环境。比如几个相关企业或几个相关部门为了同一个目标，建立临时动态

的结盟合作关系。

6.4 结合 PKI 的跨域基于角色的访问控制

6.4.1 跨域的基于角色访问控制应用背景

在目前大规模的 Internet 应用环境下，基于 Web 的浏览方式为用户获取网络上各种信息提供了便利。如果服务器提供的是敏感或重要信息，那么系统就需要针对这些信息提供访问保护。这样的 Web 服务有很多，比如大学图书馆提供的电子资源服务，通常都包含有最新的期刊、学术会议和学位论文等在线检索和下载服务。这些信息资源都具有十分重要的学术科研价值，必须是经过系统授权的用户才能进行访问。

设有 A 和 B 两所大学根据联合办学协议，互为对方的学生开放各自图书馆提供的在线服务。因为学生毕业和新生入学等情况，用户经常发生变化。因此 B 的服务器在面向 A 大学的学生提供服务时，应基于“A 大学的学生”这个角色授权，而无需针对 A 大学的每一个学生进行授权。反之，A 的服务器也是如此。

当 A 大学的学生访问 B 的服务器时，发出访问请求的用户需要证明自己确实来自于 A 大学这个组织，而且具有“A 大学的学生”这个角色，才能够获得在线服务。

针对大规模的分布式系统环境下如何识别用户的身份问题，现阶段主要采用 PKI 的技术。每一个组织内部建立 PKI 机制，为每一个用户发布 X.509 证书来证实身份。不同的组织之间借助交叉认证技术来相互识别和信任。具体实施技术可以参考下一章 PKI 技术的相关内容。

在服务器方验证用户的身份以后，必须解决的问题就是如何验证用户具备相应的角色。针对该问题，下面介绍的结合 PKI 的跨域基于角色的访问控制技术，是从服务器端的访问控制表设置、用户角色的表示、角色层次链的设计、服务器端角色验证等方面提出的具体解决方案。

6.3.2 访问控制表和用户证书

对整个系统而言，我们把提供在线服务的 B 大学称为服务器域，提出访问请求的用户所在的 A 大学称为客户域。针对服务器域和客户域，系统需要进行不同的访问控制设置。

5. 服务器访问控制列表设计

服务器提供的资源信息需要受到安全策略的保护，系统必须针对角色进行相应的授权设置。服务器采用访问控制表(ACL)的形式将权限和角色联系起来。ACL 就是将系统保护的资源或服务的访问权限授予给相应的角色。因此，ACL 目录表中的每一项可以被看作一个（角色，访问权限）序偶。下文具体分析一个 ACL 目录表的实例。

表 6.1 服务器 ACL 列表设置

角色	权限

A 大学的学生	查询
B 大学的学生	查询、下载

例 1，在 B 大学的图书馆在线服务器中，针对某项期刊查询和下载服务，可以设置如表 6.1 的 ACL 列表。

从表 6.1 可以看出，对于该项期刊查询和下载服务，系统针对不同的角色，授予不同的访问权限。来自于 A 大学的学生可以进行在线查询服务，但是没有下载的权限；来自于 B 大学的学生不但可以进行在线查询服务，而且可以对期刊进行下载操作。

6. 客户域中用户角色证书设计

从服务器域的 ACL 列表可以看出，服务器域和客户域就某个角色名字，作为共享访问的管理协议达成一致。比如，在例 1 中，A 大学和 B 服务器域以“A 大学的学生”这个角色达成一致，针对该角色进行相应的授权。

服务器域相信客户域为每一个提出访问请求的用户，分配相应的角色。客户机必须给服务器提供如下证据：通过明确的指派表明用户是客户域的一个成员；在客户域中，用户拥有相应的角色。

我们使用用户角色证书表明用户是某个客户域的成员，证书中不仅包含有该用户的公钥信息，而且为用户指派一个非常明确具体的角色，其中公钥信息主要是为了验证用户的身份。因为在用户角色证书设计时，我们主要强调其角色信息，所有省略了其公钥信息。

用户角色证书的一般形式如下：

[user → role]A

该用户角色证书表明：在客户域 A 内，用户 user 具有角色 role。

7. 客户域中角色层次证书设计

如果在客户域内角色集是具有层次关系的，我们就必须保证用户的角色能够完整的体现它在角色层次中的位置。

在一个角色层次中，如果角色 r`是角色 r 的子角色，记作 r≥r`，那么角色 r`具有的所有权限均被角色 r 继承。

我们使用角色层次证书揭示角色之间的直接继承关系。其形式为：

$[role_1 \rightarrow role_2]A$

该角色层次证书表明：在客户域 A 内，角色 $role_1$ 是 $role_2$ 的直接上级角色，角色 $role_1$ 能够继承 $role_2$ 具有的任何权限。

一般情况下，用户将从他具有的最有特权角色开始，通过角色层次继承关系，获取其具有的所有权限。我们使用形如 $r \geq r_1$，$r_1 \geq r_2$，…，$r_n \geq r`$ 角色层次证书链，建立 r≥r`的继承关系。这些证书被客户传给服务器，服务器首先证实这个链中的所有证书都是有效的，然后再确认这是一个用户与其相应角色关联的链。

8. 用户角色证书和角色层次证书格式

在客户域内具体实现用户角色证书和角色层次证书的格式如图 6.10 所示，其中省略了与用户、角色、角色层次关系不是很紧密的公钥证书的项目。

6.10 用户角色证书和角色层次证书

具体实现的证书格式与 X. 509 公钥证书和 PMI 属性证书是一致的，角色证书的“角色”项和层次证书的“子角色”项可以被定义为 X.509 标准证书的扩展项。其中 X. 509 公钥证书具体的内容项可以参考有关 PKI 技术的相关章节内容。

6.4.3 客户域内证书撤销

当客户域内的角色层次发生变化时，系统必须对相应的用户角色证书和角色层次证书进行撤销，而撤销行为当然也会影响到服务器端 ACL 列表中的相关设置。用户角色和角色层次关系均可以被撤销。

如果一个角色 r 被删除，那么所有与角色 r 直接关联的用户角色证书和角色层次证书也必须同时被撤销，然后发布新的角色层次关系证书进行相应的补充。比如，$r_1 \geqslant r_2 \geqslant r_3$，而 r_2 被删除了，那么 r_1 到 r_2 和 r_2 到 r_3 的角色层次证书都必须被撤销，然后由新的 r_1 到 r_3 的角色层次证书来替代。

最后，客户域管理员将会告诉每一个受影响的服务器域管理员。ACL 列表中有关 r_2 角色的相关设置也需要被删除，这将是角色 r_2 被撤消的直接后果。

我们是否可以这样假设：如果证书被撤消，而且客户域是可信的，为什么不能简单地认为客户将拒绝提交任何撤消证书？如果是这样的话，服务器将不必每次都需要检查客户证书的有效性，将会提高实现访问控制的效率。这里的安全威胁是如果一个攻击者或怀有恶意的用户，在他的证书被撤消以后，还可能复制、存储、重新使用被撤消证书，直至证书的有效期限到期为止。

6.4.4 应用实例分析

为了说明客户域中用户使用用户角色证书和角色层次证书，访问服务器域中服务的整个流程，我们来分析下文的一个实例。

例 2，假设在服务器域 B 大学图书馆在线服务器中，针对期刊查询和下载服务，设置的 ACL 列表如例 1 中所示。在客户域 A 大学内角色层次中，具有“教授”、“教师”、“助理”、“博士生”、“硕士生”和“学生”等角色，它们之间的继承关系如图 6.11 所示。设用户“张三”具有“博士生”角色，那么从角色层次中可以看出，他可以通过角色层次继承关系得到硕士生角色和学生角色的权限。

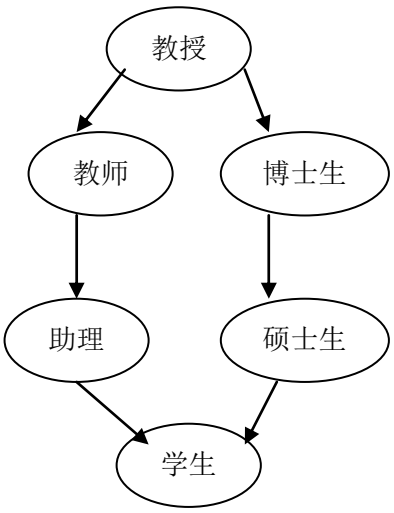


图 6.11 客户域 A 内的角色层次图

若用户张三提出对服务器端的访问，则它必须提供如下的用户角色证书和角

色证书链：

[张三 → 博士生]A，[博士生→ 硕士生]A，[硕士生 → 学生]A。

在服务器端，服务器必须验证每一个证书的有效性，而且从角色层次证书链中证明在客户域 A 内博士生角色通过硕士生而继承学生角色的权限。通过上述验证以后，服务器允许张三以角色学生的权限进行访问。

如果 A 内部角色层次发生了变化，比如删除了角色硕士生，则 A 会撤销[博士生→ 硕士生]A，[硕士生 → 学生]A 的证书，并且发布新的证书[博士生→学生]A 来替代原来的角色继承关系。

若怀有恶意的用户继续提供原有的用户角色证书和角色证书链，请求对服务器端的访问：

[张三 → 博士生]A，[博士生→ 硕士生]A，[硕士生 → 学生]A。

则服务器通过验证角色层次证书链，就会发现证书[博士生→ 硕士生]A 已经被撤销，则拒绝用户的访问请求。

6.4.5 跨域基于角色访问控制技术的安全性分析

针对在跨域的大规模应用中如何实现基于角色访问控制的问题，我们设计用户角色证书和支持角色层次的证书链，结合服务器的 ACL 列表设置方案，实现了跨域的基于角色访问控制，同时支持角色层次的授权，并讨论了相应的证书撤销机制。

从安全的角度分析，本文的方案与 6.3 节所介绍的 SVE 技术类似，服务器需要针对客户域内的角色进行单独的授权。与 SVE 技术不同的是，客户域无需把每一个用户的主体识别规则都发送给服务器域，因为客户域的用户成员众多，而且经常发生变化。服务器域不用保存客户域的主体识别规则，只需要验证提出访问请求的用户是否具有相应的角色即可。从这个方面来看，结合 PKI 的跨域的基于角色访问控制技术比 SVE 技术更加灵活和简单。

与 6.1 节所介绍的角色映射技术相比，结合 PKI 的跨域的基于角色访问控制技术在服务器域需要为客户域的角色单独授权，并没有把客户域角色当成本地的角色来对待。当客户域角色层次发生变化时，只需要撤销客户域角色的在本地 ACL 列表设置即可，不会影响到本地的角色层次关系和授权设置。

总体而言，结合 PKI 的跨域的基于角色访问控制技术的主要特点是结合 PKI 技术，采用用户角色证书和角色层次证书实现跨域的基于角色访问控制，其安全性主要与 PKI 技术相关。它主要适合于大规模的 Internet 应用环境下，客户域和服务器域的用户众多，而且经常发生变化的应用情形下。比如，大学与大学之间，大企业与大企业之间建立协作共享关系等。

习题六

1. 试设计一个算法，检测建立一个从外域角色到本域角色映射是否会发生角色冲突，若发生了冲突，给出该映射与那些映射是冲突的。

2. 试列举一个 dRBAC 的应用实例，分析其实体、角色和角色委托关系，要求至少使用一次第三方委托；并思考如何使用一系列自证明的委托来替代该第三方委托。

3. 列举一个使用 dRBAC 模型中应用值属性的应用实例，并能够使用值属性进行传递委托，使得不同实体获得的不同的访问服务。

4. 从角色映射、角色授权、角色信息披露等方面对比 6.1 节、6.3 节和 6.4 节中所跨域的访问控制技术，说明各自优点和缺点。

第 7 章 基于信任管理的访问控制技术

随着计算机网络的普及和无线通信技术的发展，移动计算、普适计算、网格计算等成为了新兴的计算模式，各种新的基于 Internet 的应用、分布式系统也随之出现。在现今这种规模庞大的、动态的、开放式的信息系统中，传统的安全机制明显不再适用。在一个大规模的、异构的分布式系统中，系统的授权者无法直接知道用户，因此他必须使用由熟知用户的第三方所提供的信息；通常授权者只

在某种程度上相信第三方提供的某类信息。这种信任关系使得分布式授权不同于传统的访问控制。第 6 章介绍的多域安全互操作访问控制技术主要采用角色映射技术或直接给外域角色授权来解决跨域的安全访问问题，对实体之间信任关系的处理方面还存在不足。基于信任管理的访问控制技术正是基于实体对第三方实体的信任关系，来解决大规模的、异构的分布式系统之间授权访问问题而出现的一种分布式访问控制。

本章将介绍几个基于信任管理的模型，并对各个模型的作用、特点作一大致的描述和比较。PoliceMake 模型是由 M. Blaze 等人提出的第一代信息管理模型，其主要组件包括安全策略、信任证和一致性证明引擎，安全策略描述了对资源的授权保护策略，信任证可以表达实体之间的信任管理，一致性证明引擎以访问请求和信任证为输入，结合本地的安全策略，根据一致性证明算法做出访问控制决策。

KeyNote 模型是第二代信任管理系统，它沿用了 PolieMake 模型的大部分思想和原则，并且从策略描述语言和一致性证明算法上对 PoliceMake 进行了改进。

RT 模型是基于角色的信任管理系统，它结合了基于角色的访问控制和信任管理系统的优点，体现了基于属性的访问控制策略，对安全策略的表达能够更强，而且语义也直观易懂。

自动信任协商通过信任证、访问控制策略的交互披露，资源的请求方和提供方能够最终建立信任关系，协商过程一般强调自动化，不需要或者需要少量的人工参与。

7.1 信任管理的概念

7.1.1 基于信任管理系统的应用背景

我们首先来看如下的一个例子：为了方便客户操作以及实施财务管理，保险公司各种保险项目的收费业务将与银行储蓄业务绑定在一起。这样以来，客户办理保险只需在银行做相关的资金转帐，银行负责收费，保险公司负责买保人的资格审查和策略制订。银行职员和保险公司的职员分别来自不同的公司，但是他们通常需要访问对方公司的资源，例如，银行需要熟知保险公司保险业务规章制度和保险收费规则等信息，而银行内有关保险公司业务收费的数据库也提供给保险公司来共享访问。

在上面的例子中，对于保险公司和银行的职员来说，他们都隶属于不同的管理域，不仅需要访问本地的资源，也需要访问对方公司的部分资源。伴随着这种广泛的资源共享，给他们各自拥有的资源也带了更多的安全风险。

一些研究表明，对信息资源的非授权访问，构成了多域组织中主要的安全问题。所有的分布式系统需要解决访问控制这个共同的问题，即资源访问的权限管理问题，也就是根据一定的安全策略来允许或拒绝请求者对资源的访问请求。在单域环境下，一般采用集中式的传统访问控制技术，实施的步骤分两步进行：认证(Authentication)和访问控制。认证回答“是谁提出请求?”，访问控制回答“请求者是否有权执行请求的行为?”。将传统的安全机制直接应用于新的多域组织环境中存在着以下的问题：

1. 传统访问控制不能给陌生用户授权

在传统的访问控制中，用户往往是被系统所熟知的，系统根据用户的身份设

置访问权限。但是在大规模、动态开放式的多域环境下，系统可能存在着大量的用户，并且系统的用户集变化频繁。在访问控制发生之前，系统无法事先认知所有请求系统服务的用户，并对其分配相应的权限。例如，在前面的例子中，银行的管理者无法事先了解保险公司所有职员的信息。

2. 传统访问控制不能适应多域环境下可伸缩性和动态性的特点

在多域环境中，往往存在着大量的分布式管理域，多域之间需要资源共享，为其他方提供相关的服务，而且这种合作共享关系多是临时的、动态的，随时可以结盟，也随时可以解除。正是由于多域环境可伸缩性和动态性的特点，解决多域环境下的授权管理问题需要安全、灵活的委托机制。委托机制就是指将资源的访问权限以实体之间的信任为基础，进行授权传递。例如，A 管理域的系统管理员可以将权限委托给 B 域的系统管理员，由其授权给相应的用户，从而使得 B 域的用户获得 A 域的访问权限。委托机制可以提高系统的灵活性，支持陌生用户的授权访问。传统的安全机制没有委托机制；不能表达多域安全访问的需求，表达能力、可处理性和扩展性较差；

3. 传统访问控制不能适应多域环境下异构的特点

传统的安全机制只能管理单个管理域，在单域环境下实施集中统一的系统安全策略。而在多域环境下，不同管理域可能需要采用不同的策略，不能强制实施统一的策略和信任关系。例如，在前面的例子中，银行和保险公司的组织结构以及访问控制策略可能是不同的，管理者对需要受保护的资源设置不同的安全策略，从而设置在这些资源之上的安全机制也会不同。因此，多域环境下的访问控制机制必然面临多个管理域内安全策略的异构问题；

4. 传统访问控制不能适应多域环境下分散式管理的特点

传统的安全机制采用集中式的管理模式，访问控制服务器将是整个系统的安全控制核心，由其负责对用户的授权和权限验证，用户和权限等相关信息由服务器集中管理和存储。而这种方法，不能直接应用于多域环境中。多域环境下，授权相关数据往往分散式的存储于系统之中，不存在集中的访问控制服务器。多域环境下实施访问控制，具有很强的复杂性。访问控制决策的信息分散在整个系统中，可能与保护的资源不处于同一地理位置，需要访问控制服务器自行在分布式环境中查找相关信息。为了保证访问判定的效率，安全机制面临着相关信息分布和查找问题。

综上所述，传统的安全机制难以适用于大规模的、开放式的分布式环境，特别是强调资源共享、相互协作的多域环境。在多域环境中，权限管理者无法直接熟知访问请求者，访问控制决策依赖于相关第三方所提供的信息。通常将权限管理者与第三方之间某种程度上的依赖关系称之为信任关系(trust relationship)。这种信任关系使得分布式授权不同于传统的访问控制。信任管理(Trust Management)方法正是为了解决上述问题而提出的一种分布式访问控制机制。

7.1.2 信任管理的基本概念

信任管理的基本思想是基于开放系统中安全信息的不完整性，系统的安全决策需要依靠可信任第三方提供附加的安全信息。信任管理的意义在于提供了一个适合 Web 应用系统开放、分布和动态特性的安全决策框架。在信任管理系统中，需要理解下列的基本概念。

1. 信任

在不同的语义环境和上下文中，“信任”本身所表达的含义也不尽相同，不

同研究者对信任的定义和理解存在着差别。在信任管理系统中，“信任”是一个实体对其它实体特定行为的可能性预测，并且认为信任在一定条件限制下是可以传递的。例如，在前面提到的例子中，银行信任某保险公司，而该保险公司信任其内部职员。那么，银行也信任来自于该保险公司的职员，形成了一个可传递的信任链(trust chain)。但在现实生活中，不是所有的信任关系都具备可传递性，如 Alice 信任 Bob，Bob 信任 Charlie，但 Alice 不一定信任 Charlie。在信任管理系统中，为了支持陌生实体对资源的授权访问，一般认为信任关系是可传递的。

2. 信任证(credential)

信任证表示发布者相信该证书的拥有者具备某些能力、属性或者性质。本书后面的部分皆称信任证。信任证的主要结构如下：

(issuer， subject， message， signatureissuer)

其中每一个字段含义如下。

issuer：信任证的发布者

subject：信任证的拥有者

message：关于 subject 的能力、属性或者性质的描述

signatureissuer：发布者对信任证前面部分的签名

例如，形如(工商银行， 张三， 职员， 工商银行专用章)的信任证表达的含义就是“工商银行声明张三是工商银行的职员”。

形如(A，B，A.role，SignByA)的信任证表达的含义就是“实体 A 声明实体 B 具有角色 A.role”，即实体 A 将角色 A.role 所具有的权限授予给了实体 B。

从上述信任证的结构可以看出，发布者通过颁发信任证将各种权限属性授予给信任证的拥有者。信任证提供了一种以密钥为中心的授权管理机制，信任证具有可存储性和可验证性，信任证中包括证书持有方的属性信息，其优势在于能够为陌生方授权。

3. 委托(delegation)

委托是分布式授权的核心技术，它主要的任务是基于信任关系将权限 p 的授予权 p^D 指派给其它实体，其它实体将权限 p 或者授予权 p^D 进一步指派给另外的实体。比方说 Alice 是资源拥有者，他将资源访问权限的授予权直接委托给 Bob，Bob 进一步将资源访问权授予 Charlie。由此，即使 Alice 与 Charlie 互不熟知，Charlie 也最终获得了资源的访问权。信任管理系统采用委托的技术，在陌生实体之间形成基于信任关系的委托链，从而支持多域环境下异构、动态和可伸缩性的授权访问。

4. 一致性证明(proof of compliance，简称 PoC)问题

在基于信任的访问控制系统中，每个实体拥有相应的信任证和基于委托机制形成的信任证集，服务提供方基于服务请求方提交的服务请求、信任证集以及本地安全策略，判定是否允许该请求。授权可以转化为回答一个一致性证明问题：“信任证集合 C 是否证明了请求 r 符合本地安全策略 P?”。

例如，在一个电子银行系统中，100 万元或以上的贷款至少经过 k 个部门经理的同意(安全策略)，这 k 个银行经理中的每一个必须能够证明其本人是部门经理(提供信任证或信任证集)，这样的信任证必须由合法的信任权威颁发(信任关系)。当有用户提出贷款请求时，必须同时提供 k 个部门经理的信任证集，系统需要验证贷款请求、信任证集是否符合系统的安全策略（一致性证明）。

7.1.3 信任管理的组件和框架

信任管理采用对等的授权模式，每个实体可以是授权者、第三方信任证发布者或访问请求者。信任管理系统的基本组件包括信任证系统、本地安全策略库、一致性性验证器和应用系统等，各个组件之间的关系如图 7.1 所示。

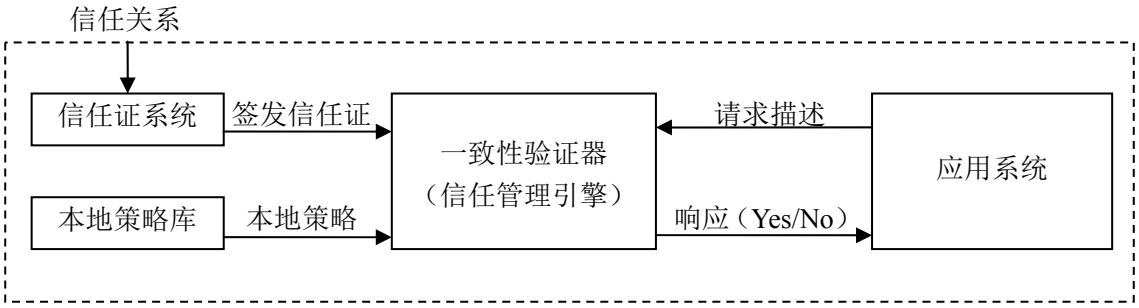


图 7.1 信任管理系统的基本框架

在图 7.1 中，本地策略库保存有对本地资源保护的授权规则，这些规则是系统作出访问控制决策的最终依据。服务方既可使用安全策略对特定的服务请求进行直接授权，也可将这种授权委托给可信任第三方。

信任证系统主要实现信任证的发布功能。服务方基于与第三方之间的信任关系发布信任证，授予给第三方相应的权限。可信任第三方可以继续向下发布信任证，形成以信任证链为基础的权限委托关系。

应用系统提出对资源的访问请求，并需要获得资源保护方的响应信息。资源保护方通过信任管理引擎(Trust Management Engine,即 TME)组件来做出访问控制决策，决定是否允许该访问请求。因此，信任管理引擎可以看作是整个信任管理模型的核心，它根据输入的三元组(服务请求、信任证集和本地策略)，采用与应用独立的一致性证明验证算法，输出访问控制决策的判断结果。

从前面的分析和图 7.1 可以看出，信任管理引擎是信任管理系统的核心。策略和信任证使用通用的、标准的语言进行编写，避免了采用的描述语言和一致性证明算法与特定应用相关，能以灵活、通用、可靠的方式解决开放分布式系统的授权问题，从而解决了多域环境下访问控制机制的异构问题。在信任管理引擎的设计目标中，通用信任管理引擎要求所作出的授权决策应只依赖于引擎外部的输入参数(服务请求、信任证集和本地策略)，而不是引擎的设计或实现中隐含的策略决策，这样能确保授权决策的可靠性。

7.1.4 信任管理技术的优点

与传统的访问控制技术相比，信任管理系统更能够适应多域环境下动态开放式和可伸缩性以及异构性的安全访问需求，具有以下显著优点：

(1) 支持不被系统所熟知的服务请求者访问系统。基于信任证集(证书链)和本地安全策略判定服务请求者是否有权在系统中执行某项操作，服务请求者不需要在系统中事先注册便可访问系统；

(2) 支持委托的机制，实现分散式授权的模式，权限的控制者能够以受控的方式将权限委托给其它域中另外的用户，从而支持多域间的跨域授权；

(3) 策略和信任证使用通用的、标准的语言进行编写，避免了采用的描述语言和一致性证明算法与特定应用相关，能以灵活、通用、可靠的方式解决开放分布式系统的授权问题，从而解决了多域环境下访问控制机制的

异构问题。

当前典型的信任管理系统有 PolicyMaker、KeyNote、REFEREE、SPKI/SDSI、DL 以及 RT 模型簇等，它们在设计和实现信任管理系统时采用了不同的方法来达到信任管理的功能。本书将以 PolicyMaker、KeyNote、RT 模型簇和自动信任协商模型为例对其进行介绍。

7.2 PoliceMake 模型

7.2.1 PoliceMake 模型简介

PolicyMaker 模型是 M. Blaze 等人依据他们所提出的信任管理思想所设计的信任管理模型。他们又根据该模型的原理和方法实现了第一代的信任管理系统，因此，通常把 PolicyMaker 当做第一代信任管理系统的代表。

PolicyMaker 为 Web 服务的授权访问提供了一个完整而直接的解决方法，取代了传统的认证和访问控制相结合的做法，并且提出了一个独立于特定应用的一致性证明验证算法，用于验证服务请求、信任证和安全策略是否匹配。PoliceMake 模型的组件主要包括：安全策略和信任证的表达和一致性证明。安全策略描述了对资源的授权保护策略，信任证可以表达实体之间的信任管理，一致性证明引擎以访问请求和信任证为输入，结合本地的安全策略，根据一致性证明算法做出访问控制决策。

1. 安全策略和信任证表示

PolicyMaker采用一种可编程的机制来描述安全策略和信任证，所有的安全策略和信任证均可归结为断言(Assertion)。简单来讲，断言可由二元组(f，s)表达，s表示授权者，f是一个由可编程语言编写的代码，描述所授权限的性质和对象。在信用证中，s为证书发行者，信用证必须由发行者签名，在信用证使用之前要验证签名。

(1) 安全策略断言

安全策略是描述对受保护的本地资源的授权访问规则，即描述那些主体能够实施对本地资源的访问。

在安全策略断言中 s 为关键词 POLICY，当应用程序调用信任管理引擎时，要输入一个或多个策略断言，这些规则是系统作出访问控制决策的最终依据。策略断言是本地存储的，因此无须签名保护。

例如，策略断言“允许来自于 Bob Labs 的成员对本地资源进行访问”，采用二元组格式表示的该策略断言如下：

（f_0，POLICY）；

f_0：“x is member of Bob Labs”。

其中 f_0 是由可编程语言描述的安全策略。

(2) 信任证断言

信任证表示发布者相信该证书的拥有者具备某些能力、属性或者性质。信任证的具体内容和表示形式参见 7.1.2 节中的描述。

在信任证断言中，s 为证书的发布者，通常采用一个公钥来代表该发布者主体。信任证必须由发布者签名，在系统采用该信任证之前必须使用发布者的公钥

来验证签名。

例如，信任证断言“Bob 声明实体 Alice 是来自于 Bob Labs 的成员”，采用二元组格式表达的该断言如下：

（f_1，Bob）；

f_1：“Alice is member of Bob Labs”。

其中 f_1 是由可编程语言描述的安全策略。

PolicyMaker 没有对书写断言内容 f 的程序语言作特别的要求，其原则是只要能被本地应用环境解释的编程语言均可用于书写断言。具体的可编程表示和编程方法可参考相关的文献。

2. 信任证的收集和验证

PoliceMake 模型规定应用程序应该负责信任证的收集和信任证的验证。此外，应用程序负责将信任证转换为 PolicyMaker 断言，以便下一步介绍的一致性证明所用。

3. 一致性证明

由于PolicyMaker没有指明特定的断言描述语言，因此其一致性证明验证算法必须独立于特定的断言描述语言。PolicyMaker的一致性证据观点，其通用形式是：

输入：请求R，证书集合{(f_0，POLICY)，(f_1，s_1)，…，(f_{n-1}，s_{n-1})，(f_n，s_n)}，其中(f_i，s_i)是 PolicyMaker的断言。

若s_i是POLICY，则表示该断言是策略断言；

若s_i不是POLICY，则表示该断言是信任证断言；s_i代表发证者，f_i是由可编程语言描述的证书拥有者具有的能力和对资源的访问权限。

问题：是否存在证书的有限序列 i_1，i_2，…，i_t，其中 $i_j \in \{0，1，…，n-1\}$ 且 i_j 可以相同并无须穷举{0，1，…，n-1}，最终证明或拒绝了 R。

PolicyMaker 进行一致性证明验证的一般步骤描述如下：

(1) 创建一个黑板，最初仅包含请求字符串 R，通常为了形式化描述的方便，最初的接受记录集表示为(Λ，Λ，R)；

(2) 选择并调用断言。从证书集合中选择一个断言，运行该断言。当断言(f，s)运行时，先读取黑板中的内容，当作断言 f 的输入，并根据 f 内部的可编程处理过程，得到相应的输出，输出的结果既是向黑板中添加零到多条接受记录(i，s_i，R_{ij})，但不能够删除其他断言已写入黑板的接受记录。R_{ij} 是发布者 s_i 根据黑板已有的接受记录所确定的特定应用动作，可以是一个输入请求 R，也可以是一些用于断言间交互的操作，这反映了断言间交互过程。

(3) 在第(2)运行的过程中，同一断言可以根据需要可以调用多次。经过有限次的调用断言后，可能得到下列三种结果：

- 若黑板中存在一条能证明请求 R 的接受记录，则一致性证明验证成功结束，返回的结果是允许该请求；
- 若黑板中存在一条能拒绝请求 R 的接受记录，则一致性证明验证成功结束，返回的结果是拒绝该请求；
- 若所有的断言经过多次重复循环调用后，仍旧不能得不到“证明”或“拒绝”请求的结果，说明该一致性证明是不可判定的。

显而易见，该问题的通用形式有可能是不可判定的。

7.2.2 PoliceMake 模型实例分析

例 7-1 在某企业中，所有超过 500$的支票需要管理者 A 和 B 同时批准，才能签发，而且管理者 A 需要在 B 批准的前提下，才批准超过 500$的支票。采用 PolicyMaker 模型实施该访问控制决策的过程如下：

1. 安全策略断言的描述

定义断言：(f_0， POLICY) ，它表示安全策略“所有超过 500$的支票需要 A 和 B 同时批准”。

2. 信任证断言

(1) (f_1， A) ：运行此断言将产生一个接受记录，(1， A， R_B)表示如果 B 批准了那么 A 将批准。

(2) (f_2， B) ：运行此断言将产生一个接受记录，(1， B， R)表示 B 批准。

3. 一致性证明的运行序列

(f_1， A)，(f_2， B)，(f_1， A)，(f_0， POLICY)

黑板的接受记录:

(1) {(Λ，Λ，R)}

(2) {(Λ，Λ，R)，(1， A， R_B)}

(3) {(Λ，Λ，R)，(1， A， R_B)， (2， B， R)}

(4) {(Λ，Λ，R)，(1， A， R_B)， (2， B， R)， (1， A， R)}

(5) {(Λ，Λ，R)，(1， A， R_B)， (2， B， R)， (1， A， R)， (0， POLICY， R)}

黑板的接受记录反映了信任证的执行过程。

(1) 表示黑板的初始化状态；

(2) 表示信任管理引擎调用信任证(f_1， A)以后得到的结果，(1， A， R_B)表示如果 B 批准了那么 A 将批准该支票；

(3) 表示信任管理引擎调用(f_2， B)以后得到的结果，(2， B， R)表示 B 批准该支票；

(4) 表示信任管理引擎再次调用(f_1， A) 以后得到的结果，(1， A， R)表示在 B 已经批准了支票的情况下，那么 A 也批准该支票；

(5) 表示信任管理引擎再次调用(f_0， POLICY) 以后得到的结果，即在 B 和 A 都批准该支票的情况下，此支票最终被授权批准。

在实际的一致性证明算法过程中，依据什么样的策略来选择某个断言进行调用是需要研究的问题。很显然，断言调用的顺序不同，得到的证明过程就不相同。因此，在一致性证明的算法中，PolicyMaker 的主要任务如下：

1. 运行服务请求者提交的所有断言，决定
   (1) 所有的断言以什么序列运行；
   (2) 每个断言运行多少次；
   (3) 丢弃所有的与一致性验证不相关的断言。
2. 维护整个一致性验证过程，保证前面所有断言运行输出的黑板接受记录的结果不被其它断言破坏；
3. 确保所有的断言被运行，并且没有其他的断言可以再运行，保证没有不相关的断言在无休止的运行，终止验证，并输出验证结果。

为此，M.Blaze等人用数学的方法精确地描述了一致性验证问题，并证明一般意义下的PoC问题是不可判定的，但一些限定的PoC问题存在多项式时间算法。

具体的证明过程和算法描述参见相关的参考文献。

7.2.3 PoliceMake 模型安全性分析

PolicyMaker是一个实验性质的信任管理系统,其功能相对简单,不提供安全凭证的收集和验证的功能.在功能和安全性方面主要具有下列特点：

(1) 由于 PolicyMaker 的设计目标是最小化，因此由应用系统负责安全信任证的收集和可靠性验证，使其在选择签名算法时具有一定的灵活性。但这加重了应用系统的负担，而且可能会因为信任证收集不充分而导致一致性证明失败。

(2) 每个实体基于对第三方的信任关系而决定是否接受其它实体提交的信任证，并且对信任证的验证功能交给应用系统完成，因此不需要建立中心权威机构（CA）。

(3) 一致性证明算法是经过严谨的逻辑证明的，具有形式化、可分析和证明的特点。但是 Poc 问题证明的过程中，只能处理满足单调性的策略断言，限制了一些策略的使用。

(4) PolicyMaker 不坚持特定的编程语言，由应用系统自由选择任意可编程的语言实现断言，具有一定灵活性。但采用任意的可编程的系统语言，使得信任管理系统的标准化成为难题。

7. 2.4 KeyNote 模型简介

KeyNote是M. Blaze 等人实现的第二代信任管理系统，它沿用了PolicyMaker的大部分思想和原则，即用信任证直接授权代替认证加访问控制，是PolicyMaker模型的改进和升级版。在设计时希望能将它作为信任管理系统的标准以便推广使用，因此其第二版设计完成后提交给IETF并成为了RFC文档。

KeyNote在设计之初就增加了两个设计目标，标准化和更容易集成到应用程序。KeyNote在系统的设计和实现上与PolicyMaker存在着很大的差别。

1. 行为属性

应用系统提交给信任管理系统评估的请求表示为一组属性名-属性值对(类似于shell变量)，被称为行为属性。属性名是行为属性的名，属性值是任意的字符串，其语义由应用系统负责解释，但需要在应用系统与使用它的信任证之间达成共识。_MIN_TRUST、_MAX_TRUST、_VALUES是系统的保留字，规定了信任评估返回值的取值范围。

- _MIN_TRUST用于表示服务请求与安全策略背离时的返回值；
- _MAX_TRUST用来表示服务请求与安全策略一致时的返回值；
- _VALUES是按照服务请求与安全策略的一致程度从高到低的返回值的序列。
- _ACTION_AUTHORIZERS表示提交服务请求(直接支持请求)的主体的名字。
- _APP_DOMAIN规定了应用域的名称。

2. 安全策略和信任证的表达

KeyNote要求信任证和安全策略由特定的assertion语言编写。通过指定特定的assertion语言，KeyNote提高了互操作性和处理的效率，并且可以使语义良好的信任证和安全策略得到广泛使用。

KeyNote沿用了PolicyMaker的信任证直接授权的思想和原则，安全策略和信

任证也统称为断言。断言由若干域组成：

- Authorizer域表示授权者即断言的签发者；
- Licensees域表示权限接受者，权限接受者可以是一个或多个主体，例如电子书店对某大学的所有学生买书打折；Licensees域的多个主体可以通过“&&”、“||”运算符连接，从而支持分散式的权限管理。
- Conditions域表示授权条件，通常是上下文环境，例如，系统时间等；Conditions是定义在行为属性集上的谓词，它利用“->”连接授权限制条件和返回值，授权限制条件是标准的正规表达式并可以嵌套，返回值缺省为_MAX_TRUST。
- Signature域表示数字签名。

3. 信任证的收集和验证

KeyNote系统中，信任证依旧由应用系统收集。与PolicyMaker不同的是数字签名的验证由KeyNote中信任管理引擎负责。

4. 一致性证明算法的单调性

KeyNote坚持单调的证据观点。单调性要求证书只能累加而不能去除证据，即没有撤消授权的观点。在此限制下，安全策略和信任关系可以表示成一个有向图(W，V，f)。W是有向图结点的集合，每个结点代表一个或多个主体；v是边的集合，边代表主体间的信任关系；权f给出信任条件。

在一致性证明之初，应用程序输入给KeyNote的参数是：

- 信任证列表：包括所有的信任证；
- 安全策略：定义了资源的保护策略；
- 服务请求者的公钥：用以验证信任证签名的有效性；
- 行为上下文：属性名/属性值的对应列表，包含与服务请求的行为和必要的信任判定有关的信息。行为上下文中的属性和属性值的指定必须反映应用系统的安全需求。因此选择哪些属性包含在action-enviroment中是集成KeyNote到应用程序的最关键工作。

KeyNote的一致性证明验证算法是一种深度优先算法。其主要思想是采用递归的方式试图查找到至少一条能够满足请求的策略断言。所谓满足一个断言，即该断言的Condition字段和Licensees字段同时得到满足。KeyNote中断言程序运行时，也能根据断言的满足情况生成类似于PolicyMaker的接受记录，但该记录仅被KeyNote的验证模块内部使用，对其他断言程序不可见。最终，当由请求、断言和断言间的证明关系所形成的图被构造出来时，该请求被证明。对比于PolicyMaker，KeyNote一致性验证算法对输入的断言要求更严格，因此，KeyNote一致性验证算法实际上解决的PoC问题仅是PolicyMaker的一个子集。

具体的证明过程和算法参见相关的文献。

7. 2.4 KeyNote 模型安全性分析

KeyNote模型作为 PolicyMaker模型的升级和改进版本，在标准化和与应用程序集成方面都有了较大地改进。

(1) KeyNote 提供一种专门的语言以描述安全策略和信任证，KeyNote 语言简练并功能强大，同时和一致性检查算法紧密结合在一起，并且负责信任证的可靠性验证。这减轻了应用系统的负担，使 KeyNote 更容易与应用系统集成；

(2) 有利于安全策略和信任证描述格式的标准化，使应用系统能够更有效地

传播、获取以及使用安全策略和信任证。

7.3 RT 模型

7.3.1 基于属性的信任管理应用背景

首先来看这样一个应用实例。假设Alice是A公司的管理员，Bob是B公司的管理员，因为A公司和B公司有业务上的来往，Alice希望可以授权给B公司的工程师对其某些资源进行访问，A公司的管理员Alice对B公司的组织结构和人员情况并不了解，Charlie宣称他是B公司工程师，在这种情况下Alice如何才能够允许Charlie对本地资源的访问？

如果采用7.2介绍的PolicyMaker和KeyNote的技术，则需要由Alice发布信任证将访问A公司资源的权限授权给Bob，而Bob再发布新的信任证进一步将权限授权给Charlie，最终Charilie获得对A公司资源的访问权限。在这样实现跨域访问的过程中，委托权限的介质就是信任证。信任证直接包含对权限的描述，每一个信任证将一些权限从发布者委托给主体。请求者访问资源时，资源拥有者根据请求者具有的相关信任证集，判定请求者是否有权访问资源。因此，PolicyMaker和KeyNote技术可称为基于能力的信任管理系统。此类技术一个缺点是不能在完全陌生的实体之间建立信任关系，必须通过信任证进行授权传递。

本节介绍的基于属性的信任管理系统，能够根据主体所具有的安全属性进行访问判定，可以解决完全陌生实体之间的授权访问需求。下文介绍采用基于属性的信任管理系统，如何解决上文应用实例中的跨域访问控制问题。

若采用基于属性的访问控制技术，A公司将修改安全策略，把资源的访问权限与实体的属性相关联。比如，定义安全策略为“具有B公司的工程师属性的主体具有本地资源的访问权限”。在B公司中，Bob可以颁发信任证给Charlie，证明他是B公司的工程师。在这里，“B公司的工程师”可以看作是Charlie所具有的安全属性。在Charlie试图访问A公司时，可以提交信任证给A公司，A公司将验证信任证的有效性，并检查信任证中是否授予Charlie具有B公司的工程师属性。若具有该属性，则允许其访问，反之则拒绝该访问请求。

在这种安全机制下，不需要B公司的管理员Bob参与到每一项具体的授权过程，其它公司对Charlie的授权也不需要Bob的直接参与，Bob只需要声明哪些员工是B公司的工程师即可，具体的授权策略由拥有相关资源的主体制定。因此大大减轻了安全管理员的管理负担。

7.3.2 基于属性的信任管理系统的基本概念

这种基于主体被认证的属性进行访问判定的系统，是基于属性的访问控制(Attribute Based Access Control，简称ABAC)，它简化了多域环境下的授权管理，并且支持分散和可伸缩性的属性授权。

与基于能力的信息管理系统相比，在这种系统中，信任证中表达的是发布者声明主体具有什么属性，信任证所委托的权限是通过属性间接表达的，并不特定于某个应用。当完全陌生的请求者访问资源时，可以验证主体拥有的相关信任证，检查请求者是否具有相应的属性，从而进行访问判定。一般而言，基于属性的信任管理系统需要具有下列表达安全策略的能力：

(1) 分散式的属性：一个实体可以声明另一个实体具有相应的属性。例如工商管理部门宣称某公司是合法的商业实体；

(2) 基于属性的委托：一个实体可以将对某属性的声明权委托给另一个实体，即一个实体信任另一个实体对此属性的判断。例如国家级的工商管理部门将宣称公司是合法商业实体的权限委托给省级的工商管理部门；

(3) 属性的推导：基于一个实体具有的某个属性可以推理出其具有的其他属性。例如企业声誉评定联盟宣称某公司是行为良好的公司，前提是该公司是合法公司。注意，这与(2)的不同在于，这里有两个属性“合法公司”和“行为良好公司”，从“行为良好公司”这一属性推理出“合法公司”属性。而在(2)中，只有一个属性“合法公司”。

(4) 值属性：对属性通常需要设置一些参数。例如营业执照中可以对经营内容、级别和营业时间等属性进行定量声明。

7.3.3 RT 模型简介

RT是一族基于角色的信任管理语言，RT的设计目标是建立一个表达力强且逻辑清晰的系统，具有直观的、正式定义的、易处理的语义。它结合了基于角色的访问控制和信任管理系统两者的优点，更具表达力且简洁直观。

基于属性的访问控制是一种有效的跨域授权方式。RT使用角色的概念来表示属性。RT中的角色定义了一个实体集合，其中的实体是这个角色的成员。角色可以看作一种属性，实体是一个角色的成员当且仅当该实体有角色所标识的属性。这些概念简化了结盟环境下的授权管理，可以表达其它信任管理系统不能表达的策略。比如，在7.3.1介绍的应用实例中。假设在B公司中除了Charlie是工程师外，Dark、Engager也都是工程师；那么“B公司的工程师”的角色就代表了“Charlie、Dark、Engager”这些实体的集合，该集合中的每一个实体都具有“B公司的工程师”这个安全属性。

RT最基本的部分是RT_0，其他模型还包括RT_1，RT_2，RT^T，RT^D。下文首先对RT_0进行概要的介绍。

7.3.4 RT_0 模型基本组件

RT_0是RT的基础，它主要用来定义实体、角色名字和角色。一个实体对应一个用户，一般使用公钥来唯一标识一个主体。实体可以发送信任状和提出访问请求。RT_0要求每个实体可以单独的被验证，而且可以判断哪个实体发布一个特殊的信任状或者访问请求。

通常采用大写字母A、B或C来表示实体。角色名字是一个标识符，通常使用小写字母加下标来表示，比如r_1、r_2等。角色通过实体名加角色名字来表示，实体名字和角色名字之间用逗号隔开。例如在实体A和B中分别定义角色r_1和r_2，可以表示为：A.r_1和B.r_2。

1. RT_0中的信任状

在RT_0中有4种信任状类型，每一类信任证的具体定义和类型如下：

（4） A.r←B

其中A和B是实体，r是角色名。该信任证表示实体A定义B成为A的r角色。按照基于角色访问控制的观点来看，实体B被指派为A.r角色，从而具有该角色具备的所有权限。按基于属性访问控制的观点来看，这个信任状可以看做实体B具有角色属性A.r。

比如，HuBei.university ← Hust，该信任证表达的含义就是华中科技大学“Hust”实体具有湖北的大学“HuBei.university”角色的属性。

（5） $A.r \leftarrow B.r_1$

其中A和B是实体，r和r_1是角色名。该信任证表示实体A定义A.r角色包含所有B.r1角色的成员。换言之，A定义角色B.r1比角色A.r更有权限，即B.r1的成员可以做A.r授权的任何事。按照基于角色访问控制的观点来看，角色B.r1支配角色A.r。从基于属性的特性看，如果B声明实体X具有属性B.r1，那么A声明X具有属性A.r。

若r和r1是相同的角色名，则该信任证表示从实体A到实体B的有关角色r的委托。

比如，HuBei.student ← Hust.student，该信任证表达的含义就是如果一个实体X具有“Hust.student”的属性，那么X也将具有 “HuBei.student”的属性。

若A和B是相同的实体，则该信任证表示在实体A中，能够从属性$A.r_1$推出属性A.r。

比如，HuBei.student ← HuBei.StuID，该信任证表达的含义就是如果一个实体X具有“HuBei.StuID”的属性，那么X也将具有 “HuBei.student”的属性。

（6） $A.r \leftarrow A.r_1.r_2$

A是实体，r、r_1、r_2是角色名，称$A.r_1.r_2$是链接角色。该信任证表示：如果A声明实体B有属性r_1，且B声明实体D有属性r_2，那么A声明D有属性r。按照基于角色访问控制的观点来看，若实体B具有$A.r_1$角色，那么则角色$B.r_2$支配角色A.r。从基于属性的特性看，这是基于属性的委托，A声明实体B具有$B.r_2$的权限不是通过B的标志符，而是另一个属性$A.r_1$。

比如，HuBei.student ← HuBei.university.stuID，这个信任证表达的含义就是如果一个实体Hust具有“HuBei.university”的属性，那么具有角色“Hust.stuID”属性的实体也同时具有“HuBei.student”的属性。

如果r和r_2是相同的，则该信任证表示实体A把声明A.r角色的权限委托具有的A. r_1角色的实体。

比如，工商管理总局.合法公司 ←工商管理总局.省级分支.合法公司，该信任证表达的含义就是，如果实体“湖北省工商管理局”具有“工商管理总局.省级分支”的属性，那么具有“湖北省工商管理局.合法公司”角色属性的实体也同时具有“工商管理总局.合法公司”的属性。该信任证表示基于属性的授权委托，即“工商管理总局”将声明某个公司是“工商管理总局.合法公司”的权限委托给“湖北省工商管理局”。若湖北省工商管理局授予X公司是“湖北省工商管理局”的“合法公司”，则该公司也相应的成为“工商管理总局”的“合法公司”。

（7） $A.r \leftarrow f_1 \cap f_2 \cap \ldots \cap f_k$

A是实体，k是一个大于1的整形数，每个f_j，$1 \le j \le k$，是实体、角色或者是起始于A的链接角色。$f1 \cap f2 \cap \ldots \cap fk$是个交集。该信任证表示任何拥有所有属性$f_1$、$f_2$、…、$f_k$的成员拥有属性A.r。

比如，Hust.csstudent ← Hust.stuID ∩ CS.student，这个信任证表达的含义就是如果一个实体具有“Hust.stuID”和“CS.student”的属性，那么该实体也将具有角色“Hust. csstudent”的属性。

2. RT_0中的角色操作函数

RT_0提供了丰富的角色操作函数，包括：

(1)成员函数members(role expression)：确定角色表达式所包含的成员集；

(2)实体函数base(role expression)：确定角色表达式所对应的实体集，如base(A.r_1.r_2) = A,base(B_1.r_1 ∩ …… ∩ B_k.r_k)={B_1,……, B_k}；

(3)实体函数Entities(C)：确定信任证集C中的实体集，Entities(C)与base(role expression)的区别在于针对的对象不同而已；

(4)角色函数Roles(C)：确定信任证集C中的角色集；

(5)角色名称函数Names(C)：确定信任证集C中角色名称集。

7.3.5 RT_0 模型实例分析

例7-2 eBookMarket是一个电子书店，可向某大学College的学生提供购书优惠服务。College将发放学生证的权限委托给学位办公室(CredOffice)，且CredOffice为学生Alice发放了学生证。当学生Alice向电子书店提出打折请求时，电子书店需要进行控制决策，根据学生Alice提供的信任证集确实是否为其打折。

1. 实例中包含的实体

在该实例中体现出的实体包括有电子书店 eBookMarket、大学 College、学位办公室(CredOffice)、学生 Alice，这些实体分别简写为：B、C、CO、A。

2. 实例中包含的角色

在该实例中体现出的角色包括有电子书店的折扣角色、大学 College 的学生角色、学位办公室(CredOffice)的学生角色，这些角色分别简写为：B.discount、C. student、CO.student。电子书店将打折的访问权限指派给角色 B.discount，具有该角色属性的用户就可以享受打折的优惠。

3. 实例中的信任证

在该实例中，用 RT0 语言描述的信任证如下：

（1）B.discount←C.student

该信任证表达了电子书店 eBookMarket 将赋予 College 的所有学生购书打折优惠的权利；

（2）C.student←CO.student

该信任证表达了大学 College 将发放学生证的权限委托给学位办公室(CredOffice)；

（3）CO.student←Alice

该信任证表达了学位办公室(CredOffice)为学生 Alice 发放了学生证；

4. 一致性证明算法

根据上述的实体、角色和RT0语言描述的相关信任证集，针对该实例的一致性证明必须回答这样的问题“实体Alice是否拥有B.discount的属性？”

从逻辑上看，信任证（1）、（2）、（3）通过不断的委托授权便构成了一个信任证链，能够证明实体Alice拥有B.discount的属性，从而可以享受电子书店打折的优惠。一致性证明通过查找实体A具有的所有角色集合R，判断角色属性B.discount是否属于该集合R，可以得到正确的决策结果。

但是在分布式环境下，如何进行信任证链的查找过程，是实施RT_0模型一致性证明的关键性技术。

7.3.6 信任证的分布式存储和查找

在上述的实例中，从逻辑上看，信任证(1)由eBookMarket保存，信任证(3)由Alice保存，而信任证(2)则由任意用户保存。Alice需要购买书籍，为了享受优

惠，他提交了信任证(3)给eBookMarket。eBookMarket查看信任证，发现Tom 的信任证是由CredOffice发布的，于是发送查询请求到CredOffice以核实(3)的有效性；CredOffice又将该请求转发到College等。于是，信任证通过不断的委托授权便构成了一个信任证链。

多域环境下信任证的存储方式决定了信任证链发现的关键性技术。一个合理的信任证存储方案，不仅要满足存储信任证的基本要求，而且要保证对信任证高效的访问。

在一致性证明过程中，需要查询到信任证链中的所有信任证，对信任证的有效性进行核实，以保证一致性证明的正确性和完备性：所谓正确性，是指一致性证明能够正确执行，不会出现过程错误而导致证明失败；所谓完备性，是指不应该存在由于信任证收集不完整而导致证明失败的情况。而查找信任证时，需清楚信任证的存储位置，这样才能做到有的放矢，提高协商效率。具体的存储和查询算法，可以参考相关的研究文献。

7.3.7 RT_0 模型的扩展

RT_1相对于RT_0增加了参数化角色，为了约束信任证中参数的取值范围，通常定义为不同的数值集。参数的数据类型包括整型、闭枚举型、开枚举型、浮点型、日期和时间型。此外，RT_1还定义了如何将信任证转化为逻辑规则，使得系统具有较强的处理能力。

RT_2相对于RT_1增加了o-sets的概念，即逻辑客体。逻辑客体能够将逻辑相关的客体组合在一起，使得关于它们的允许可以一起委派。RT_2中的信任证可以是一个角色定义信任证，也可以是一个o-set定义信任证。与RT_1相比，RT_2中的信任证在下面两个方面更具一般性：

- 一个类型变量能被基类型的动态数值集约束，如角色或o-sets；
- 变量的安全需求被放宽，一个变量是安全的，当它满足以下三个条件：①变量出现在信任证中的一个角色名或o-set中；②变量被一个角色或o-set约束；③变量出现在一个约束另一个变量的一个角色或o-set中。这极大的增强了RT2的表达能力。

RT^T提供了多重角色和角色乘积操作符，通过使用角色交集，能够表达门限机制和职责分离(Separation of Duty,SoD)。RT^D提供角色激活的委托，可以表达权力(capacity)的选择使用和这些权力的委托。

这些模型的详细内容可以参考相关的安全文献。

7.3.8 RT 模型的安全性分析

针对跨越安全域进行访问的安全问题，RT模型提供了一种基于角色的信任管理框架，能够在陌生主体之间，通过主体拥有的安全属性而建立信任关系。它使用角色的概念来表示实体的安全属性，利用基于角色的委托来表达实体之间的授权传递关系，综合了基于角色的访问控制和信任管理系统两者的优点，更具表达力且简洁直观，代表了信任管理的最新研究水平。

与其他基于信任管理的访问控制技术相比，RT 模型具有以下优势。

基于角色的访问控制系统通过角色与用户、角色与权限的配置为系统实现其安全控制提供了灵活且强有力的保护。这种保护可适用于多种不同的安全需求，获得广泛的应用。RT 模型采用 RBAC 技术，将角色看作实体的安全属性，实现对跨域陌生实体的基于属性的授权。鉴于 RBAC 技术的优点，RT 模型的语义简

洁直观、易于处理，而且表达安全策略的能力强。

相对于 7.2 节的 Policy 和 KeyNote 模型等基于能力的信任管理系统而言，RT 模型采用基于属性的权限委托，实体具备的属性并不特定于某个应用。当完全陌生的请求者访问资源时，通过验证主体拥有的相关信任证，检查请求者是否具有相应的属性，从而进行访问判定。

7.4 自动信任协商

7.4.1 自动信任协商应用背景

在基于服务为中心的网格计算(grid computing)环境、应急处理、供应链管理和在线服务等具有多个安全管理自治域的应用中，为了实现多个虚拟组织间的资源共享和协作计算，需要通过一种快速、有效的机制为数目庞大、动态分散的个体和组织间建立信任关系，而服务间的信任关系常常是动态地建立、调整，需要依靠自动协商方式达成协作或资源访问的目标，并能维护服务的自治性、隐私性等安全需要。

7.3节所介绍的RT模型基于第三方的主体发布的安全属性，实现为陌生实体授权的安全目标。它提供了一种基于角色的信任管理框架，能够在陌生主体之间，通过主体拥有的安全属性而建立信任关系。由于信息安全的隐患源于多个方面，在对陌生方建立信任所依赖的属性信任证和访问控制策略中，都可能泄露交互主体的敏感信息，特别是陌生方之间很难对彼此信任的第三方达成共识，来协助它们建立信任关系。

自动信任协商(Automated Trust Negotiation，简称 ATN)就是为解决陌生实体之间自动建立信任关系的问题，而产生的新的访问控制技术。通过信任证、访问控制策略的交互披露，资源的请求方和提供方能够自动地建立信任关系。它与传统访问控制的主要区别在于协商双方是否事先知道对方身份、拥有的权限和访问控制策略。而在自动信任协商里，通过在请求方和服务方之间逐步披露属性证书，最终建立信任关系，协商过程一般强调自动化，不需要或者需要少量的人工参与。

7.4.2 自动信任协商主要研究内容

1. 自动信任协商的安全需求

在不同安全域的陌生实体之间建立动态的信任关系通常面临着如下的安全需求：

(1) 当隶属两个独立安全域的陌生主体进行资源访问时，如何提供一种有效的机制以动态地建立两者的信任关系；
(2) 当开放网络中的协商主体在维护其自治性和隐私性时，需要设置何种访问控制策略；
(3) 对资源的访问控制结论不再单纯是Yes 或No，需要根据各自的协商策略给出相应提议，来实现进一步的协商，既要实施各自的信息保护，又要建立协作的信任关系，如何建立协商策略机制以兼顾二者的要求；
(4) 信任建立将依赖于一套完整的信任协商协议，该协议具体体现为实体之

间交互披露消息的过程。

2. 自动信任协商的主要组件

针对上述的安全需求，ATN 的主要组件主要涵盖下列方面：信任协商的基础模型与框架、访问控制策略和信任证描述、协商策略(Strategy)。

(1) ATN模型与框架

信任协商模型是信任双方在建立信任关系中所采取的披露信任证书和访问控制策略的方式。ATN 的抽象模型体现为一条信任证披露序列(Credential Disclosure Sequence)。设请求方信任证书集为 ClientCreds，提供方信任证书集为 ServerCreds，协商的信任证披露序列定义为 $\{C_i\}_{i\in[0,2n+1]} = C_0, C_1, \cdots, C_{2n+1}$，其中 $n \in N$，$C_{2i} \subseteq ClientCreds$，$C_{2i+1} \subseteq ServerCreds$。

该模型较为简洁，其主要意义在于对双方信任建立的过程进行了详细的描述，体现了服务请求方和提供方进行信息交互的过程。

根据 ATN 的抽象模型，实现信任协商的典型框架如图 7.3 所示。

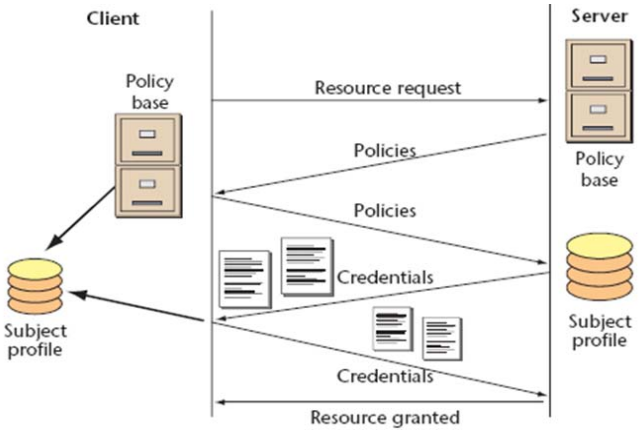


图 7.3 典型的信任协商框架

在图7.3中，服务请求方（Client）主动发出访问请求，服务提供方（Server）根据本地的安全策略做出相应的回应，要求Client提供相关的信任证；然后Client根据己方的安全策略作出回应；随之，Client和Server逐步向对方披露信任证，最后达成互信，自动的建立信任关系。

(2) 访问控制策略与信任证描述

访问控制策略是ATN的关键组成部分，它规定了访问保护资源所需提供的信任证书集。根据描述的复杂程度，访问控制策略可分为简单策略(元策略)与复合策略，简单策略是组成复合策略的基本元素。访问控制策略一项关键特征是强调单调性(monotonicity)，因为在分布式广域协作环境中，很难判断某实体不拥有某种信任证，所以单调性就是只能强调实体具有哪些信任证。

信任证是一种由发布者实体签名的数字凭证，它表示发布者相信该证书的拥有者具备某些能力、属性或者性质。具体含义参见 7.1.2 节中的描述。

(3) 信任协商策略

信任协商策略意义在于控制信任关系的合理建立，并提出建立信任关系必须满足的三项约束条件，即可完成性、可结束性和高效性。根据该约束条件，ATN 信任的建立过程主要采用下列两种信任协商策略。

- 积极(eager)策略：要求协商方在接收到对方披露的信息后，披露所有可满足访问控制策略保护的信任证书；
- 谨慎(parsimonious)策略：协商双方在披露足够的访问控制策略后才会披露所需的信任证书。

积极策略往往会披露过多与信任建立无关的信任证书，而在谨慎策略中，协商者从安全策略出发，按照严格受控的方式，通过交换指定的访问控制策略，尽可能地减少无关信任证书的披露。这两种协商策略控制的协商交互次数与两方持有的信任证书数量呈线形关系。

7.4.3 自动信任协商实例分析

例 7-3 在线电子书城系统中，电子书店只对注册了的用户提供浏览书目以及订购书籍服务。用户 Alice 试图访问电子书店 eBookMerket，整个信任协商过程如图 7.4 所示：

1. Alice 向 eBookMarket 发送访问请求；
2. eBookMarket 制定了策略“只对注册了的用户提供服务”，所以 eBookMarket 向 Alice 发送披露包含了注册信息的证书(C1:register)的请求；
3. 同时，Alice 制定策略“访问注册证书 C1 的主体必须首先披露营业执照”，所以 Alice 向 eBookMarket 发送披露营业执照(C2:businesslicense)的请求；
4. eBookMarket 认为营业执照 C2 是可以被任意用户查询验证的，所以 eBookMarket 披露营业执照 C2；
5. Alice 接收到 eBookMarket 向 Alice 披露的营业执照 C2 后，C1 的保护策略被满足，Alice 向 eBookMarket 披露 C1；

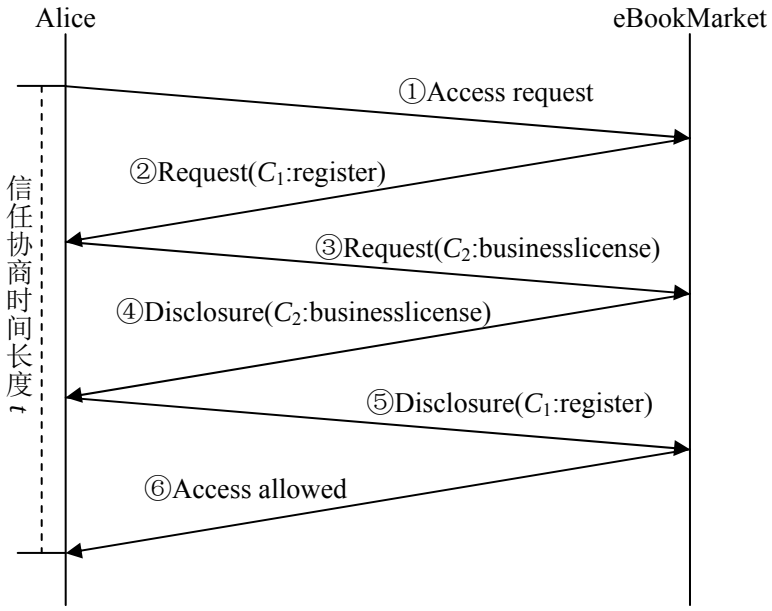


图 7.4 注册用户与电子书店的协商过程

6. 最后，Alice 的访问被批准。 整个信任协商过程经历的时间长度为 t。

在上述实例中，信任证的披露序列为{C2,C1},其中 C2 属于服务方证书集；C1 属于请求方证书集。服务方的安全策略体现为“只对注册了的用户提供服务”，

即要求请求方提供注册信息的证书(C1:register)；请求方的安全策略体现为“访问注册证书 C1 的主体必须首先披露营业执照”，即要求服务方提供营业执照(C2:businesslicense)。交互双方采用的协商策略为谨慎策略，即协商双方在披露足够的访问控制策略后才会披露所需的信任证书。

7.4.4 自动信任协商敏感信息保护

在 ATN 中，信任证和访问控制策略是特殊的资源，可能包含敏感信息。为了避免信息泄露，防止推理攻击，需要受到相应的保护。

1. 敏感信息分类

根据敏感信息内容的不同，可以将访问控制策略和信任证涉及的敏感信息分为两大类。

- 资源的内容敏感，属于显式敏感信息，包括访问控制策略本体和信任证中的某些属性值，例如，具体的年龄不能随意泄露；
- 资源的拥有敏感，属于隐式敏感信息，协商方的响应和信息流动会隐式地暴露其保密信息，例如，某个实体是否是 FBI 的成员属于敏感信息，如果某恶意者宣称拥有某资源，并设定资源的访问控制策略为：访问此资源需要 FBI 成员资格，即便实体不会直接提交 FBI 证书给此恶意者，但他也能根据访问请求推断出实体可能是 FBI 的成员。

2. 针对敏感信息的推理攻击

针对上述敏感信息，存在着推理攻击。所谓推理攻击，即当访问者拥有满足访问控制策略的证书时，其行为和表现与没有证书有很大的差别，协商的另一方可以根据在协商过程中的反应推测出该用户是否具有某些信息，以及哪些信息是敏感的。例如，某用户在公司的具体职位，可以通过他的薪金水平推测出来，如果该用户认定职位是隐私信息的话，那么就存在着隐私信息的泄露。推理攻击主要有下列四种类型：

(1) 向前肯定推测：假设攻击者 O 知道在协商者 M 与 P 之间存在 A.t←B.r。O 通过向 M 询问是否满足 B.r，若 M 具有属性 B.r，则 O 可推测 M 满足 A.t 的需求；

(2) 向前否定推测：假设攻击者 O 知道在协商者 M 与 P 之间存在 A.t←B.r。O 通过向 M 询问是否满足 B.r，若 M 不具有属性 B.r，则 O 可推测 M 可能亦不满足 A.t 的需求；

(3) 向后肯定推测：假设攻击者 O 知道在协商者 M 与 P 之间存在 A.t←B.r。O 通过向 M 询问是否满足 A.t 的需求，若 M 满足 A.t，则 O 可推测 M 可能具有 B.r 的需求；

(4) 向后否定推测：假设攻击者 O 知道在协商者 M 与 P 之间存在 A.t←B.r。O 通过向 M 询问是否满足 A.t 的需求，若 M 不满足 A.t，则 O 可推测 M 亦不具有 B.r 的需求。

3. 敏感信息的保护策略

为了解决推理攻击，达到保护信任证中敏感属性的目标，有关文献提出了 ACK 策略。ACK 策略的目标在于，协商者在没满足 ACK 策略前，是不清楚对方是否具有某些属性。ACK 策略的原理在于，对于需要保护的证书，使用 ACK 函数进行标记，在协商者没有满足本地的安全策略以前，不暴露标记的证书或证书中的信息，达到敏感信息保护的目标。

4. ACK 策略的实例分析

下面，通过实例 7-4 来说明 ACK 策略的工作原理。

实例(7-4) 安全医疗中心(SMC)是一个医疗供应机构，可提供可医疗打折服务。享受折扣的机构必须是由福利公司(ReliefNet)所承认的单位或个人。Alice 是一家慈善机构(MedixFund)的员工，该机构属于 ReliefNet 所管辖。

为了表示 Alice 能够享受到药品的打折优惠，我们使用 RT 模型中的信任证来描述为 Alice 授权的过程。

(a) MedixFund.Clerk←Alice

该信任证表示 Alice 是 MedixFund 的员工；

(b) ReliefNet.coamember←MedixFund

该信任证表示 ReliefNet 相信 MedixFund 是所承认的单位；

(c) SMC.partner←ReliefNet.coamember；

该信任证表示 SMC 相信 ReliefNet 福利公司；

(d) SMC.diSCount←SMC.partner.clerk。

该信任证表示 SMC 将享受打折权限的委托给 ReliefNet.coamember。

经过上述的授权过程，现在可以确定 Alice 能够享受到 SMC 的打折优惠。现在信任证(a)由 Alice 自己保管；信任证(b)、(c)、(d)由 SMC 保管。那么 Alice 和 SMC 协商的过程如图 7.5 所示。

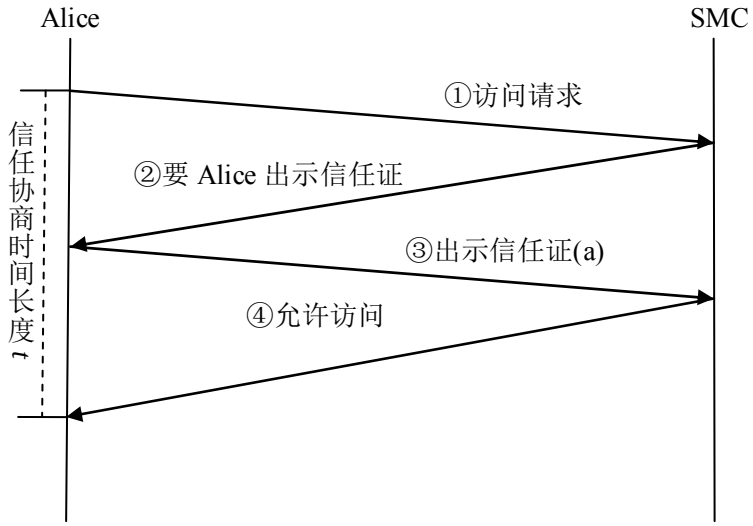


图 7.5 Alice 与 SMC 的协商过程

在图 7.5 中，Alice 和 SMC 的信任协商是这样的。

1. 表示 Alice 向 eBookMarket 发送访问请求；
2. 表示 SMC 要求出示相应的信任证，证实 Alice 是“由福利公司(ReliefNet)所承认的单位或个人”；
3. 表示 Alice 向 SMC 提交信任证(a)；
4. 表示 SMC 在验证 Alice 的信任证满足本地的安全策略后，允许 Alice 享受打折请求。

现设 Alice 认为其员工身份包含敏感信息，只能披露给政府授权的且名声较好的机构，不是对任意的机构都能够披露的。即证书(a)的披露必须建立在一定的信任关系上。在使用 ACK 策略后，证书(a)的访问控制策略描述可描述为：

(e)ACK：Alice[(a)]=Gov.goodIndeed。

该 ACK 策略表示信任证（a）只能披露给 Alice 政府授权的且名声较好的机构。

为了确保协商的顺利进行，SMC 将提供证书；

(f)Gov.goodIndeed←SMC；

该信任证表示 Gov 声明 SMC 是政府授权的且名声较好的机构。

加入 ACK 策略后，Alice 和 SMC 的信任协商过程将改变为如同 7.6 所示。

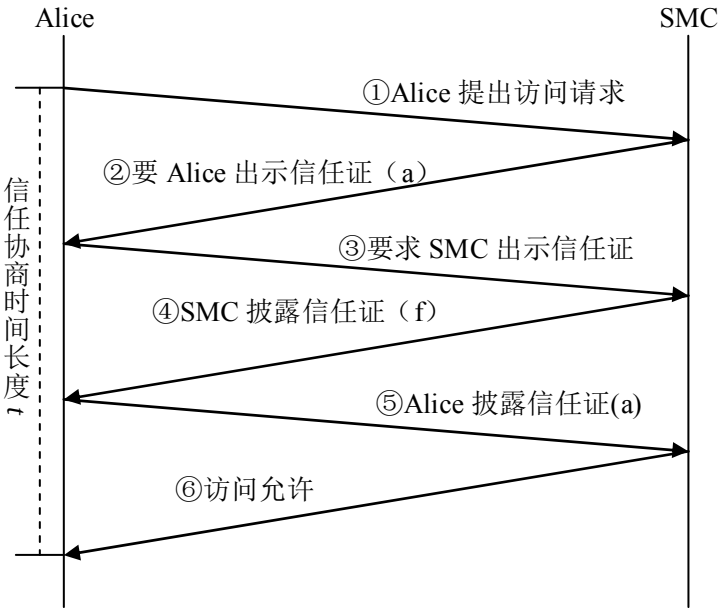


图 7.6 加入 ACK 策略后 Alice 与 SMC 的协商过程

在图 7.6 中，增加 ACK 策略以后，Alice 和 SMC 的信任协商过程发生了相应的改变。

1. 表示 Alice 向 eBookMarket 发送访问请求；
2. 表示 SMC 要求出示相应的信任证，证实 Alice 是“由福利公司(ReliefNet)所承认的单位或个人”；
3. 表示 Alice 根据本地 ACK 策略，要求 SMC 提供信任证，证实 SMC 是“政府授权的且名声较好的机构”；
4. 表示 SMC 披露信任证（f）给 Alice；
5. 表示 Alice 在验证信任证（f），满足本地 ACK 安全策略后，提供信任证(a)给 SMC；
6. 表示 SMC 在验证 Alice 的信任证满足本地的安全策略后，允许 Alice 享受打折请求。

从上面的例子看来，ACK 策略是通过增加访问控制策略，以达到建立更高级别的信任关系后，再释放一些重要的信息，从而避免了攻击方的非授权推测。

此外，针对信任证中敏感信息保护问题，还有学者提出了隐藏证书和零知识证明协议等概念，隐藏证书基于椭圆曲线加密的原理，对证书中敏感信息进行机密性和完整性保护，可以防止敏感信息的泄露。具体内容可参考相关研究文献。

7.4.5 自动信任协商安全性分析

自动信任协商通过使用访问控制策略提供了一种方法来规范敏感证书和敏感策略的交换，从而保护了用户的敏感证书信息、敏感访问控制策略和个人隐私。当访问者与资源/服务提供方不在同一个安全域时，自动信任协商则可为合法用户访问资源提供安全保障,防止非法用户的非授权访问。但现有的 ATN 模型存在一些不足：

1. 缺乏对相关概念统一的安全定义

自动信任协商中没有对信任证、访问控制策略、协商者等给出安全定义,即没有回答“什么样的证书,才算是安全的”、“什么样的访问控制策略才可提供安全保障”等问题。一般地，为了提高证书的安全级别，学者们增加了签名算法的研究力度；为了保证访问控制策略的安全，则规范访问控制策略的披露过程，对敏感信息的释放进行严格限制等。但高安全级别的证书，增加了加密/解密的开销，访问控制的谨慎披露策略，降低了协商效率。

2. 访问控制策略描述语言研究需要加强

访问控制策略是ATN 中的一个重要研究内容，时至今日，有关策略语言的研究仍属于一个热点。ATN 的广泛应用需要表达力强、计算效率高而且易于理解和处理的策略语言来支持，策略语言不仅局限于对信任证属性的约束，而且要能够表达跨安全域间复杂的信任关系。同时，在访问控制策略动态变化情况下，需要研究其对协商策略的影响，即二者间互相影响的联动关系。

3. 信任决策模型需要综合考虑各种因素

现在的ATN的信任决策模型仍凸显单薄。在现实社会中,陌生方之间信任的确立往往需要凭借主观和非主观信任相结合的方式。主观信任模型的研究主要侧重于从主观性入手研究信任的数学模型，解决信任的表述、度量、推导和综合运算等问题。因此，研究一种结合主观信任和客观安全策略相结合的信任决策模型将为跨域协作提供更有力的理论基础。

习题七

1. 试列举一个信任管理系统的应用实例，并且对照信任管理系统的基本组件，说明信任管理系统的运行框架和大致流程。

2. 试列举一个 Policy 模型的应用实例，列出其安全策略、信任证断言和一个具体的访问请求，使得其一致性证明算法是不可判定的。

3. 对比 dRBAC 模型与 RT_0 模型的，说明他们之间有何相同点，有何不同的地方。

4. 列举一个 ATN 的应用实例，并给出 Client 与 Service 建立信息关系的过程，要求协商双方至少都要定义一条 ACK 策略。

第 8 章 权限管理基础设施

在分布式多域环境下，基于角色映射的安全互操作技术要求各个安全域都必须采用基于角色的访问控制策略，并且各域的角色层次结构具有一定的相似性，因此不适合异构的复杂多域环境。基于信任的分布式访问控制技术虽然能适应访问控制策略各不相同的多域环境，但是其访问控制的粒度较粗，也容易出现信任欺诈方面的问题。在实际应用中，人们往往需要一种能够既能适应于分布式异构多域环境，又能提供细粒度访问控制的技术，并希望这种访问控制机制与具体应用系统和管理无关，能简化应用中访问控制和权限管理系统的开发，降低维护成本。为此，权限管理基础设施(PMI，Privilege Management Infrastructure)应运而生。PMI也称为属性特权机构，是基于属性证书的权限管理平台，PMI将用户的权限信息以数字证书的形式保存，并由统一的权威机构分发，特别适用于企业级多应用系统的权限管理。PMI通常采用公开密钥技术，以公钥基础设施(PKI，Public Key Infrastructure)为基础进行构建。本章首先将简要介绍PKI相关基础知识，然后着重分析和说明PMI的意义，作用，原理和应用情况。

8.1 公钥基础设施

8.1.1 构建公钥基础设施的必要性

随着计算机和网络技术的迅猛发展，人们迎来了互联网时代。与此同时，基于互联网的计算机信息系统安全问题也成为国内外关注的焦点。近年来，非法侵入计算机网络系统进行信息窃取、系统破坏等“黑客”行为不断出现，犯罪技术手段也越来越高明，直接威胁着各行各业计算机系统的正常运行，给个人和社会带来了巨大的经济损失，甚至严重威胁着国家的利益。

网络安全问题的根本原因在于通信信道的不安全。由于目前广泛使用的TCP/IP 协议在设计之初，未充分考虑安全方面的因素，使得通信中的数据包很容易被监听、截获甚至篡改。为了解决这种安全问题，可以应用加密技术对数据包进行保护。然而，在这种不安全的信道上如何交换加密密钥又成了一个新的问题。公开密钥技术的出现，为解决密钥交换提供了一种有效方案。在这种方案中，

每个通信用户均被分配一对密钥，其中用于加密的密钥可以公开，被称为公开密钥(或公钥)，通常记为 K_{pub}；用于解密的密钥不可公开，被称为秘密密钥(或私钥)，通常记为 K_{prv}，用 E 和 D 分别表示加密和解密算法，P 和 C 分别表示明文和密文，经过精心设计，有：

$$E_{K_{pub}}(P)=C, D_{K_{prv}}(C)=P$$

由算法特点可以知道，如果要给某用户发送消息，可以利用他的公钥对消息进行加密，这样只有持有对应私钥的用户才能正确解密出该消息。

公开密钥技术很好地解决了密钥交换问题，不过在实际应用还会有新的问题，比如怎样分发和获取用户的公钥？如何建立和维护用户与其公钥的对应关系？获得公钥以后如何鉴别该公钥的有效性？通信双方如果发生争议如何仲裁？为了解决上述问题，就必须有一个权威的第三方机构对用户的公私密钥进行集中管理，确保能安全高效地生成、分发、保存、更新用户的密钥，提供有效的密钥鉴别手段，防止被攻击者篡改和替换。

PKI 是目前建立这种公钥管理权威机构的最成熟的技术。PKI 是在公开密钥的理论和技术基础上发展起来的一种综合安全平台，能够为所有网络应用透明地提供加密和数字签名等密码服务所必需的密钥和证书管理，从而达到在不安全的网络中保证通信信息的安全、真实、完整和不可抵赖的目的。其必要性体现在两个方面：

首先，PKI 是确保用户身份的惟一性的需要。PKI 通过一个证书签发中心(CA)为每个用户(包括服务器)颁发一个数字证书，用户和服务器、用户和用户之间就可以通过数字证书相互验证对方的身份。

其次，PKI 是管理海量用户的需要。PKI 采用 LDAP(Lightweight Directory Access Protocol)技术管理用户， LDA 是目录服务在 TCP/IP 上的实现，对 X.500 的目录协议进行了移植和简化，可以轻易管理百万级用户。

总之，利用 PKI 可以方便地建立和维护一个可信的网络安全平台，从而使得人们在这个无法直接相互面对的环境里，能够确认彼此的身份和所交换的信息，能够安全地从事各种交互活动。

8.1.2 数字证书

PKI 使用数字证书将用户的身份信息和公钥绑定在一起，是实现用户身份验证的关键数据。在实际运用中，数字证书是各类实体(持卡人/个人、商户/企业、网关/银行等)在网上进行信息交流及商务活动的身份证明，在电子交易的各个环节，交易的各方都需验证对方证书的有效性，从而解决相互间的信任问题。证书是一个经证书认证中心数字签名的包含公开密钥拥有者信息以及公开密钥的文件。因此，证书的权威性取决于证书颁发机构的权威性。

(1) 数字证书结构

简单的说，数字证书是一段包含用户身份信息、用户公钥信息以及身份验证机构数字签名的数据。身份验证机构的数字签名可以确保证书信息的真实性。数字证书的形式有很多种，由于 PKI 必须适用于异构环境，所以证书的格式在所使用的范围内必须统一。其中使用最为广泛的是遵循 X.509 标准的数字证书 v3 版本。X.509 制定的标准的证书结构如图 8-1 所示：

版本号
序列号
算法标识
发布者
有效期
主体
主体的公开密钥
签发者的唯一标识
主体的唯一标识
扩展域
签名

图 8-1 X.509 数字证书结构

其中，

版本号——表示证书采用的版本号，决定了证书的格式

序列号——证书在 CA 中的唯一性标识

算法——用来对证书进行签名的算法(如 DSA)，以及算法所需的参数

发布者——CA 中心的名称

有效期——用于标识证书有效期的一对时间值

主体——用户名、国家、邮件地址等

主体的公开密钥——公开密钥值、公开的密钥算法(如 RSA)、算法的参数

签发者的唯一标识——标记 CA

主体的唯一标识——标记用户

扩展域——用于存放附加信息或扩展证书功能，保持兼容性

签名——CA 中心对该证书内容的签名

(2) 数字证书的应用

现有用户甲通过互联网向用户乙传送消息，为了保证信息传送的真实性、完整性和不可否认性，需要对要传送的信息进行数字加密和数字签名，其传送过程如下：

① 甲准备好要传送的消息(明文)。

② 甲对消息进行散列运算，得到一个信息摘要。

③ 甲用自己的私钥对信息摘要进行加密得到甲的数字签名，并将其附在消息尾部。

④ 甲随机产生一个加密密钥(如 DES 密钥)，并用此密钥对要发送的信息进行加密，形成密文。

⑤ 甲用乙的公钥对刚才随机产生的加密密钥进行加密，将加密后的 DES 密钥连同密文一起传送给乙。

⑥ 乙收到甲传送过来的密文和加过密的 DES 密钥，先用自己的私钥对加密的 DES 密钥进行解密，得到 DES 密钥。

⑦ 乙然后用 DES 密钥对收到的密文进行解密，得到明文的数字信息，然后将 DES 密钥抛弃(即 DES 密钥作废)。

⑧ 乙用甲的公钥对数字签名进行解密，得到信息摘要。

⑨ 乙用相同的散列算法对收到的明文再进行一次运算，得到一个新的信息摘要。

⑩ 乙将收到的信息摘要和新产生的信息摘要进行比较，如果一致，说明收

到的信息没有被修改过。

上例中，甲和乙均须获得对方的公开密钥，公开密钥以数字证书作为载体。甲从数据库(或目录服务器)中取得乙的证书，根据 CA 的公钥验证证书的签名，并从中取得乙的公开密钥。

(3) 数字证书的管理

由前节可知，数字证书是保证应用安全的关键。因此，必须对数字证书进行严格有效的管理，确保数字证书在一个受控的环境中被合理的使用，这也是 PKI 系统的核心功能之一，即管理和维护数字证书的申请、生成、使用、验证和撤销。

① 数字证书的申请

数字证书的申请通常是由用户向 PKI 系统的认证中心提出书面请求，由 RA 机构对用户的个人信息进行审核，这一过程通常采用半手工的方式完成，耗时较长。

② 数字证书的生成

RA 中心审核用户个人信息无误后，将个人资料提交 CA 中心，CA 中心为用户的数字证书分配一个唯一的序号，并负责生成高质量的公私钥对，将用户的个人信息和公私钥信息结合在一起，并用 CA 中心的私钥进行签名，形成数字证书。形成的用户公钥数字证书存入证书库，另一方面采用特定的存储媒介(如 IC 卡、USB Key 等)进行保存，分发给用户使用。

③ 数字证书的使用

用户获得数字证书后，即可通过具体的应用系统使用数字证书。公钥数字证书存放在证书库中供其他用户自由读取，签名证书则由用户个人妥善保管。签名证书通常采用附加的安全手段进行保护(如口令)，仅限于本地使用。为了防止公钥的协议攻击，不可用私钥对任意消息进行直接加密，而只对消息的摘要进行加密形成数字签名。

④ 数字证书的验证

甲如果要向乙发送保密消息，首先需要通过网络访问证书库，得到乙的公钥证书。如果这个过程被丁攻击，他可以用自己的公钥替换乙证书中的公钥，从而截获甲向乙发送的消息。因此，用户在使用他人的公钥证书进行加密或验证签名

前，需要确认数字证书的真实性。具体的做法是利用证书发放机构的公钥去验证证书的签名，从而确定证书内容的完整和有效。

⑤ 数字证书的撤销

数字证书在使用过程中，当发生用户私钥泄漏，个人信息更改或数字证书到期等情况时，需要为用户生成新的证书。但是，旧的证书不能简单销毁，否则由旧证书加密的历史数据将无法还原。PKI 系统采用维护 CRL 列表的方式管理作废的证书。对于历史数据，可以使用过去的数字证书进行解密或验证，对于新的数据，则可用更新后的证书进行处理。

8.1.3 PKI 的组成

PKI 由公开密钥密码技术、数字证书、证书发放机构和关于公开密钥的安全策略等基本成分共同组成的。一个完整的 PKI 系统应包括以下基本组件：

- 认证中心 CA
- 注册中心 RA
- 证书库
- 密钥备份及恢复系统
- 证书作废处理系统
- PKI 应用接口系统

(1) 认证中心 CA

认证中心，又称为 CA 中心，作为电子商务交易中受信任的第三方，专门解决公钥体系中公钥的合法性问题。CA 中心为每个使用公开密钥的用户发放一个数字证书，数字证书的作用是证明证书中列出的用户名称与证书中列出的公开密钥相对应。CA 中心的数字签名使得攻击者不能伪造和篡改数字证书。

在数字证书认证的过程中，认证中心作为权威的、公正的、可信赖的第三方，其作用是至关重要的。认证中心就是一个负责发放和管理数字证书的权威机构。同样也允许管理员撤销发放的数字证书，在证书废止列表(CRL)中添加新项并周期性地发布经过数字签名的 CRL。

认证中心的功能：

- 验证并标识证书申请者的身份
- 确保 CA 用于签名证书的非对称密钥的质量
- 确保整个签证过程和签名私钥的安全性
- 证书材料信息(如公钥证书序列号、CA 等)的管理
- 确定并检查证书的有效期限
- 确保证书主体标识的唯一性，防止重名
- 发布并维护作废证书表
- 对整个证书签发过程做日志记录
- 向申请人发通知

(2) 注册中心 RA

注册中心(RA，Registration Authority)，又称为注册审批机构或 RA 中心。RA 系统是 CA 中心的证书发放、管理功能的延伸。它负责证书申请者的信息录入、审核以及证书发放等工作；同时，对发放的证书完成相应的管理功能。发放的数字证书可以存放于 IC 卡、硬盘或软盘等介质中。RA 系统是整个 CA 中心得以正常运营不可缺少的一部分。

注册中心的功能：

- 接收和验证新注册人的注册信息；
- 代表最终用户生成密钥对；
- 接收和授权密钥备份和恢复请求；
- 接收和授权证书吊销请求；
- 按需分发或恢复硬件设备。

(3) 证书库

证书库是一种网上公共信息库，用于证书的集中存放，用户可以从此处获得其他用户的证书和公钥。构造证书库的最佳方法是采用支持轻量级目录访问协议(LDAP，Lightweight Directory Access Protocal)的目录服务系统，用户或相关的应用通过 LDAP 接口来访问证书库。PKI 系统必须保证证书库的完整性和真实性，

防止伪造、篡改证书。

(4) 密钥备份及恢复系统

为了防止用户丢失解密私钥导致密文数据无法还原成明文，从而造成数据丢失，PKI 系统应该提供密钥的备份和恢复的机制。密钥的备份和恢复应该由可信机构来完成，如 CA。

(5) 证书撤销处理系统

由于私钥泄漏、密钥更换、用户身份变化等等原因，PKI 必须提供证书的作废机制。PKI 中注销证书是通过维护一个撤销证书列表(CRL，Certificate Revocation List)来实现的。PKI 系统将撤销证书列入 CRL，当用户需要验证证书时，由 CA 负责检查证书是否在 CRL 之中。比较常见的证书撤销处理系统采用基于 Web 的 CRL 服务来实现，将检查 CRL 的统一资源定位符(URL, Uniform Resource Locator)内嵌在用户的证书中，可以提供安全途径(如 SSL)访问 URL。

(6) PKI 应用接口系统

PKI 应用接口系统可以为用户提供良好的应用接口，使得各种各样的应用能够以安全、一致、可信的方式与 PKI 交互，确保所建立起来的网络环境是可信的，同时降低管理维护成本。为了向应用系统屏蔽密钥管理的细节，PKI 应用接口系统应该是跨平台的。

8.1.4 PKI 的工作过程

PKI 技术通过第三方的可信任机构——认证中心(CA，Certificate Authority)，把用户的公钥和用户的其他标识信息(如名称、电子邮件、身份证号等)捆绑在一起形成数字证书，在互联网上验证用户的身份。建立 PKI 基础设施的目的是管理密钥和证书。通过 PKI 对密钥和证书的管理，一个组织可以建立并维护可信赖的网络环境。作为网络安全的一种基础设施，PKI 必须具备良好的性能，同时还应具备以下特性：

• 透明性：PKI 必须尽可能地向上层应用屏蔽密码实现服务的实现细节，向用户屏蔽复杂的安全解决方案，使密码服务对用户而言简单易用，并且便于单位、

企业完全控制其信息资源。

• 可扩展性：满足系统不断发展的需要，证书库和 CRL 有良好的可扩展性。

• 互操作性：不同企业、不同单位的 PKI 实现可能是不同的，必须支持多环境、多操作系统的 PKI 的互操作性。

• 跨平台：PKI 应该支持目前广泛使用的各种操作系统平台，如 Windows、UNIX、MAC、OS/400 等。

• 灵活性：支持多种网络应用，能为文件传送、文件存储、电子邮件、电子表单、WEB 应用等不同应用提供安全服务。

PKI 的工作过程可以从其提供的几项基本服务具体分析：

(1) 鉴别(Authentication)

即确认实体就是它自己所声明的。具体过程如下：

① 用户在向验证者声明自己的身份时，可提供自己的公钥数字证书；

② 验证者事先获取证书颁发权威机构的公钥证书，得到权威机构的公钥；

③ 验证者根据数字证书结构判断用户数字证书的签名算法，即摘要算法和加密算法；

④ 验证者根据数据证书结构将证书信息和权威机构的签名分离；

⑤ 验证者使用摘要算法对证书信息进行处理，得到信息的摘要；

⑥ 验证者使用权威机构的公钥对签名部分进行解密；

⑦ 验证者比较解密内容和摘要内容是否一致，如果内容相同说明鉴别成功。

在上述过程中，验证者也可以直接从权威机构提供的 LDAP 服务器上获得用户的数字证书，根据具体应用需求，还可能需要访问 CRL 检查证书是否已经撤销。

(2) 数据完整性(Data Integrity)

即确认信息在传递或存储过程中没有被篡改、重组或延迟。具体过程如下：

① 发送方使用摘要算法对需要发送的消息进行处理，得到消息摘要；

② 发送方使用自己的私钥对摘要进行加密，得到数字签名；

③ 发送方将消息和数字签名一起发送给接收方；

④ 接受方获得发送方的公钥证书，并验证证书的有效性（参考鉴别过程）；

⑤ 接收方从公钥证书中获取发送方的公钥，使用公钥对受到的数字签名进

行解密；

⑥ 接收方使用相同的摘要算法对受到的消息进行摘要运算；

⑦ 比较解密结果和摘要运算的结果，如果一致说明消息在传送过程中没有被篡改。

如果需要防止重放攻击，发送方可在消息中增加时间戳信息，接收方可根据时间戳判断消息是否已经过期。

(3) 数据保密性(Data Confidentiality)

即确保数据的秘密，除了接收者之外，无人能够读出数据信息。具体过程如下：

① 发送方选择一个随机的密钥，使用对称密码算法对要发送的消息进行加密；

② 发送方获得接收方的公钥证书，并验证证书的有效性（参考鉴别过程）；

③ 发送方从公钥证书中获取接收方的公钥，使用公钥对随机密钥进行加密，形成数字信封；将加密的消息连同数字信封一起发送给接收方；

④ 接收方使用自己的私钥对数字信封进行解密，得到随机密钥；

⑤ 接收方使用随机密钥和相同的对称密码算法对消息密文进行解密，从而得到消息明文。

以上过程需要注意的是，对消息进行加密的时候，仍然使用的对称密码算法，而不是直接使用公钥密码算法对消息进行加密。这是因为公钥密码的运算效率远低于对称密码算法，这里使用公钥算法的主要目的是进行对称密钥的交换。

(4) 不可抵赖性(Non-Repudiation)

即发送者不能否认已发送的信息，可使用数字签名获得不可抵赖性，具体过程可参考数据完整性实现过程。如果使用公钥可以解密某消息，则可确定该消息是使用对应的私钥加密的，而私钥由用户个人负责保管，从而可证明消息是私钥所有者发送的。

8.2 权限管理基础设施

8.2.1 构建 PMI 的必要性

PKI 通过方便灵活的密钥和证书管理，为在线身份认证提供了有效的手段，成为当前电子商务、电子政务、企业办公等网络应用中不可缺少的安全支撑系统。然而，随着网络应用的扩展和深入，仅仅能通过第三方权威机构确定“他是谁”，已经不能满足日益复杂的安全应用需求。在很多情况下，还需要有一个权威机构确定“他能做什么”。

解决上述问题的思路之一是利用身份证书中的扩展项保存用户的授权记录，即由 CA 中心完成权限的集中管理。应用系统通过查询用户的身份证书即可得到用户的权限信息。该方案的优点在于可以直接利用已经建立的 PKI 平台进行统一授权管理，实施成本低，接口简单，服务方式一致。但是，将用户的身份信息和授权信息捆绑在一起管理存在以下几个方面的问题：

首先，身份和属性的有效时间有很大差异。身份往往相对稳定，变化较少；而属性如职务、职位、部门等则变化较快。因此，属性证书的生命周期往往远低于用于标识身份的公钥证书。举例来说，公钥证书类似于日常生活中护照，而属性证书类似于签证。护照代表了一个人的身份，有效期往往很长；而签证的有效期则几个月、几年不等。

其次，公钥证书和属性证书的管理颁发部门有可能不同。仍以护照和签证为例，颁发护照的是一个国家，而颁发签证则是另一个国家；护照通常只有一个，而签证数量却决定于要访问的国家的多少。与此相似，公钥证书由身份管理系统进行控制，而属性证书的管理则与应用紧密相关：什么样的人享有什么样的权力，随应用的不同而不同。一个系统中，每个用户只有一张合法的公钥证书，而属性证书则灵活得多。多个应用可使用同一属性证书，但也可为同一应用的不同操作颁发不同的属性证书。

由此可见，身份和授权管理之间的差异决定了对认证和授权服务应该区别对待。认证和授权的分离不仅有利于系统的开发和重用，同时也有利于进行安全方面更有效的管理。只有那些身份和属性生命期相同、而且 CA 同时兼任属性管理

功能的情况下，可以使用公钥证书来承载属性，即在公钥证书的扩展域中添加属性字段。大部分情况下应使用公钥证书+属性证书的方式实现属性的管理。在这种背景下，国际电信联盟 ITU 和因特网网络工程技术小组 IETF 进行了 PKI 的扩展，提出权限管理基础设施(PMI，Privilege Management Infrastructures)，允许该基础设施支持和处理授权。

PMI 适用于大型企业的多应用系统集中授权。大型企业通常拥有多种不同的业务信息系统，这些系统往往由不同的软件开发商提供，每个系统都有独立的用户和权限管理功能，这些重复的功能不但增加了系统权限管理的负担，也是一种软件资源的浪费。更重要的是，彼此分离的权限管理系统可能存在权限管理的不一致，更容易产生权限分配的失误。在这种环境下，将企业各个不同系统的访问控制功能集中统一实现，不失为一种有效的权限管理策略，如果企业已经建立了统一的身份鉴别平台(如 PKI)，则非常适合采用 PMI 完成上述功能。

8.2.2 属性证书

和 PKI 一样，PMI 也应支持不同异构环境下的各类应用，因此要求有一个统一的记录用户属性的格式。1997 年，ITU-T 在 X.509 V3 中提出了标准的属性证书格式，称为第一版(V1)，在 2000 年又提出 X.509 V4，其中给出了属性证书格式的第二版(V2)。X.509 V3 定义了基本的属性证书语法，而 X.509 V4 在此基础上增加了如下内容：

(1) 扩展了属性证书语法；

(2) 定义了 PMI 模型；

(3) 定义了代理路径处理；

(4) 定义了 PMI 扩展标准集；

(5) 增加了目录模式对象定义。

PKIX 规定属性证书采用 V2 格式，它可以向后兼容 V1 格式的属性证书，其格式如图 8-2 所示：

版本号(version)
持有者(holder)

颁发者(issuer)
签名算法标识符(signatureAlgorithm)
序列号(serialNumber)
有效期(attrCertValidityPeriod)
属性(attributes)
颁发者唯一标识(可选)(issuerUniqueID)
扩展(可选)(extensions)
签名算法标识符(signatureAlgorithm)
签名值(signatureValue)

图 8-2 属性证书结构

属性证书中各字段含义如下：

(1) 版本号：X.509 属性证书格式的版本号，当前版本是第二版。0 表示版本 1，1 表示版本 2。

(2) 持有者：用于标识证书所有者的身份。有三种标识方法：① 持有者的公钥证书的序列号和颁发者名称，可以据此建立到公钥证书的链接，从而进行身份认证；② 指定一个实体名称，但是这样不能进行证书所有者的认证；③ 使用一个客体的散列值，如公钥的散列值和公钥证书的散列值，通过把这一字段同用户提供的相应散列值作比较来进行认证。

(3) 颁发者：用于标识签发属性证书的授权管理机构。有三种标识方法：① 指定一个通用名称，② 颁发者的公钥证书的序列号和上级颁发者名称，③ 使用一个客体的散列值。

(4) 签名算法标识符：颁发者对证书进行数字签名时使用的密码算法(包括摘要算法和签名算法)的标识符。

(5) 序列号：由颁发者分配给属性证书的唯一整数标识符，它在颁发者范围内唯一。

(6) 有效期：定义证书有效的起始时间和终止时间。

(7) 属性：给出证书持有者的一些属性或特权信息。属性域包含了一个属性序列，每个属性含有一个或多个属性值。

PKIX 定义了下列属性项：

① 服务认证信息：提供一些认证信息，用于服务器鉴别属性证书持有者的身份。当包含敏感信息时，例如口令，该属性将被加密。该属性不能与“访问身

份”属性同时使用。

② 访问身份：提供属性证书持有者的认证信息，且该属性证书被用于特定目的。该属性可以被属性证书验证者用来授权属性证书。该属性不能与“服务认证信息”属性同时使用。

③ 收费身份：用于收费的属性证书持有者。

④ 团体：关于属性证书持有者团体成员之间的关系。

⑤ 角色：包含分配给属性证书持有者的角色信息，包括定义角色层次的机构和分配给持有者的角色。

⑥ 更新：包含属性证书持有者的更新信息。

(8) 颁发者唯一标识(可选)：当颁发者名称出现重用时，使用该标识符来唯一标识颁发机构。

(9) 扩展(可选)：由一个或者多个扩展项组成，用来标识证书的额外信息。

① 时间限制：用于限制特权可以应用的时间段。例如，激活一个应用程序的特权只能在周一到周五的上午 9 点到下午 5 点之间使用。这一扩展也可以用于上级在度假期间对下属进行临时委托授权。该扩展必须标记为关键扩展。

② 目标信息：用于指定可以使用属性字段里标识的权限的应用程序集合。该扩展必须为关键扩展。

③ 用户通知：允许 AA 在声明权限时对属性证书的持有者进行用户通知。该扩展可用于解释一些条件限制或给出一些关于权限使用的合法性提示。该扩展既可以标记为关键扩展也可以标记为非关键扩展。

④ 可接受权限策略：用于指定在验证权限的过程中必须使用的策略集合。该扩展必须为关键扩展。

⑤ ACRL 分布点：用于指定属性证书撤销信息的存放位置。

⑥ 无撤销信息扩展：明确指出该属性证书不存在撤销信息。这可能是因为该证书的有效期十分短。该扩展总是标记为非关键扩展。

⑦ SOA 标识扩展：指出该证书的主体是一个负责权限管理的 SOA。这并不是标识 SOA 的唯一机制。该扩展为非必须扩展。

⑧ 属性描述符扩展：提供了用于定义权限属性的机制和在授权时应用于权限的规则。该扩展仅用于属性描述符证书，这种证书用于定义某种特定的特权属

性，是自颁发证书，即颁发者和持有者字段的值相同。该扩展总是标记为非关键扩展。

⑨ 角色说明证书标识符：AA 使用该扩展指向一个角色说明证书。该扩展可用于角色分配证书，当权限验证者接收到一个角色分配证书时，该扩展用于定位对应的角色说明证书。该扩展总是标记为非关键扩展。

⑩ 基本属性限制：标识该属性证书的所有者是否可以颁发属性证书，即指明该实体是否是授权机构。该扩展在 AA 证书中必须存在，并建议标记为关键扩展。

⑪ 委托名称限制：标识授权路径中剩余的属性证书必须遵循的命名空间限制。该扩展既可以标记为关键扩展也可以标记为非关键扩展。

⑫ 可接受证书策略：标识对委托授权路径上的属性证书颁发者 AA 必须具有相应身份证书的策略。该扩展只有在颁发给其他 AA 的证书中才有效。该扩展必须为关键扩展。

⑬ 授权机构属性标识符：用于指向颁发者 AA 的属性证书。特权验证者可以使用该扩展来检查 AA 是否具有足够的特权来对证书所有者进行授权。该扩展总是标记为非关键扩展。

(10) 签名算法标识符：颁发者对证书进行数字签名时使用的密码算法(包括摘要算法和签名算法)的标识符。

(11) 签名值：颁发者对属性证书进行签名的结果。

8.2.3 PMI 的功能和组成

PMI是属性证书、属性机构、属性证书库等部件的集合体，用来实现属性证书的产生、管理、存储、颁发和撤销等功能。PMI在体系上可以分为三级，分别是信任源点(SOA，Source of Authority)、属性权威机构(AA，Attribute Authority)和AA代理点。在实际应用中，这种分级体系可以根据需要进行灵活配置，可以是三级、二级或一级。

(1) 信任源点 SOA

信任源点也称 SOA 中心，是整个授权管理体系的中心业务节点，也是整个授权管理基础设施 PMI 的最终信任源和最高管理机构。SOA 中心的职责主要包

括：授权管理策略的管理、应用授权受理、AA 中心的设立审核及管理和授权管理体系业务的规范化等。

(2) 属性权威机构 AA

属性权威机构也称授权服务中心或 AA 中心，是授权管理基础设施 PMI 的核心服务节点，是对应于具体应用系统的授权管理分系统，由具有设立 AA 中心业务需求的各应用单位负责建设，并与 SOA 中心通过业务协议达成相互的信任关系。AA 中心的职责主要包括：应用授权受理、属性证书的发放和管理，以及 AA 代理点的设立审核和管理等。AA 中心需要为其所发放的所有属性证书维持一个历史记录和更新记录。

出于和 PKI 中设立 RA 中心相同的目的，通常可设立属性注册授权机构(ARA，Attribute Registered Authority)分担 AA 中心的有关工作，具体负责接收、审核和处理用户关于属性证书的注册申请工作。

(3) 授权服务代理点

AA代理点是授权管理基础设施PMI的用户代理节点，也称为资源管理中心，是与具体应用用户的接口，是对应 AA 中心的附属机构，接受 AA 中心的直接管理，由各 AA 中心负责建设，报经主管的 SOA 中心同意，并签发相应的证书。AA 代理点的设立和数目由各 AA 中心根据自身的业务发展需求而定。AA 代理点的职责主要包括应用授权服务代理和应用授权审核代理等，负责对具体的用户应用资源进行授权审核，并将属性证书的操作请求提交到授权服务中心进行处理。

(4) 访问控制执行者

访问控制执行者是指用户应用系统中具体对授权验证服务的调用模块，因此，实际上并不属于授权管理基础设施的部分，但却是授权管理体系的重要组成部分。访问控制执行者的主要职责是：将最终用户针对特定的操作授权所提交的授权信息(属性证书)连同对应的身份验证信息(公钥证书)一起提交到授权服务代理点，并根据授权服务中心返回的授权结果，进行具体的应用授权处理。

8.2.4 属性证书的管理

和 PKI 中的身份证书一样，PMI 的核心功能之一是确保属性证书在一个受

控的环境中被合理的使用，即管理和维护属性证书的申请、生成、使用、验证和撤销。

(1) 属性证书的颁发

用户应向 ARA 申请属性证书，ARA 在审核资料的合法性后发送属性证书请求给 SOA 服务器。SOA 服务器负责签发属性证书，并将属性证书发布到 LDAP 目录服务器，上述过程也支持在线的属性证书申请。处理流程如下：

① 用户访问 SOA 的 Web 应用服务器，并提交其身份证书 PKC。

② Web 应用服务器将 PKC 传递给 SOA 服务器。

③ SOA 服务器调用 PKI/CA 系统中的证书验证接口，验证证书的有效性并将响应消息回传给 Web 应用服务器。

④ Web 服务器会根据响应消息的类型判断该证书是否通过验证。若通过验证，则提示用户是否进行属性证书的申请，若用户选择申请，则链接到角色信息页面，否则结束。

⑤ 用户在角色信息页面填写与自己相对应的角色信息后提交 Web 服务器。Web 应用服务器将用户提交的信息发送给 SOA 服务器。

⑥ SOA 服务器提取用户的身份信息并进行数据库查询。数据库中存放了系统中所有用户的身份与角色的对应信息，通过查询验证用户申请的角色信息与他所应具有的角色信息是否一致。若二者不一致，则返回出错信息给 Web 服务器，Web 服务器返回用户出错页面；若二者一致，则将相应的用户信息填入相应字段中，生成属性证书，存放到 LDAP 目录服务器上，同时生成响应信息回传给 Web 服务器，Web 服务器返回页面提示用户生成证书成功。

(2) 属性证书验证

由于 PMI 是基于 PKI 构建的，因此属性证书的验证也必须在身份证书验证的基础上完成，即先进行身份的检查，再进行权限的检查。通常，一个有效的属性证书必须满足下面的条件：

① 如果属性证书的拥有者使用公钥证书向属性验证者认证自己的身份，那么该公钥证书的完整的证书路径必须由验证者进行验证，路径中每个证书从最终实体到根证书都必须是有效的；

② 属性证书的签名必须是可信任的，即属性证书颁发者的公钥证书的证书

路径也必须通过验证，验证过程同①；

③ 属性证书被验证的时间必须在有效期内。值得注意的是，在某些应用中，被验证的时间可能并不等于当前时间；

④ 属性证书的目标信息扩展内容是一系列的服务器或者服务，表示此属性证书只能在这些服务器或者服务中有效。如果被验证的属性证书有此扩展，而证书验证者没有出现在该扩展的内容中，则该属性证书将验证失败；

在使用属性证书的应用系统中，系统必须能够验证属性证书中权限的真实性，防止虚假的权限信息和冒名顶替。但属性证书中的信息不足以提供完整的验证信息，因此，属性证书不能单独使用，而且必须支持某种形式的身份认证过程。基本的身份和权限验证过程应该如下：

① 验证用户的身份证书

用户提交身份证书，应用系统验证证书的有效性，包括查看证书是否经过 CA 的正确签名，并确定签发证书 CA 的合法性、证书是否已被撤销等。用户身份证书的具体验证过程参见 PKI 详细设计报告第 4 章中的证书验证、CRL 验证以及证书链的验证。

② 获取属性证书

属性证书的获取有两种模式：推模式和拉模式。

所谓推模式，是当用户要求访问资源时，由用户自己直接提供其属性证书，即用户将自己的属性证书“推”给权限验证者。这意味着在属性证书所有者和验证者之间不需要建立新的连接，而且对于服务器来说，这种方式不会带来查找证书的负担，从而提高了性能。

所谓拉模式，是由属性权威授权机构发布属性证书到目录服务系统，当权限验证者需要使用属性证书的时候，主动从属性证书发放者(属性权威 AA)或存储证书的目录服务系统“拉”回属性证书。这种“拉”模式的一个主要优点在于实现这种模式不需要对客户端以及客户－服务器协议作任何改动。

其中“推模式”适用于跨平台应用，“拉模式”适用于单平台应用。“推模式”机制实现起来相对简单，不需要特权验证者对目录中的相关属性证书进行查找，验证效率较高，可提高系统性能。但它需要改变已有客户和服务器间的通信协议，并且系统的灵活性较“拉模式”机制要差一些，处理一些例外情况比较复杂，如

属性证书的撤销问题等。而“拉模式”机制参照 X.509 的标准惯例，颁发的证书以 X.500 目录形式管理。实施中可沿用已有的通信协议，不需对现有客户端做大的修改，同时系统的灵活性较好，可由特权验证者加入自定义的一些检查步骤。

③ 验证属性证书

根据授权层次结构设计，用户的属性证书直接由 SOA 颁发，因此，验证属性证书路径相对比较简单，只需要直接验证 SOA 的证书是否有效。

应用程序获得了属性证书后，首先需要获取 SOA 的公钥并验证其有效性；然后用 SOA 的公钥对属性证书的签名进行验证；检查属性证书的有效期；对属性证书持有者进行身份认证以及检查 ACRL 确保属性证书未被撤销。

④ 最后检查属性证书中的内容，根据访问控制策略，获得用户角色所对应的权限，来确定是否允许此用户访问其所需的资源及服务。

(3) 属性证书的撤销

属性证书的撤销可以采用和 PKI 中身份证书撤销类似的方案，即维护一个属性证书撤销列表(ACRL，Attribute Certificate Revocation List)，将作废的属性证书存放在该列表中。但是，由于属性证书只用于权限的判断，并不参与数据加脱密处理，不存在历史数据需要还原的问题；而且大多数情况下，属性证书的有效期是比较短的。所以，通常可以舍弃 ACRL 列表，验证者每次采用拉模式获得被验证用户的最新属性证书。

PMI 支持上述两种不同的撤销方案，即无 ACRL 列表的方案和使用 ACRL 列表的方案。在属性证书结构字段中，有专用的扩展信息项指示 PMI 所使用的方案。采用第一种方案时，验证者每次从公共的属性证书目录中查找最新的属性证书；采用第二种方案时，属性证书中需要利用 ACRL 分布点扩展指出撤销信息的来源，属性证书的撤销和 PKI 中公钥证书的撤销过程类似。开发者可以根据不同的应用环境需求选择合适的撤销方案。

属性证书的撤销请求可由用户或安全管理员向 SOA 中心提出。具体处理流程如下：

① 用户通过访问 Web 站点的方式访问 SOA 服务器，发送他的 PKC 证书和用相应私钥签名的信息(如对用户名称的签名)；

② SOA 服务器首先调用 PKI 系统中证书验证接口，验证 PKC 证书的合法

性，验证通过则说明用户为合法用户，如果是非法用户则退出，否则继续；

③ SOA 服务器根据 PKC 的用户名字段(Subject)从 LDAP 目录服务器中取出该用户的所有属性证书，把用户的属性证书列表返回给用户；

⑤ 用户选择要撤销的属性证书，把撤销请求发给 SOA 服务器，SOA 撤销该证书，签发 ACRL。

对于无 ACRL 的方案，当证书过期时只需由 SOA 管理员直接从 LDAP 服务器上将该属性证书删除即可。验证的时候，如果从 LDAP 上找不到该证书，则表示该证书已被撤销或已过期。

对大多数广泛使用的属性证书来说，由于它们的有效期很短，当用户还没有来得及撤销时，它们已经过期了，因此没有必要进行专门的撤销操作。系统不用发布和维护属性证书撤销列表，证书的验证过程也更简单。因此，无 ACRL 的撤销方案更容易被实际应用系统采纳。

8.2.5 基于 PMI 的授权与访问控制模型

PMI 主要围绕特权的分配使用和验证来进行。属性证书框架采用四个模型来描述敏感资源上的权限是如何分配、流转、管理和验证的。通过这四个模型，我们可以明确 PMI 中的主要相关实体，主要操作进程，以及交互的内容。

(1) 通用模型

通用模型也称为基本模型，如图 8-3 所示。模型中包含三个实体：授权机构（SOA 或 AA）、特权持有者（Privilege Holder）和特权验证者（Privilege Verifier）。授权机构向特权持有者授权，特权持有者向资源提出访问请求并声称具有权限，由特权验证者进行验证。特权验证者总是信任授权机构，从而建立信任关系。

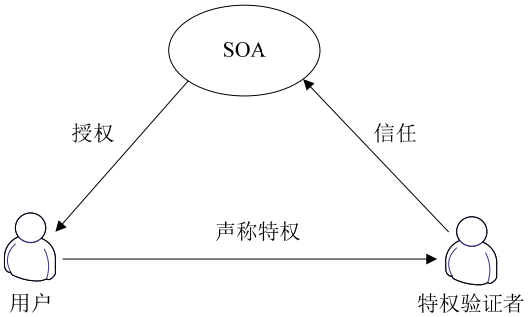


图 8-3 PMI 通用模型

PMI 基本模型描述了在授权服务体系中主要三方之间的逻辑关系，以及两个主要过程：特权分配和验证。这是 PMI 框架的核心。PMI 基本模型的体系结构类似于单级 CA 的体系结构，SOA 的作用就可以看作是 CA。对特权的分配是由 SOA 直接进行的，SOA（AA）通过为特权持有者颁发属性证书来进行授权。由于 SOA 同时要完成很多宏观控制功能，如制订访问策略，维护撤消列表，进行日志和审计工作等，特别是当用户数目增大时，就会在 SOA 处形成性能瓶颈。SOA 也会显得庞大而臃肿。

如果实际应用中用户数量非常庞大，就需要对基本模型进行改进，以实现真正可行的 PMI 体系。一个明确的思路就是对授权管理功能进行分流，减少 SOA 的直接特权管理任务，使得 SOA 可以进行自身的宏观管理功能。

(2) 控制模型

控制模型(如图 8-4)阐述了如何控制对敏感对象方法的访问。模型中有五个对象：特权声称者、特权验证者、对象方法(敏感)、特权策略和环境变量。特权声明者具有特权；对象方法具有敏感性。这里描述的技术使特权验证者在一致特权策略下控制特权声明者对对象方法的访问。特权和敏感性都是多值参数。特权声明者可能是一个被公钥证书所鉴别的实体，或一个被磁盘镜象摘要所鉴别的可执行对象等。

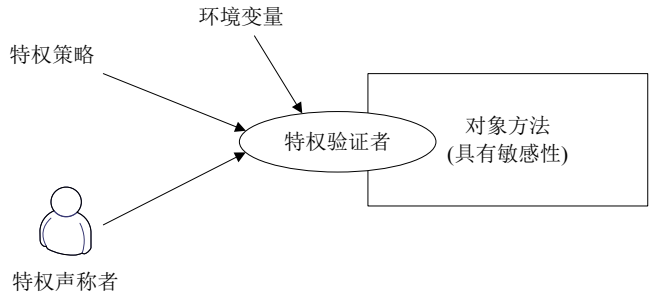


图 8-4 PMI 控制模型

(3) 委托模型

特权委托提出了授权机构的层次级联。特权经由若干 AA 从上至下流转，最后到达特权持有者。这样，SOA 和高层 AA 就只负责制订访问策略，维护有限

的证书分配和管理功能，从而实现了系统规模的扩张和整体性能的优化。特权分配类似于一个树形结构，如图 8-5 所示。图 8-6 给出了委托方式下进行权限管理和验证的模式。委托模型中有四种角色：SOA、AA、特权验证者和特权声称者。SOA 将特权委托给 AA，通过发布属性证书来确定 AA 所拥有的权限或权限子集，同时赋予 AA 进行特权委托的权力。SOA 可以通过限制委托路径深度、限制名字空间等方法来控制后继委托。AA 所委托的特权不能超过他本身所拥有的特权。

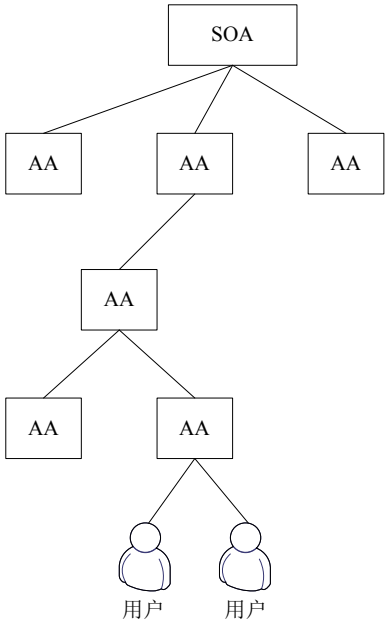


图 8-5 委托树模型

特权验证者信任 SOA 对资源的访问控制特权。当一个特权持有者提出请求时，若其持有证书并非有 SOA 颁发，特权验证者将定位通往 SOA 的委托路径进行验证。并对该路径上每一 AA 节点进行判断是否具有该权限。

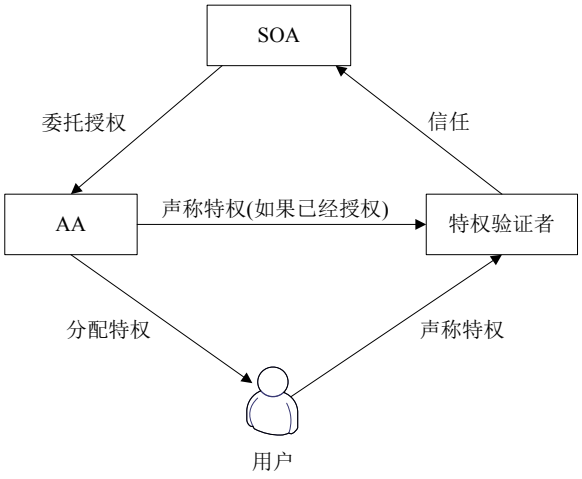


图 8-6 委托方式下的管理和验证

特权委托路径与公钥证书的有效路径是不同的。委托路径将既包括属性证书又包括公钥证书，如果是通过公钥证书获得特权则只能通过发布公钥证书来委托特权。如果是通过属性证书获得特权则只能通过发布属性证书来委托特权。只有AA 才能委托，最终用户不具备委托权限。

在委托模型中，特权的委托路径是一个关键问题。不光在分配过程需要由策略对此进行约束，在验证过程也要对委托路径进行有效性判断。

(4) 角色模型

PMI 角色模型如图 8-7 所示，角色模型提供一种间接分配特权给个体的方式。在角色模型中，SOA 不再将特权直接分配给特权持有者，而是建立若干角色并将权限授予角色。该模型使用两个证书完成角色的定义和分配。特权持有者申请特权时，SOA 通过角色分配证书(Role Assignment Certificates)，为个体赋予角色身份，从而使个体获得角色所具有的特权集合。SOA 颁发角色分配证书给个体，通过证书中的角色属性使他们能够扮演一个或多个角色。而角色的特权是通过角色规范证书（Role Specification Certificates）来赋予的。角色分配证书可以是公钥证书，也可以是属性证书，但角色规范证书只能是属性证书。

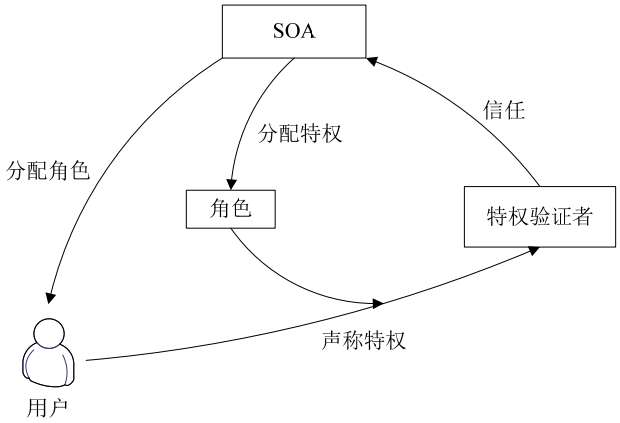


图 8-7 PMI 角色模型

在角色模型中，SOA 持有角色规范证书，特权持有者只持有角色分配证书，类似于一个指向特权的指针。在进行访问请求时，特权验证者根据特权持有者提交的角色分配证书，在本地的角色规范证书库中查找对应者；如果没有，则根据分配证书中的信息到 SOA 处查找。

基于角色授权的 PMI 模型通过引入角色这个中间层次能有效地简化授权管

理，进一步降低系统复杂度和管理成本，提高了系统的可理解性和易修改性，提供授权管理的灵活性和可靠性。将用户按角色进行分类授权后，系统管理者只需要对角色的特权进行修改，就可以控制具有该身份的所有用户的特权，另一方面，基于角色的管理贴近实际应用，更加人性化，易于理解。一个角色的特权的更新并不影响终端实体，实现了对授权管理较大的灵活性。但是这种方式需要颁发和处理两种不同的证书，这给证书管理增加了很大的负担，和委托授权的方式一样，由于验证的时候需要角色规范证书，所以也带来了特权验证者的复杂性。采用这种方式时，可以考虑和 RBAC 机制进行结合，从而减少授权管理的复杂性，降低管理开销。

8.2.6 PMI 的产品和应用

作为国内外信息安全领域研究的热点之一，PMI 得到了许多研究机构和公司的关注，如 IBM、True Trust、Baltimore、Entrust、Verisign、SSE、Salford 大学等，市场上也出现了不少 PMI 产品。在这些产品中，一类是独立的 PMI，如 PERMIS，另一类是结合了其他功能模块的安全产品解决方案的一部分，如 Baltimore 等公司的产品。下面简要介绍几个比较具有代表性的产品。

(1) PERMIS PMI

PERMIS PMI 是英国 True Trust 公司推出的 PMI 产品。该产品的前身是 Dr.Chadwick 负责的一个欧共体项目，即从 2000 年 12 月到 2002 年 6 月完成的 PERMIS(PrivilEge and Role Management Infrastructure Standards Validation)。PERMIS 是基于 X.509 标准的 PMI 实现。并且在三个不同城市 Salford，Bologna 和 Barcelona 的不同应用中实施。该项目据此提出了一项 RFC(Request for Comment)来标准化电子商务应用中的特权需要以及描述 PERMIS 的 API。

(2) Akenti

Akenti 是一个由美国 Lawrence Berkeley National 实验室开发的授权管理基础设施。它也是依据 RFC2704 所提出的五个组件的可信管理基础设施。

(3) SelectAccess

Baltimore 公司的 SelectAccess 是个提供特权管理基础设施 PMI 服务的世界领先的授权管理解决方案，它允许管理并执行用户的特权，对电子商务和企业资

源交易进行授权。在一个企业外联网环境中，SelectAccess 对基于 Web 的资源提供基于角色的授权，使企业为客户、供应商和伙伴提供安全的丰富的用户体验。SelectAccess 用二维表表示所有用户和资源。SelectAccess 支持多种鉴别方法来维持一个安全可信的环境。支持口令、数字证书（软件形式的和智能卡形式的）和安全 ID 令牌（SecureID tokens）；管理基于角色和组；动态角色特征确保用户特权改变与商务过程的变化同步；可支持多级委托；鉴别措施可通过 API 支持。

SelectAccess 贯彻了 XML 技术，XML 为数据的传输和整合到现存及未来的应用提供了充分的灵活性，不管是基于 WEB 还是非 WEB 的。支持 Windows NT/2000,Linux,Solaris，HP-UX 等操作系统。访问控制通过插件强迫执行，可以用于 Microsoft IIS，iPlanet，Apache，BEA WebLogic，IBM WebSphere，Silver Stream，Oracle 9i 以及 Plumtree 等。

(4) IBM Tivoli secureWay Authorization

The Open Group aznAPI 由 DASCOM(现已被 IBM 收购)提出。其目的是将授权与信任分开、授权与具体实现机制（如 ACL，Entitlements、Rules、Classification，等等）分开。

(5) Entrust GetAccess

Entrust GetAccess 是一个安全的、灵活性好的、高性能、可升级 Web 访问控制解决方案。它采用集中式的管理方式，提供用户身份认证与授权，它通过一个单一的入口和访问点来管理多个应用，它对用户被授权访问的应用和内容可实现单点登录（SSO）。

Entrust GetAccess 支持基于角色和规则两种不同机制的细粒度的访问控制机制。GetAccess 7.0 给出了一个详尽的策略引擎，该引擎对在线资源允许管理员定义和执行细粒度的访问控制策略。

强大的基于规则的访问控制为访问控制策略定义特定的属性能够简单规则的创建。简单规则的例子包括强制一个特定的认证方法，一个物理位置或者一个特定的时间段。基于规则的访问控制策略建立在 XACML 标准基础之上，XACML 标准提供对策略和规则与建立在相同标准上的其他应用的更好的集成。

(6) 北京耐丁网络技术有限公司用户授权和访问控制系统

耐丁公司的用户授权和访问控制体系的建设就是设计统一的授权策略，建立

统一的用户管理中心，提供统一的应用系统访问控制基础平台和实现机制，解决企业网现有的多种不同类型的用户对多个不同业务应用系统的授权和访问控制问题，防止用户越权访问或修改数据，确保对信息的访问限制在授权范围内。

(7) 吉大正元权限管理和授权服务基础平台

PMI 权限管理和授权服务基础平台是由吉大正元开发的通用权限管理平台。主要应用在权限管理，访问控制领域。为进行资源管理的二次开发人员提供了一个方便，安全，高效的决策平台。PMI 权限管理和授权服务基础平台主要用于 web 资源的访问控制，磁盘资源的访问控制，数据库资源的访问控制，网络资源的访问控制和硬件资源的访问控制。

(9) 维豪集团安全平台

维豪集团将基于公钥基础设施 PKI 和授权管理基础设施 PMI 的智能化信任与授权技术与 J2EE 技术结合，搭建了应用于不同领域的各种安全平台。

以上产品，在各国不同行业领域得到了广泛的使用，其中比较具有代表性的是 PERMIS 项目的实施。该项目分别在西班牙的巴赛罗那 Barcelona、意大利的博罗尼亚 Bologna 和英国索尔福德 Salford 三个城市的不同应用中得到了应用，较全面地测试了 PMI 系统的通用性和安全性。

在巴赛罗那，城市建筑师们在进行市政建设时，需要向政府邮寄纸质的审请文件和建筑计划书，在指定资料室查阅本市的街道地图。利用 PERMIS 系统，城市规划办公室对各类资源的权限和相关人员的角色进行了集中管理，允许城市建筑师下载本市的街道地图，同时可以按照自己的意图计划来更新这些地图，也允许他们向城市规划办公室服务器上载新的建筑计划，以及向城市规划办公室申请建筑许可证。这大大提高了原有系统的效率。

博罗尼亚是意大利重要的旅游和商业中心，但是停车是限制的。许多的停车罚单经常发给了被雇佣的汽车，但是在汽车租赁公司接收到停车罚单的时候，租借者可能已经离开了这个地方。PERMIS 项目就是计划给汽车租赁公司对城市停车罚单数据库提供一个在线的访问，那么公司将立刻检查到是否有任何的停车罚单被颁发给返回的汽车。每个公司将能向城市发送司机的详细情况，因此罚金将传送给实际用户。立法要求一个汽车租赁公司能够仅仅访问颁发给它自己的汽车的罚单，所以授权需要在记录层。

索尔福德实现了一个电子招投标应用。招投标的过程开始于城市给出在它自己的网站上给出提议请求文档，并允许任何人去下载这些文档。然而，在招投标中有一些严格的要求，只允许具有索尔福德授权的公司才能提交投标方案。在其他情况下，可能需要一个公司具有 ISO9000 或者其他的认证才可以提交投标方案。一旦投标方案被提交，它必须保持匿名直到选出优胜者。城市的招标官员在提议请求文档关闭日期前不可以访问电子投标方案，而投标者在提议请求文档关闭日期之后不允许提交投标方案。PERMIS 项目利用 PMI 平台，为授权的投标公司和招标官员颁发属性证书，并实现了控制模型对投标方案资料进行权限的管理和访问的控制。

习题八

1. PKI 系统能提供哪些安全服务?

2. PKI 系统和 PMI 系统两者之间有何关系?

3. 使用身份证书保存用户的权限信息存在什么问题?

4. 如何验证权限证书的有效性?

5. 考虑在第五章提到的医院管理信息系统中使用 PMI 系统, 应采用何种模型实现权限的管理?

参考文献

[1] 1 （美）Matt Bishop. 计算机安全学——安全的艺术与科学. 北京:电子工业出版社, 2005

[2] 2 洪帆, 崔国华, 付小青. 信息安全概论. 武汉: 华中科技大学出版社, 2005

[3] 3 段云所, 魏仕民, 唐礼勇, 陈钟. 信息安全概论. 北京: 高等教育出版社, 2003

[4] 4 李中献, 詹榜华, 杨义先. 认证理论与技术的发展. 电子学报, 1999, 27(1): 98~102

[5] 5孙冬梅, 裘正定. 生物特征识别技术综述. 电子学报, 2001, 29(12A): 1744~1748

[6] 6 Frederick Butler,Iliano Cervesato,Aaron D Jaggard et al. A formal analysis of some properties of kerberos 5 using MSR.IEEE COMPUTER SOCIETY,2002;(11)

[7] 7 Ian Downnard.Public-key cryptography extensions into Kerberos.2002 IEEE POTENTIAL,2003

[8] 8 K Raeburn.Encryption and Checksum Specifications for Kerberos 5.Internet RFC 3961,February, 2005

[9] 9 K Raeburn.Advanced encryption standard(AES)encryption for kerberos 5.Internet RFC 3962,February, 2005

[10] 10肖国镇,白恩健,刘晓娟.AES密码分析的若干新进展.电子学报,2003;(10):1549～1554

[11] 11 Joan Daemen Vincent Rijmen.高级加密标准(AES)算法Rijndael的设计.北京:清华大学出版社,2003

[12] 12 clark D D, Wilson D R. A Comparison of Commercial and military Computer Security policies. In IEEE Symposium on Computer Security and Privacy, 1987(4)

[13] 13 Bell D E, LaPadula L J. Secure Computer Systems: Mathematical Foundations. Technical Report. The MITRE Corporation, 1973:74~244

[14] 14 Bell D E, LaPadula L J. Secure Computer Systems: A Mathematical Model. Technical Report. The MITRE Corporation, 1973:74~244

[15] 15 Bell D E, LaPadula L J. Secure Computer Systems:A Refinement of the Mathematical Model. Technical Report M74-2547,The MITRE Corporation, 1973

[16] 16 Denning D E. A Lattice Model of Secure Information Flow. Communication of The ACM, 1976, 19(5):236~250

[17] 17 Goguen J A, Meseguer J. Security Policy and Security Models. In Proceedings of the 1982 IEEE Symposium on Security and Privacy, 1982, 11~20

[18] 18 Sandhu R S, Coyne E J,Feinstein H L. Role－Based Access Control Models. IEEE Computer, 1996, 29(2):38~47

[19] 19 The International Organization for Standardization. Common Criteria for Information Technology Security Evaluation-Part 1: Introduction and General Model, ISO/IEC 15408-1:1999(E)

[20] 20 The International Organization for Standardization. Common Criteria for

Information Technology Security Evaluation-Part 2: Security Functional Requirements, ISO/IEC 15408-2:1999(E)

[21] 21 The International Organization for Standardization. Common Criteria for Information Technology Security Evaluation-Part 3: Security Assurance Requirements, ISO/IEC 15408-3:1999(E)

[22] 22 U.S. Department of Defense.1985. Trusted Computer System Evaluation Criteria, DoD 5200.28-STD

[23] 23 Sandhu R S, Bhamidipati V, Munawer Q. The ARBAC97 Model for Role-Based Administration of Roles, ACM Transactions on Information and System Security, 1999, 2(1):105~135

[24] 24卿斯汉，刘文清，刘海峰. 操作系统安全导论. 北京：科学出版社，2003

[25] 25 陈爱民，于康友，管海明. 计算机的安全与保密. 北京：电子工业出版社，1992

[26] 26 中国国家质量技术监督局. 中华人民共和国国家标准：计算机信息系统安全保护等级划分准则，GB17859－1999

[27] 27阙喜戎，孙悦，龚向阳. 信息安全原理及应用. 北京：清华大学出版社，2003

[28] 28 谢冬青，冷健. PKI原理与技术. 北京：清华大学出版社，2004

[29] 29 Denning D E. Cryptography and Data Security. Addison-Wesley, 1982

[30] 30 方勇，刘嘉勇. 信息系统安全导论. 北京：电子工业出版社，2003

[31] 31 洪帆 等. 离散数学基础(第二版). 武汉：华中科技大学出版社，1995

[32]

[33] [] JG Ko, KY Kim and KW Ryu. Certification and Security in Inter-Organizational E-Service. Springer Boston, 2005

[34] [] URL: http://www.tp-link.com.cn/surpport/showarticle.asp?id=64

[35] [] 王芳,韩国栋,李鑫. 路由器访问控制列表及其实现技术研究. 计算机工程与设计,2007,28(23):5638~5640

[36]

[37] 孙晓蓉　徐春光　王育民，网络和分布式系统中的认证，计算机研究与发展，35（10）：865-868

[38] 原浩，移动通信中身份认证的研究，福建电脑，2005，6：33-36

[39] Blaze M, Feigenbaum J, Ioannidis J et al. The role of trust management in distributed systems security. In Secure Internet Programming: Issues for Mobile and Distributed Objects, Berlin: Springer-Verlag, 1999:185~210

[40] Blaze M, Feigenbaum J and Lacy J. Decentralized trust management[C]. In:

Proceedings of 17th Symposium on Security and Privacy, IEEE Computer Society Press, 1996:164~173

[41] Blaze M, Feigenbaum J, Keromytis D A. Keynote: trust management for public-key infrastructures[C]. In: 1998 Security Protocols International Workshop, Springer-Verlag, 1999:59~63

[42] Chu Y-H, Feigenbaum J, LaMacchia B et al. REFEREE:trust management for Web applications[J]. World Wide Web Journal, 1997(2):127~139

[43] Ellison C. SPKI/SDSI Certificate Documentation. http://world.std.com/~cme/html/spki.html

[44] Ellison C, Frantz B, Lampson B et al. SPKI Certificate Theory[S]. IETF RFC 2693, http://www.ietf.org/rfc/rfc2693.txt, 1999-09

[45] Ellison C, Frantz B, Lampson B et al. Simple Public Key Certificate[S]. Internet Draft(Work in Progress), http://world.std.com/~cme/spki.txt, 1999-07

[46] Ninghui Li, Grosof N B, Feigenbaum J. A logic-based knowledge representation for authorization with delegation(extend abstract)[C]. In:Proceedings of the 1999 IEEE Computer Security Foundations Workshop, IEEE Computer Society Press, 1999-06:162~174

[47] Ninghui Li, Grosof N B, Feigenbaum J. A practically implementable and tractable Delegtion Logic[C]. In:Proceedings of the 2000 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 2000-05:27~42

[48] Ninghui Li, Grosof N B, Feigenbaum J. Delegation Logic: A Logic-based Approach to Distributed Authorization[J]. ACM Transactions on Information and System Security(TISSSEC), 2003; 6(1):128~171

[49] Blaze, M., Feigenbaum, J., Ioannidis, J., et al. The KeyNote trust management system version 2[s]. IETF RFC 2704, http://www.ietf.org/rfc/rfc2704.txt, 1999-09

[50] N. Li, J. C. Mitchell, W. H. Winsborough. Design of a Role-based Trust-management Framework. In: Proceeding of the 2002 IEEE Symposium on Security and Privacy, Claremont Resort Oakland, California, USA, 2002. 114~130

[51] N. Li, J. C. Mitchell. RT: A role-based trust-management framework. In: B. Werner ed. Proceedings of the 2003 DARPA Information Survivablility Conference and Exposition (DISCEX'03). Washington, DC, USA. 2003. Los Alamitos: IEEE CS Press, 2003. 2(1): 201~212

[52] N. Li, J. C. Mitchell. Datalog with constraints: a foundation for trust-management

languages. In: V. Dahl and P. Wadler eds. Proceedings of the 15th International Symposium on Pratical Aspects of Declarative Languages (PADL 2003). LNCS 2562. New Orleans, LA, USA. 2003. Berlin Heidelberg: Springer-Verlag, 2003. 58~73

[53] N. Li, W. H. Winsborough, J. C. Mitchell. Distributed credential chain discovery in trust management. Journal of Computer Security, 2003, 11(1): 35~86

[54] K. E. Seamons, M. Winslett, T. Yu. Trust Negotiation in Dynamic Coalitions. In: B. Werner ed. Proceedings of the 2003 DARPA Information Survivability Conference and Exposition (DISCEX'03). Washington, DC, USA. 2003. Los Alamitos: IEEE CS Press, 2003, 2(2): 240~245

[55] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based Access Control Models. IEEE Computer, 1996, 29(2): 38~47

[56] L. L. Xin, C W. Min, H. S. Lian. Realizing Mandatory Access Control in Role-Based Security System. Journal of Software, (in Chinese with English abstract) 2000, 11(10): 1320-1325.

[57] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman. Role-based access control models. IEEE Computer, 29(2): 38-47, Feb, 1996.

[58] W. H. Winsborough, K. E. Seamons, V. E. Jones. Automated Trust Negotiation. In: Proceedings of DARPA Information Survivability Conference and Exposition. Hilton Head, South Carolina, Los Alamitos: IEEE press, January 2000, Volume I, 88~102

[59] Barlow T, Hess A, Seamons KE. Trust negotiation in electronic markets. In: Proc. of 8th Research Symp. In Emerging Electronic Markets. Maastricht, 2001.

[60] T. Yu, M. Winslett, K. E. Seamons. Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation. ACM Transactions on Information and System Security(TISSEC), February 2003, 6(1):1~42

[61] W. H. Winsborough, N. Li. Towards Practical Automated Trust Negotiation. In: Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002). Monterey, California, USA, IEEE CS Press, June 2002, 92~103

[62] W. H. Winsborough, N. Li. Safety in Automated Trust Negotiation. In: Proceedings of the IEEE Symposium on Security and Privacy. IEEE Press, May 2004, 147~160

[63] K. E. Seamons, M. Winslett, T. Yu, B. Smith, et al. Requirements for Policy Languages for Trust Negotiation. In: proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks, Monterey, California, Washington: IEEE Computer

Society Press, June 2002, 68~79

[64] T. Yu, M. Winslett. A Unified Scheme for Resource Protection in Automated Trust Negotiation. In: proceedings of the 2003 IEEE Symposium on Security and Privacy, Berkeley, CA, USA. Washington: IEEE Computer Society Press, May 2003. 110~122

[65] J. Li, N. Li, W. H. Winsborough. Automated Trust Negotiation Using Cryptographic Credentials. In: Proceeding of the 12th conference on computer and communications security, Alexandria, Virginia, USA. ACM Press, November, 2005, 46~57

[66] N. Li, W. Du, D. Boneh. Oblivious Signature-Based Envelope. In: Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003). Boston, Massachusetts, USA, New York: ACM Press, July 2003, 182~189

[67] W. H. Winsborough, N. Li. Protecting Sensitive Attributed in Automated Trust Negotiation. In: Proceedings of the 1st ACM Workshop on Privacy in the Electronic Society. ACM Press, 2002, 41~51

[68] J. E. Holt, R. W. Bradshaw, K. E. Seamons, H. Orman. Hidden Credentials. In: Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society, Washington, DC. ACM Press, October 2003, 1~8

[69] K. Frikken, M. Atallah, J. Li. Hidden Access Control Policies with Hidden Credentials. In: Proceedings of the 3rd ACM Workshop on Privacy in the Electronic Society, Washington, DC. ACM Press, October 2004, 27~28

[70] R. W. Bradshaw, J. E. Holt, K. E. Seamons. Concealing Complex Policies with Hidden Credentials. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington, DC. ACM Press, October 2004, 146~157

[71] J. Li, N. Li. OACerts: Oblivious Attribute Certificates. In: Proceedings of the 3rd Intenational Conference on Applied Cryptography and Network Security (ACNS 2005), New York, USA. Springer, 2005, Volume 3531 of Lecture Notes in Computer Science, 301~316

[72] C. Castelluccia, S. Jarecki, G. Tsudik. Secret Handshakes from Ca-oblivious Encryption. In Advances in Cryptology – ASIACRPT 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security. Springer, 2004, Volume 3329 of Lecture Notes in Computer Science, 293~307

[73] E. Bertino, E. Ferrrari, A.Squicciarini. Privacy-Preserving Trust Negotiation. Technical Report CERIAS-TR-2004-75, Center for Education and Research in Information Assurance and Security, Purdue University, 2005

[74] 张荣清,李建欣,怀进鹏.网格计算环境中的安全信任协商系统.北京航空航天大学学报.2006,32(3):347~351

[75] 李建欣, 怀进鹏, 李先贤. 自动信任协商研究. 软件学报, 2006. 17(1): 124~133

[76] 廖振松, 金海, 李赤松, 邹德清. 自动信任协商及其发展趋势. 软件学报, 2006, 17(9):1933~1948

[77] ITU-T Recommendation X.509, The Directory : Public Key and Attribute Certificate Frameworks, 2003.

[78]

[79] Farrell S, Housley R. An Internet Attribute Certificate Profile for Authorization, RFC3281, 2002.

[80]

[81] John Linn, Magnus Nystrom. Attribute Certification: An Enabling Technology for Delegation and Role-based Controls in Distributed Environments. Proceedings of the 4th ACM Workshop on Role-based Access Control, 1999. 121-130.

[82]

[83] Chadwick D, Otenko A. The PERMIS X.509 Role Based Privilege Management Infrastructure. ACMAT, Monterey, California, USA, 2002.

[84]

[85] Hwang Jingjang, Wu Kouchen, Liu Duenren. Access Control with Role Attribute Certificates. Computer Standards & Interfaces, 2000, 22: 43-53

[86]

[87] Wohlmacher P, Pharow P. Application in Health Care Using Public Key Certifications and Attribute Certificates. IEEE Journal, 2000:1063-9527

[88]

[89] Doshi V, Fayad A, Jajodia S, et al. Using Attribute Certificates with Mobile Policies in Electronic Commerce Applications. IEEE Journal, 2000:1063-9527