第四章 访问控制

- 4.1 访问控制原理
- 4.2 主体、客体和访问权
- 4.3 自主访问控制
- 4.4 实例：UNIX文件访问控制
- 4.5 基于角色的访问控制
- 4.6 基于属性的访问控制
- 4.7 身份、凭证和访问管理
- 4.8 信任框架
- 4.9 案例学习：银行的RBAC系统

4.1 访问控制原理

- NIST IR 7298（《信息安全关键名词术语，2013年5月）定义访问控制为授予或拒绝下列特定要求的过程：(1) 获得并使用信息及相关信息处理服务；(2) 进入特定物理设施。

- RFC 4949《Internet安全术语》定义访问控制为这样一个过程：实现依据安全策略对使用系统资源进行控制，且仅许可授权实体（用户、程序、进程或其他系统）依据该策略使用系统资源。

4.1 访问控制原理

NIST SP 800-171（保护非联邦信息系统和组织的受控非保密信息，2016年8月）提供了一个访问控制服务安全要求的有效列表。

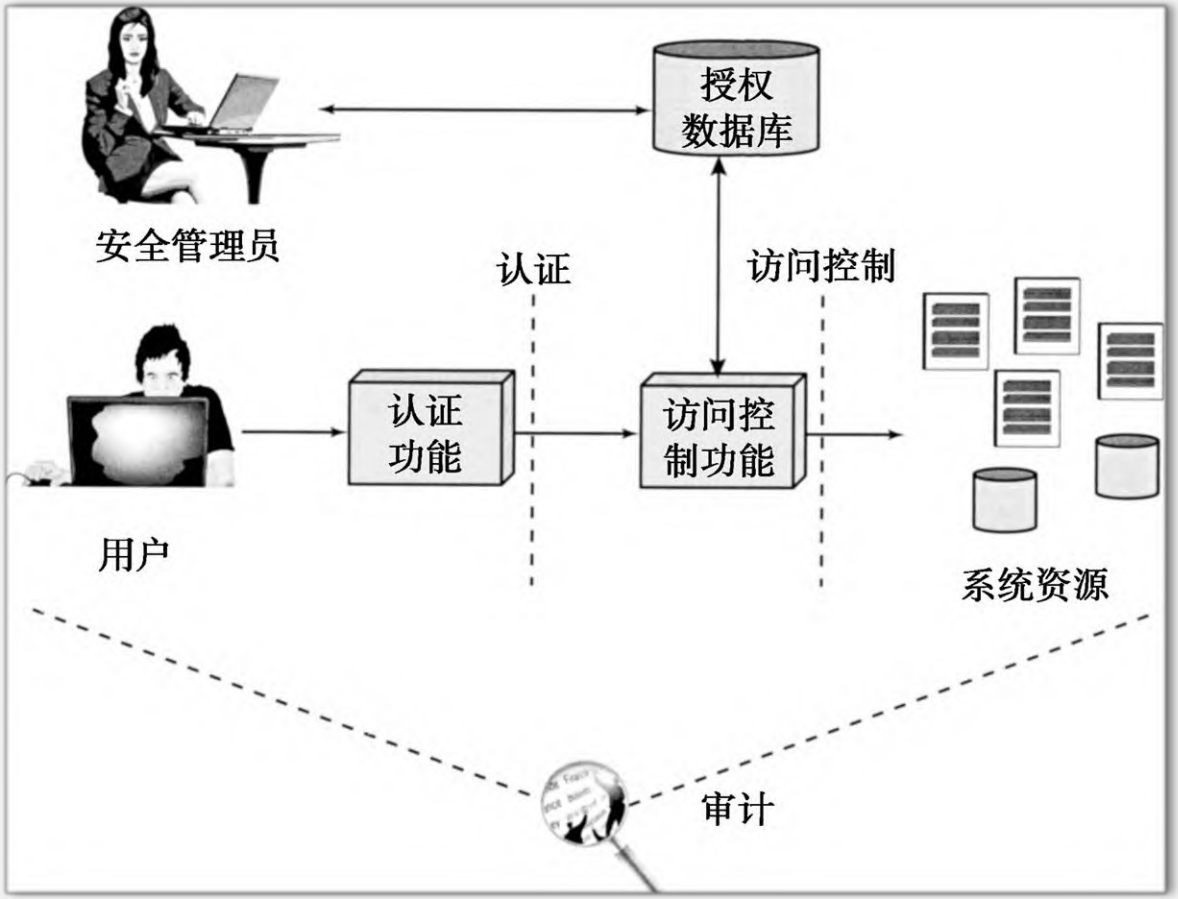
基本安全要求	1. 限制信息系统对授权用户、代表授权用户的进程或设备的访问（包括其他信息系统）。 2. 限制信息系统对各种类型的事务和授权用户允许执行的功能的访问。
派生的安全要求	3. 根据批准的授权控制CUI流。 4. 分离个人职责以减少不共谋的恶意活动的风险。 5. 采用最小权限原则，包括特定安全功能和特权账号。 6. 当访问非安全功能时，使用非特权账号或角色。 7. 阻止非特权用户执行特权功能和执行这些功能的审计。 8. 限制不成功的登陆尝试。 9. 提供与合适的CUI规则一致的隐私和安全注意通知。 10. 使用带有模式隐藏显示的会话锁，以防止在未活动期间访问和查看数据。 11. 在定义的条件后（自动）终止一个用户会话。 12. 监测和控制远程访问会话。 13. 使用密码机制保护远程访问会话的机密性。 14. 通过受管理的访问控制点路由远程访问。 15. 授权特权命令的远程执行和安全相关的信息的远程访问。 16. 在允许连接之前授权无线访问。 17. 使用认证和加密保护无线访问。 18. 控制移动设备的连接。 19. 在移动设备上加密CUI。 20. 对外部信息系统的连接和使用进行验证和控制/限制。 21. 限制在外部信息系统上使用组织的移动存储设备。 22. 控制CUI在公共可访问系统上发布或处理。

4.1.1 访问控制语境

广义来讲，所有的计算机安全都与访问控制有关。

RFC 4949定义计算机安全如下：

“用来实现和保证计算机系统的安全服务的措施，特别是保证访问控制服务的措施。”



访问控制与其它安全功能的关系

4.1.2 访问控制策略

- 自主访问控制（discretionary access control，DAC）：基于请求者的身份和访问规则（授权）控制访问，规定请求者可以（或不可以）做什么。这种策略被称为“自主的”是因为允许一个实体按其自己的意志授予另一个实体访问某些资源的权限。
- 强制访问控制（mandatory access control，MAC）：通过比较具有安全许可（表明系统实体有资格访问某种资源）的安全标记（表明系统资源的敏感或关键程度）来控制访问。这种策略被称为“强制的”是因为一个具有访问某种资源的许可的实体不能按其自己的意志授予另一个实体访问那种资源的权限。
- 基于角色的访问控制（role-based access control，RBAC）：基于用户在系统中所具有的角色和说明各种角色用户享有哪些访问权的规则来控制访问。
- 基于属性的访问控制（attribute-based access control，ABAC）：基于用户、被访问资源及当前环境条件来控制访问。

4.2 主体、客体和访问权

主体

能够访问客体的实体

三类主体：

所有者；组；世界

客体

外界对其访问受到控制的资源

客体是一个用来包含和/或接收信息的实体

访问权

描述了主体可以访问客体的方式

可包括:

读；写；执行；删除；创建；搜索

4.3 自主访问控制

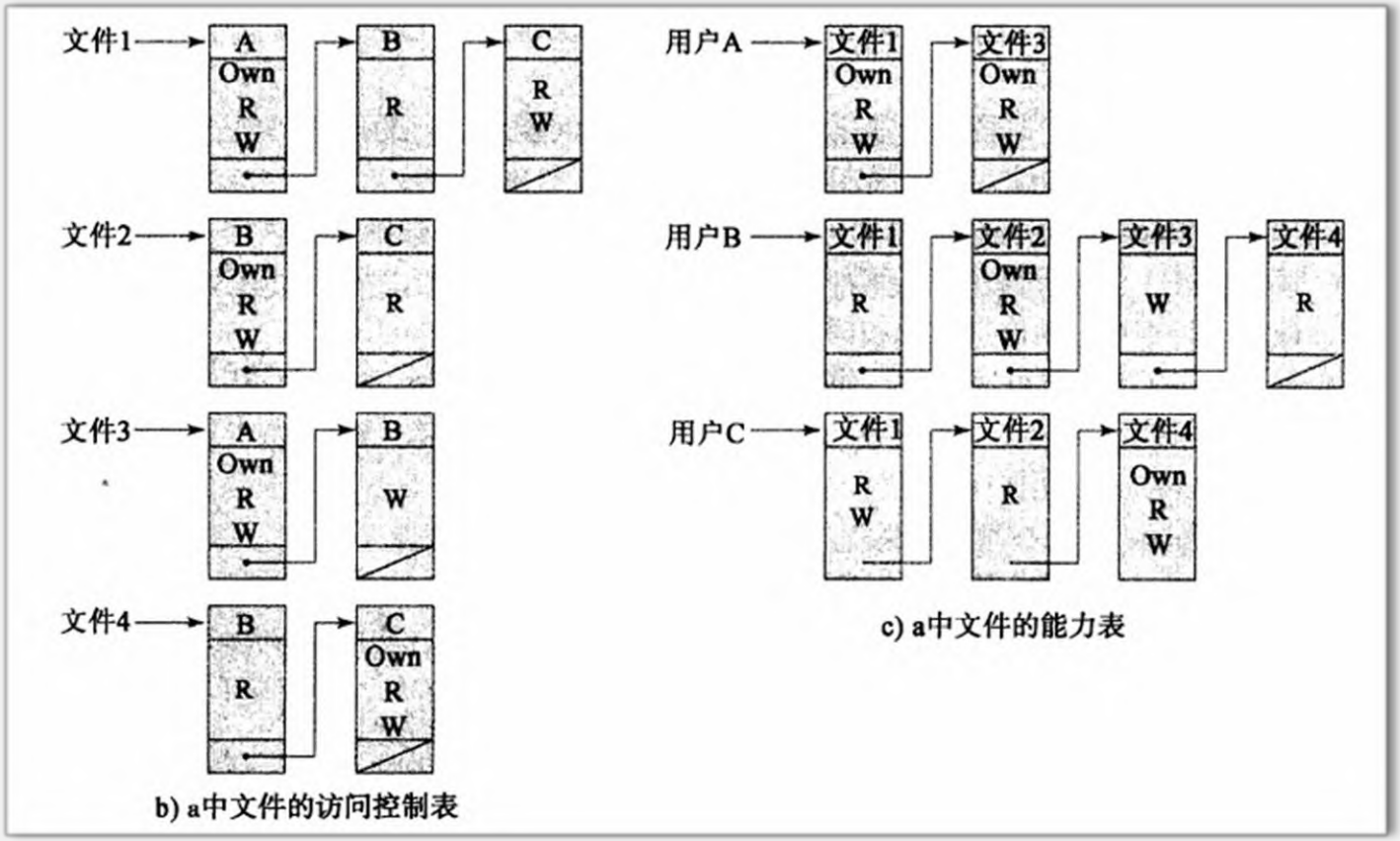
- 如前所述，自主访问控制方案是指一个实体可以被授权按其自己的意志使另一个实体能够访问某些资源。
- DAC（discretionary access control，自主访问控制）的一种通常方式是在操作系统或数据库管理系统中运用的访问矩阵（access matrix）。
  - 访问矩阵的概念由Lampson [LAMP69，LAMP71]系统提出，随后由Graham与Denning [GRAH72]、Harrsion等人[HARR76]细化。
  - 矩阵中的一维由试图访问资源的被标识的主体组成。
  - 另一维列出可以被访问的客体。
  - 矩阵中的每项表示一个特定主体对一个特定客体的访问权。

4.3 访问矩阵的一个简单例子

	文件1	文件2	文件3	文件4
用户A	Own Read Write		Own Read Write	
用户B	Read	Own Read Write	Write	Read
用户C	Read Write	Read		Own Read Write

由图可知，用户A拥有文件1和3并具有对这些文件的读、写权限，用户B具有对文件1的读权限，依此类推。

4.3 访问矩阵的分解



将矩阵按列分解，产生访问控制表（access control list，ACL），见图b。

按行分解产生能力权证（capability ticket），见图c。

4.3 授权表

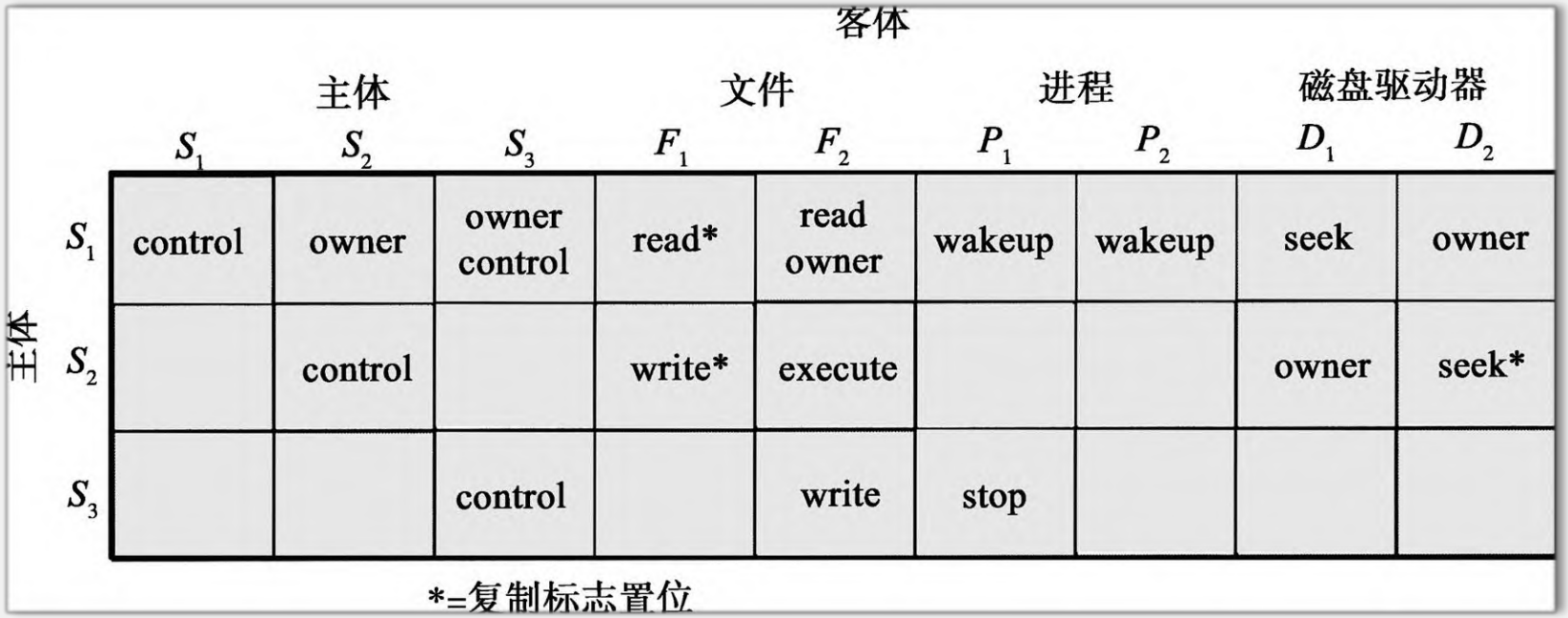
主体	访问模式	客体	主体	访问模式	客体
A	Own	文件 1	B	Write	文件 2
A	Read	文件 1	B	Write	文件 3
A	Write	文件 1	B	Read	文件 4
A	Own	文件 3	C	Read	文件 1
A	Read	文件 3	C	Write	文件 1
A	Write	文件 3	C	Read	文件 2
B	Read	文件 1	C	Own	文件 4
B	Own	文件 2	C	Read	文件 4
B	Read	文件 2	C	Write	文件 4

[SAND94]提出了一种数据结构，它不像访问矩阵那么稀疏，但比ACL或能力表更为方便。授权表中的一行对应于一个主体对一种资源的一种访问权。按主体排序或访问该表等价于能力表。按客体排序或访问该表等价于ACL。关系数据库很容易实现这种类型的授权表。

4.3.1 一个访问控制模型

- 本节介绍由Lampson、Graham和Denning [LAMP71, GRAH72，DENN71]开发的一个DAC通用模型。
- 该模型假定了一组主体、一组客体以及一组控制主体访问客体的规则。
- 把系统的保护状态定义为在一定的时间点指定每个主体对每个客体的访问权的信息集。可以识别出三种需求：表示保护状态、执行访问权以及允许主体以某些方式更改保护状态。该模型给出了DAC系统的一个通用的逻辑描述，满足所有这三种需求。

4.3.1 一个访问控制模型



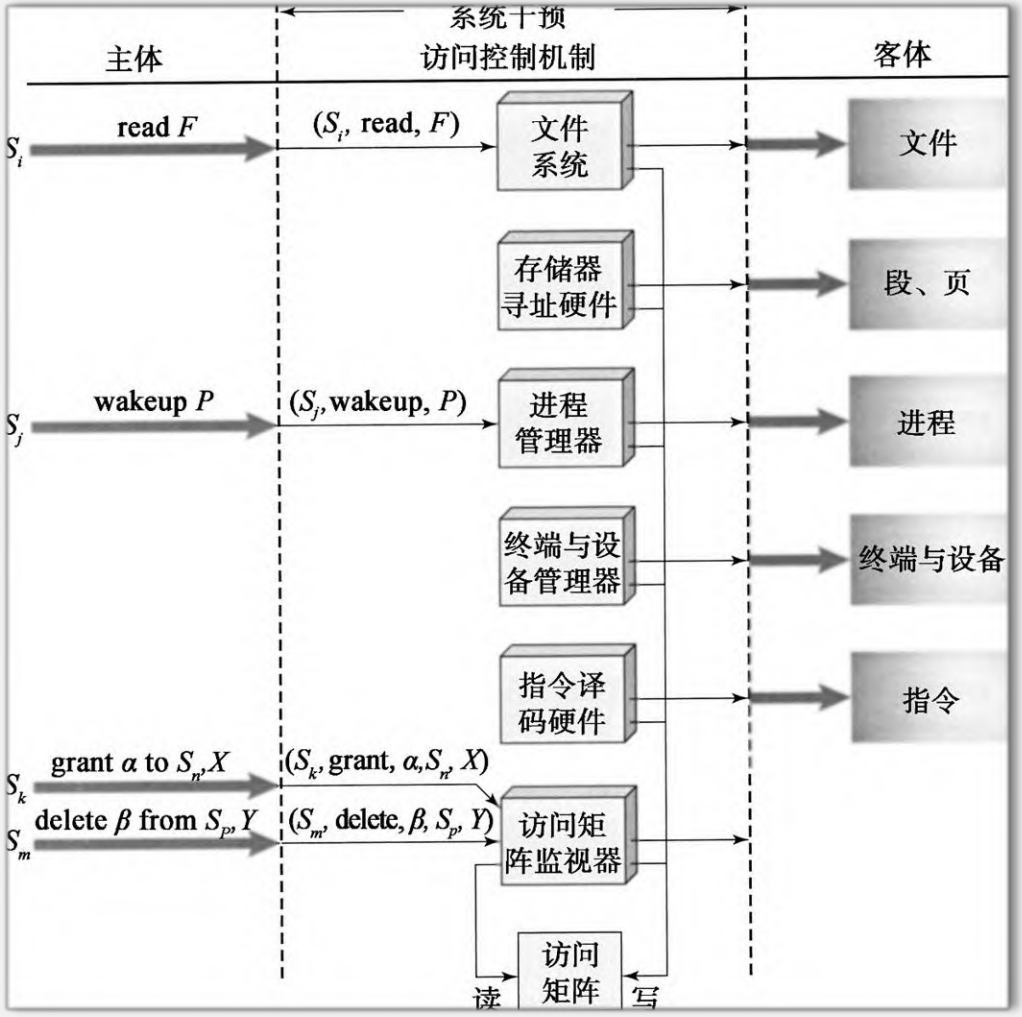
主体 \ 客体	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read*	read owner	wakeup	wakeup	seek	owner
S_2		control		write*	execute			owner	seek*
S_3			control		write	stop			

*=复制标志置位

对于访问控制矩阵A，其中的每一项A[S, X]都包含被称为访问属性的字符串，用来指定主体S对客体X的访问权。例如，S1可以读取文件F2，因为“read”出现在A[S1, F1]中。

4.3.1 一个访问控制模型

主体对客体的每次访问通过该客体的控制器完成。控制器的决定取决于矩阵的当前内容。此外，某些主体具有对访问矩阵进行特定修改的权力。修改访问矩阵的请求被看作对矩阵的一次访问，其中将矩阵的单个项作为客体。这些访问通过控制矩阵更新的访问矩阵控制器完成。



4.3.1 一个访问控制模型

规　则	命令（S_0 发出）	授　权	操　作
R1	transfer $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ to S, X	“α^*”在 $A[S_0, X]$ 中	将 $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ 存储到 $A[S, X]$ 中
R2	grant $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ to S, X	“owner”在 $A[S_0, X]$ 中	将 $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ 存储到 $A[S, X]$ 中
R3	delete α from S, X	“control”在 $A[S_0, X]$ 中 或 “owner”在 $A[S_0, X]$ 中	从 $A[S, X]$ 中删除 α
R4	$w \leftarrow$ read S, X	“control”在 $A[S_0, X]$ 中 或 “owner”在 $A[S_0, X]$ 中	复制 $A[S_0, X]$ 到 w
R5	create object X	无	在 A 中插入 X 对应的列；将“owner”存储到 $A[S_0, X]$ 中
R6	destroy object X	“owner”在 $A[S_0, X]$ 中	在 A 中删除 X 对应的列
R7	create subject S	无	在 A 中插入 S 对应的行；执行 create object S；将“control”存储到 $A[S, S]$ 中
R8	destroy subject S	“owner”在 $A[S_0, X]$ 中	在 A 中删除 S 对应的行；执行 destroy object S

这个模型还包括控制修改访问矩阵的一组规则，如表所示。前三个规则用来处理转授、授予和删除访问权，其余规则用来控制主体和客体的创建与删除。

4.3.2 保护域

- 保护域（protection domain）是一组客体及对这些客体的访问权。
- 保护域这个更一般的概念用来提供更多的灵活性。例如，用户可以通过定义一个新的保护域来创建权限为其访问权子集的进程。这就限制了进程的能力。
- 进程与保护域之间的关联可以是静态的，也可以是动态的。例如进程可以执行一个过程序列，对其中每个过程要求不同的访问权。
- 保护域的一种形式与很多操作系统（如UNIX）的用户与内核模式的区别有关。

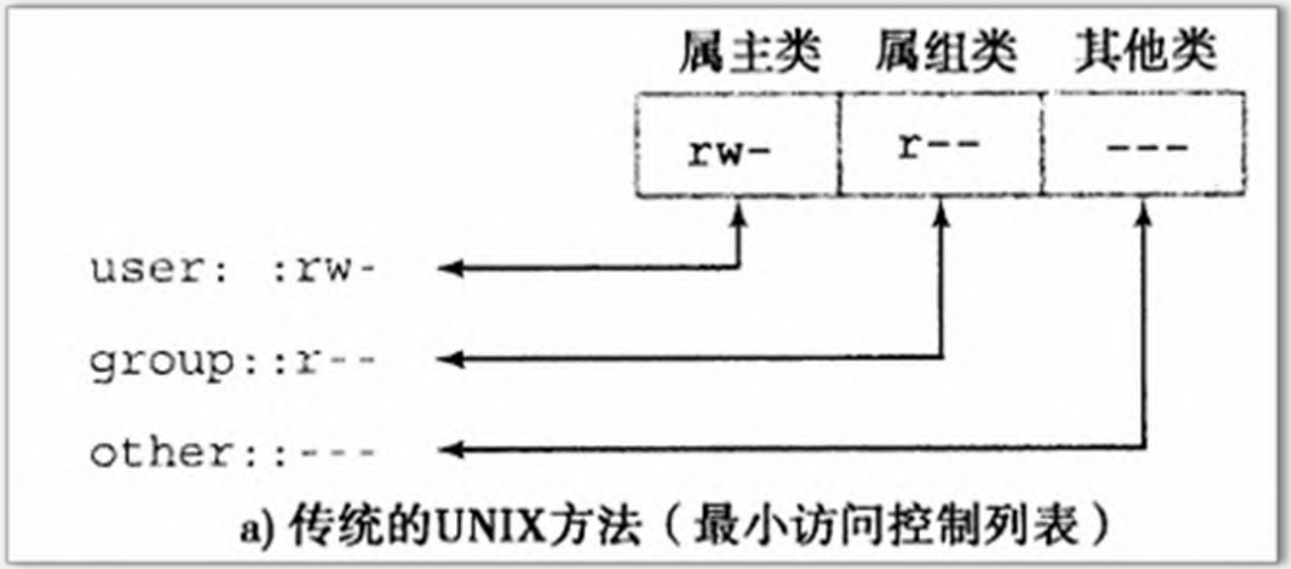
4.4 实例：UNIX文件访问控制

- 数据加密标准（DES），它是使用最广泛的加密体制，1977 年被美国国家标准局（现国家标准和技术研究所，NIST）采纳为联邦信息处理标准 FIPS PUB 46。
- DES采用 64 位长度的明文分组和 56 位长度的密钥，产生 64 位长度的密文分组。
- DES（数据加密标准）的强度：就算法本身，多年来人们尝试发现其弱点，但至今无致命弱点报道；而对于 56 位密钥的使用，考虑到现在的商用处理器速度，该密钥长度是严重不足的。

4.4 UNIX的文件和目录

- 所有类型的UNIX文件都由操作系统通过inode管理。
  inode（index node，索引结点）是包含操作系统对一个文件所需的关键信息的控制结构。几个文件名可以与一个inode关联，但一个活动inode仅与一个文件关联，一个文件也仅被一个inode控制。文件的属性及访问许可和其它控制信息都存储在inode中。在磁盘上有个inode表，包含文件系统中所有文件的inode。打开一个文件时，它的inode被读进主存，存储在驻留内存的inode表中。
- 目录呈分层树状结构。每个目录包含文件和/或其它目录。包含在另一个目录中的目录被称为子目录。目录仅仅是一个包含文件名和指向关联inode的指针的列表。因而，每个目录都与其自己的inode关联。

4.4.1 传统的UNIX文件访问控制



a) 传统的UNIX方法（最小访问控制列表）

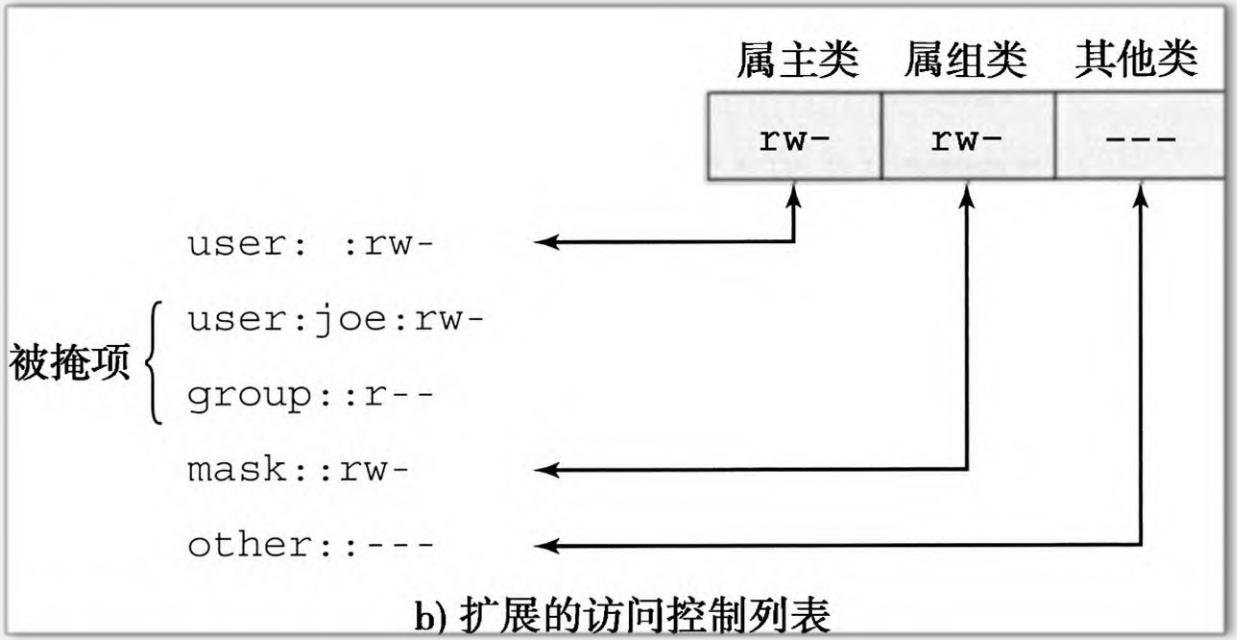
大多数UNIX系统都依赖或者至少是基于UNIX早期版本引入的文件访问控制方案。与每个文件相关联的是12个保护位的组合。属主ID、属组ID和保护位都是文件inode的一部分。

九个保护位分别用来指定文件属主、同组用户与其他用户的读、写和执行许可。剩余的三个位定义了文件或目录的其它特殊行为，其中两个是“设置用户ID”（SetUID）和“设置组ID”（SetGID）许可。最后一个许可位是“粘滞（sticky）”位，当对文件设置该位时，最初表示系统应在文件执行后将其内容保留在内存中，现在已不使用。

4.4.2 UNIX中的访问控制列表

- 很多现代UNIX及基于UNIX的操作系统都支持访问控制表，其中包括FreeBSD、OpenBSD、Linux和Solaris。
- 本节将描述FreeBSD中的访问控制表，不过其它实现在本质上也具有相同的特征和接口。这种特征被称为扩展的访问控制表，而传统UNIX方法被称为最小访问控制表。
- FreeBSD允许管理员通过setfacl命令为文件分配一个UNIX用户ID和组的列表。任何数目的用户和组都可以通过三个保护位（读、写、执行）与文件关联，这提供了分配访问权的一种灵活机制。文件不是必须具有ACL，也可以仅用传统的UNIX文件访问机制保护。
- FreeBSD文件包括一个附加的保护位，用来指出文件是否具有扩展的ACL。

4.4.2 UNIX中的访问控制列表

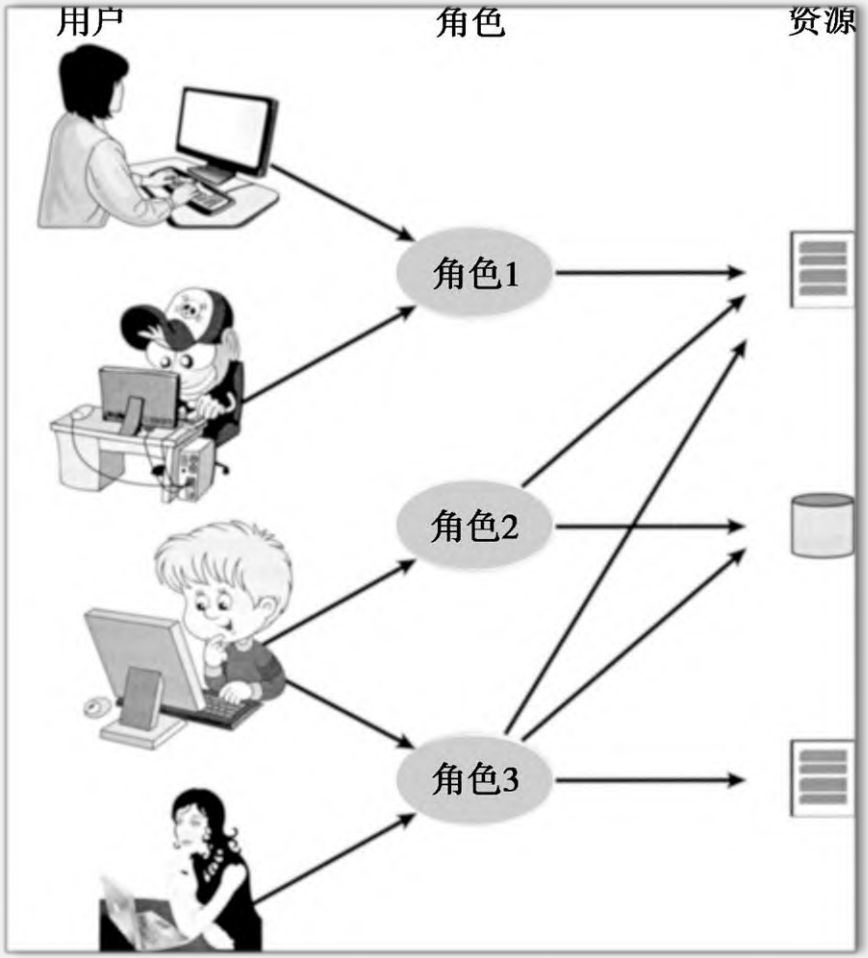


b) 扩展的访问控制列表

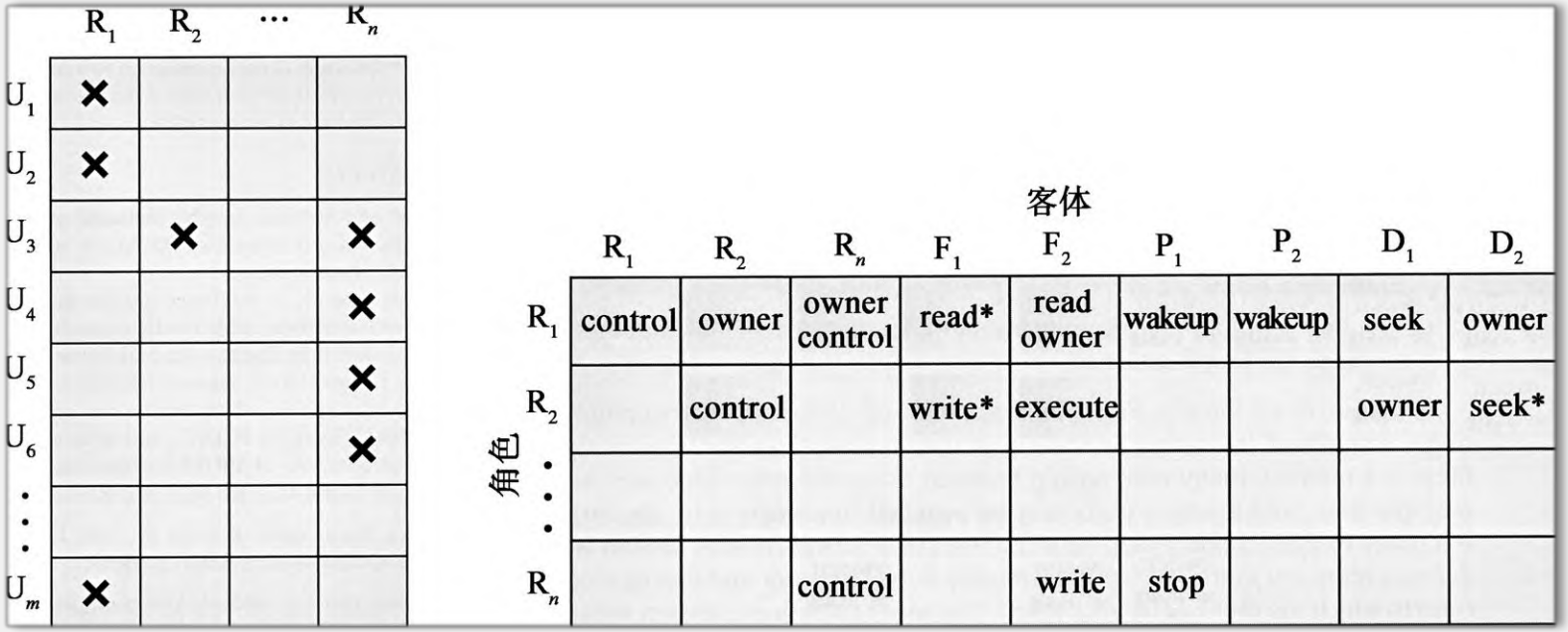
- 9位许可字段中的属主类和其他类项目与最小ACL中的含义相同。
- 属组类项目指定了属组对该文件的访问许可。这些许可还代表可以分配给属主之外的命名用户或命名组的最大访问许可。在后一个角色中，属组类项目起到掩码的作用。
- 附加的命名用户和命名组可以通过3位许可字段与文件关联。将命名用户或命名组的许可与掩码字段相比较，任何不在掩码字段中出现的许可都不被允许。

4.5 基于角色的访问控制

- 传统的DAC系统定义了单独的用户和用户组的访问权。与之相反，RBAC基于用户在系统中设定的角色而不是用户的身份。
- 一般地，RBAC模型定义角色为组织中的一项工作职责。RBAC系统给角色而不是给单独的用户分配访问权。
- 用户与角色的关系是多对多的，角色与资源或系统对象的关系也是多对多的



4.5 基于角色的访问控制

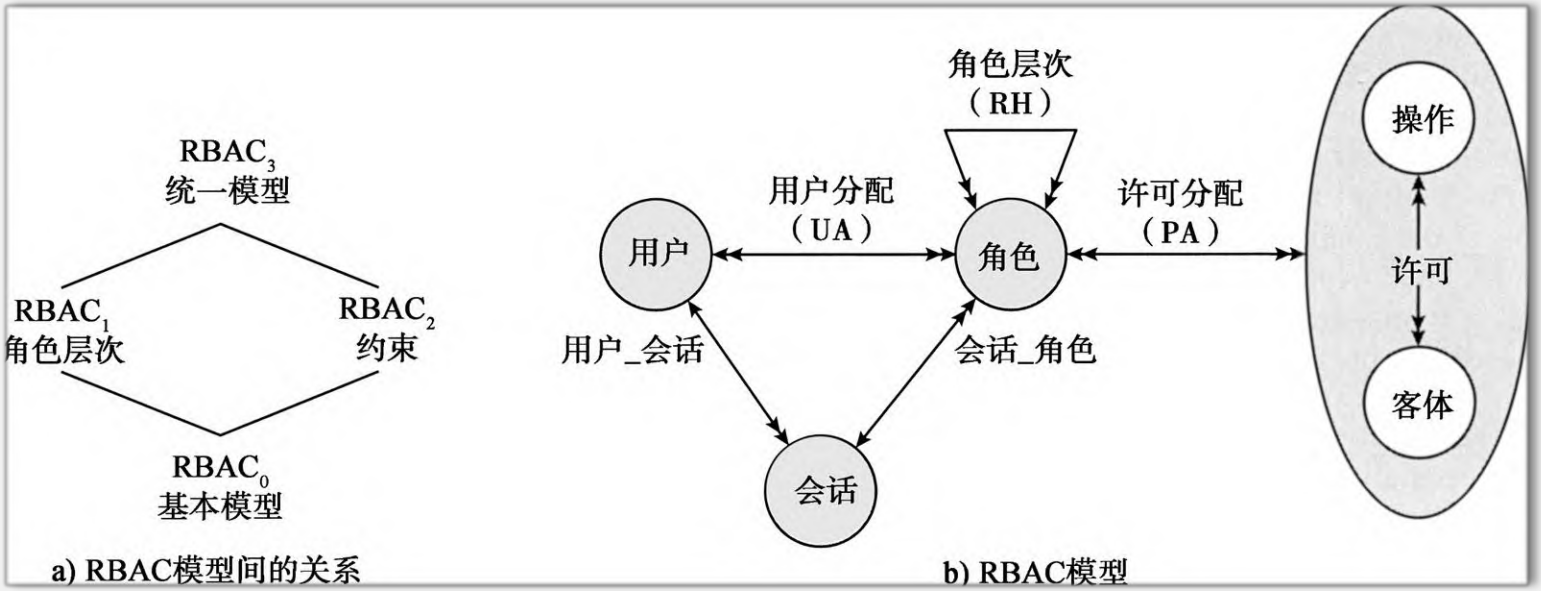


	R_1	R_2	…	R_n
U_1	×			
U_2	×			
U_3		×		×
U_4				×
U_5				×
U_6				×
⋮				
U_m	×			

角色 \ 客体	R_1	R_2	R_n	F_1	F_2	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read*	read owner	wakeup	wakeup	seek	owner
R_2		control		write*	execute			owner	seek*
⋮									
R_n			control		write	stop			

- 我们可以用访问矩阵来简单描述RBAC系统中的关键元素。
- 左侧的矩阵将单个用户与角色联系起来。一般情况下用户比角色多得多。
- 每个矩阵项或者为空，或者被标记，后者表示该用户被分配给该角色。注意一个用户可以被分配多个角色
- 右侧的矩阵具有与DAC访问控制矩阵相同的结构，其中将角色作为主体。一般地，角色少而客体或资源多。

4.5 RBAC参考模型

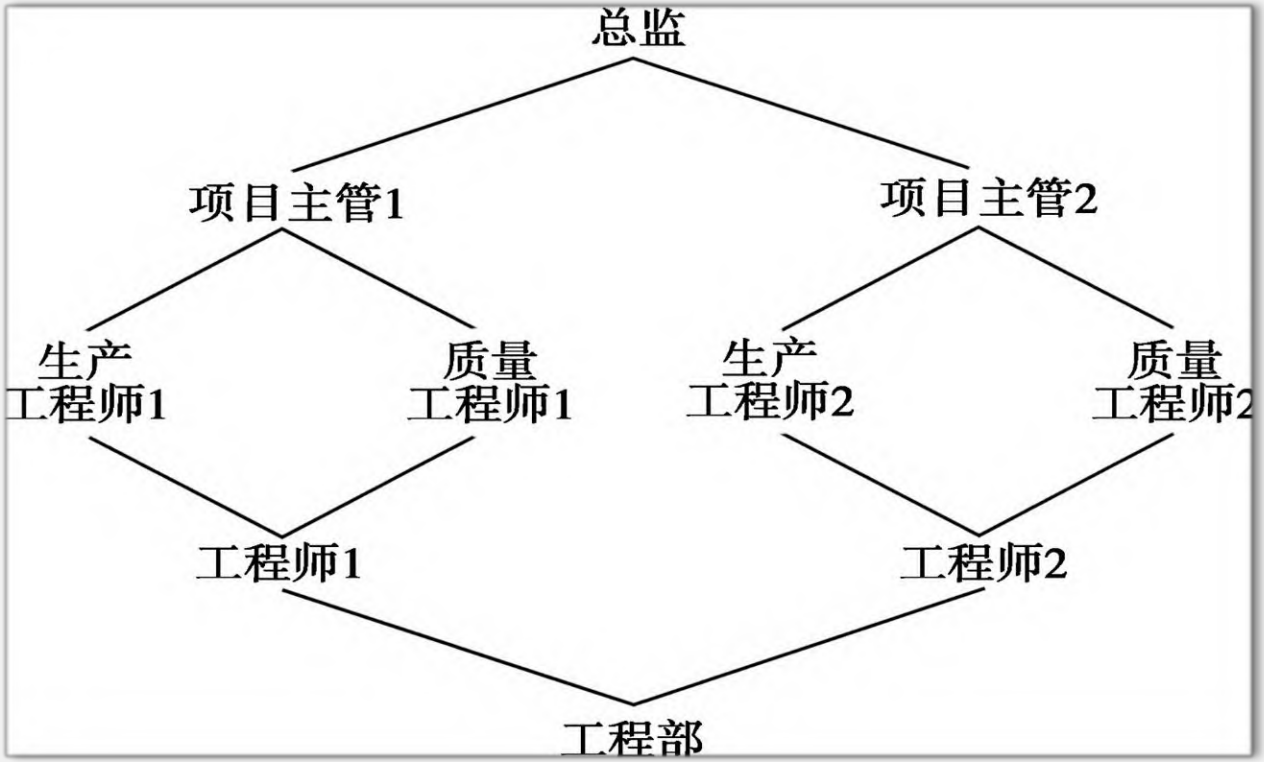


- 通用RBAC方法包括很多功能和服务。为了阐述RBAC的各个方面，有必要定义RBAC功能性的一组抽象模型。
- 上图是基于角色的访问控制模型家族。$RBAC_0$是访问控制系统的最低要求，$RBAC_1$增加了角色层次，$RBAC_2$增加了约束，$RBAC_3$包括$RBAC_1$和$RBAC_2$。

4.5 RBAC模型作用域

模　型	层　次	约　束
$RBAC_0$	否	否
$RBAC_1$	是	否
$RBAC_2$	否	是
$RBAC_3$	是	是

4.5 $RBAC_1$参考模型



- 角色层次——$RBAC_1$
- 角色层次提供了一种反映组织中角色层次结构的方式。
- 一般地，责任越大的工作岗位获得的访问资源的权力越多。

4.5 $RBAC_2$参考模型-约束

互斥角色

- 指一个用户只能被分配给集合中的一个角色。
- 一个用户（在会话中或静态地）只能被分配集合中的一个角色。
- 任何许可（访问权）只能被授予集合中的一个角色。

基数

- 指设置关于角色的最大数值。
- 这种类型的一个约束是设置可以分配给一个指定角色的最大用户数。
- 另一种形式的约束是设定可以被授予某个特定许可的最大角色数，这对于敏感的或功能强大的许可是很有意义的。

先决条件

- 用来规定如果已被分配另一个指定角色时，用户只能被分配一个特定角色。
- 先决条件可以用来构建最小特权概念的实现。
- 在一个层次中，可以要求用户仅当已被分配直接下级（低级）角色，才能被分配上级（高级）角色。

4.6 基于属性的访问控制-ABAC模型

- 访问控制技术的一项较新的进展是基于属性的访问控制（attribute-based access control，ABAC）模型。
- ABAC模型能够定义表达资源和主体二者属性条件的授权。
- ABAC方法的优势在于它的灵活性以及表达能力。
- [PLAT13]指出ABAC应用于真实系统的主要障碍是，需要考虑每次访问对资源和用户属性的评价所造成的性能影响。
- 但对于某些应用诸如Web服务和云计算的结合运用，每次访问所增加的性能代价相对于本已相当高的性能代价是微不足道的。
- ABAC模型有三个关键要素：
  - 属性，为配置中的实体而定义
  - 策略模型，定义ABAC策略
  - 架构模型，应用于实施访问控制的策略

4.6.1 ABAC模型中的属性

主体属性

主体是一个主动的实体（如用户、应用、进程或设备）

能引起客体间的信息流动或者系统状态的改变

每个主体都有能够定义其身份和特征的关联属性。

客体属性

客体，也被称为资源，是一个（在给定请求的语境中）被动的包含或接收信息的与信息系统相关的实体

与主体一样，客体具有可以用来制定访问控制决策的属性。

环境属性

这类属性到目前为止，在很大程度上被大多数访问控制规则所忽视

它们描述了信息访问发生时所处的运行的、技术的、甚至态势的环境或情境。

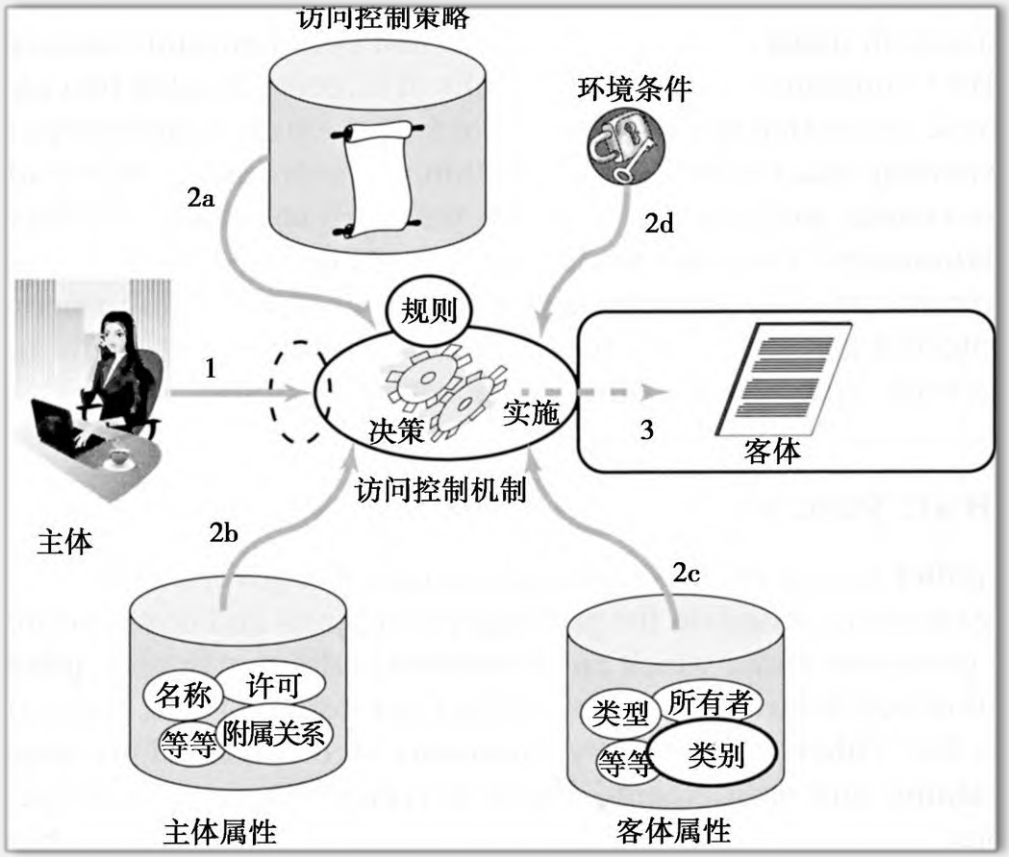
4.6.2 ABAC逻辑架构

右图说明了ABAC系统的基本组件的逻辑架构。主体对客体的一次访问将遵循下列步骤进行:

① 主体向客体提出访问请求。该请求被路由到一个访问控制装置。

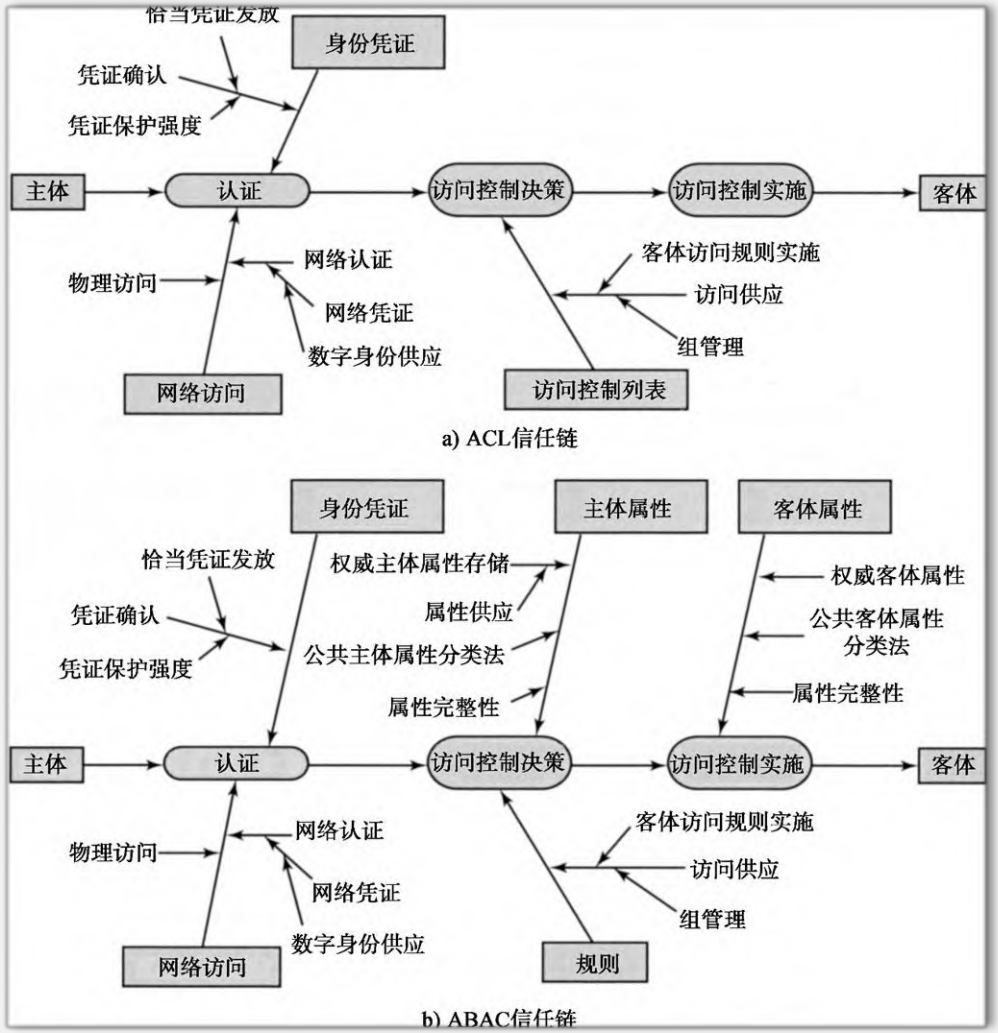
② 该访问控制装置通过一组由预先配置的访问控制策略所定义的规则（2a）进行控制。基于这些规则，访问控制装置对主体（2b）、客体（2c）和当前环境条件（2d）的属性进行评估，决定是否授权。

③ 若访问获得授权，则访问控制机制授权主体访问客体；若访问未被授权，则拒绝访问。



4.6.2 ABAC逻辑架构

- 右图提供了一种实用的方法来理解ABAC模型与采用访问控制表（ACL）的DAC模型的范围的比较。
- 该图不仅说明了这两种模型的相对复杂度，还阐明了这两种模型的信任要求。
- 对于ACL和ABAC分别采用的有代表性的信任关系（用箭头线表示）的比较表明，要想ABAC正常工作，需要许多更加复杂的信任关系。



4.6.3 ABAC策略

策略（policy）是一组用来管理组织内部的允许行为的规则和关系，其基础是主体所具有的特权，以及在哪种环境条件下资源或客体需要被保护。

策略通常是从需要保护的客体以及主体可用的特权角度编写的。

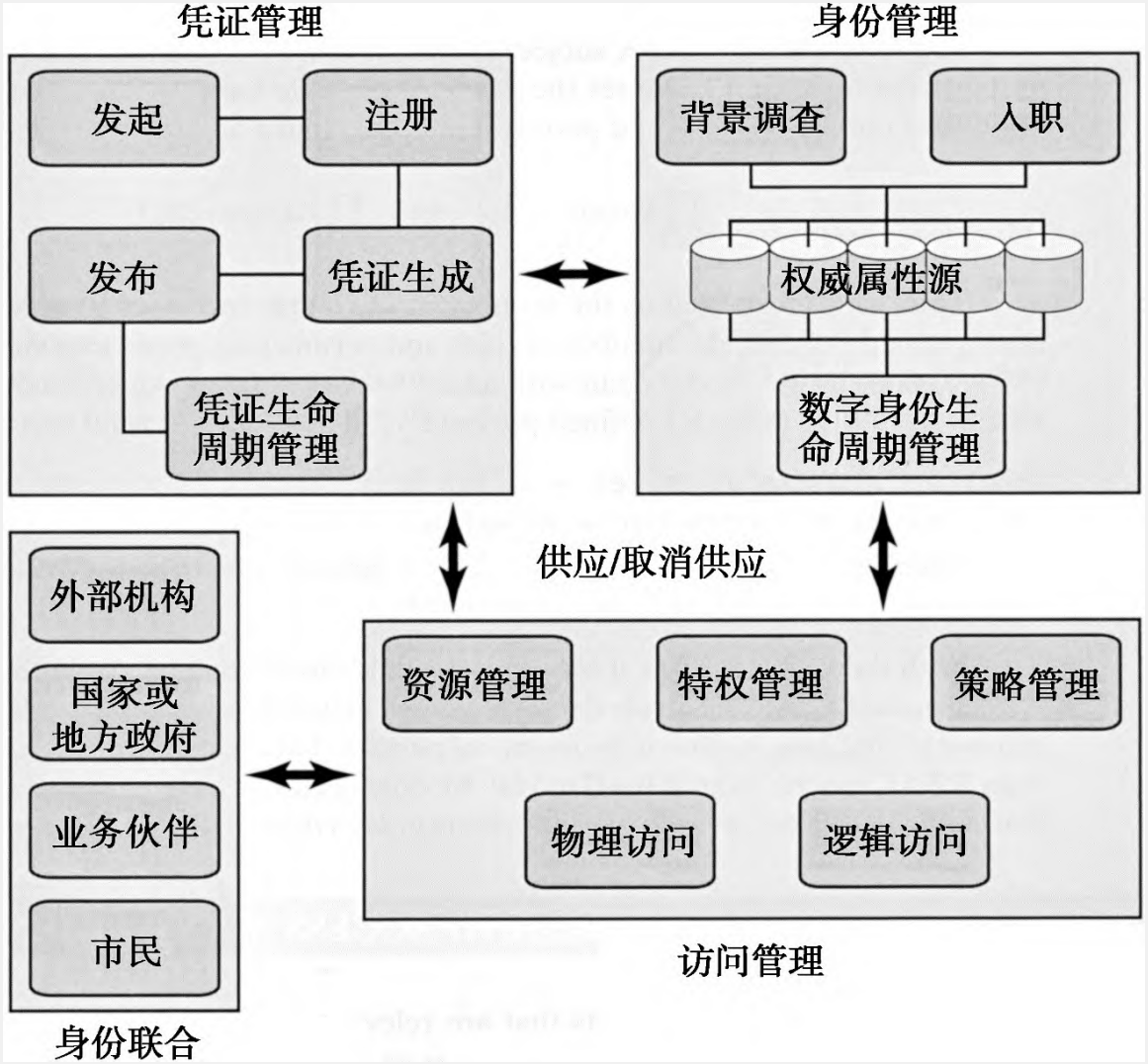
反过来，特权（privilege）代表主体的授权行为，它们由机构定义并体现在策略中。

其他常常用来代替特权的术语有权利（right）、授权（authorization）和资格（entitlement）

4.7 身份、凭证和访问管理-ICAM

- ICAM是一种用来管理和实现数字身份（及相关属性）、凭证和访问控制的综合性方法。
- ICAM由美国政府开发，但并非仅适用于政府机构，还可以由寻求访问控制统一方法的企业来部署。
- ICAM旨在：
  - 创建个体以及ICAM文件中所谓的“非人实体（NPE）”的可信数字身份表示。后者包括寻求资源访问许可的进程、应用程序和自动化设备。
  - 将这些身份绑定到可能为个体或NPE提供访问交易代理的凭证。凭证是一个对象或数据结构，将身份（及可选的附加属性）权威地绑定到用户所拥有并控制的权标。
  - 使用凭证对机构资源提供授权访问。

4.7 身份、凭证和访问管理-ICAM



4.7.1 身份管理

- 身份管理关注的是将属性分配到数字身份上去，并且将数字身份与个体或NPE连接起来。
- 其目标是建立一个独立于特定应用或情境的可信的数字身份。
- 传统的且仍在广泛使用的应用和程序访问控制方法是为使用这些资源创建一个数字化表示的身份。结果，维护和保护身份自身被视为仅次于与应用相关的任务。
- 身份管理的最后一个要素是生命周期管理，包括以下内容：
  - 保护个人身份信息的机制、策略和规程
  - 控制对身份数据的访问
  - 用于将权威身份数据分享给相关应用的技术
  - 撤销企业身份

4.7.2 凭证管理

- 凭证是一个对象或数据结构，将身份（及可选的附加属性）权威地绑定到用户所拥有并控制的权标。
- 凭证的实例包括智能卡、私有/公开密钥和数字证书。凭证管理是对凭证生命周期的管理。
- 凭证管理包括下列五个逻辑组件：
  ① 授权的个体发起需要凭证的个体或实体建立对凭证的需求。例如，部门主管发起部门员工。
  ② 受发起的个体注册凭证，该过程一般包括证明身份、采集个人经历与生物特征数据。这个步骤可能还涉及到合并由身份管理组件维护的权威属性数据。
  ③ 凭证生成。根据凭证类型，生成过程可能涉及到加密、使用数字签名、生成智能卡及其他功能。
  ④ 凭证颁发给个体或NPE。
  ⑤ 最后，凭证必须在其生命周期内得到维护，可能涉及撤销、补发/替换、重新注册、到期、个人标识号（PIN）重置、挂起或者恢复。

4.7.3 访问管理

访问管理组件对实体被授权访问资源的方法进行管理和控制。

包括逻辑上和物理上的访问

可以在系统内部，也可是外部单元。

访问管理的目的是，确保当个体试图访问安全敏感的建筑物、计算机系统或数据时，进行适当的身份验证。

访问控制功能单元利用请求访问者提交的凭证及其数字身份。

企业级的访问控制设施需要以下三个支持要素：

- 资源管理
- 特权管理
- 策略管理

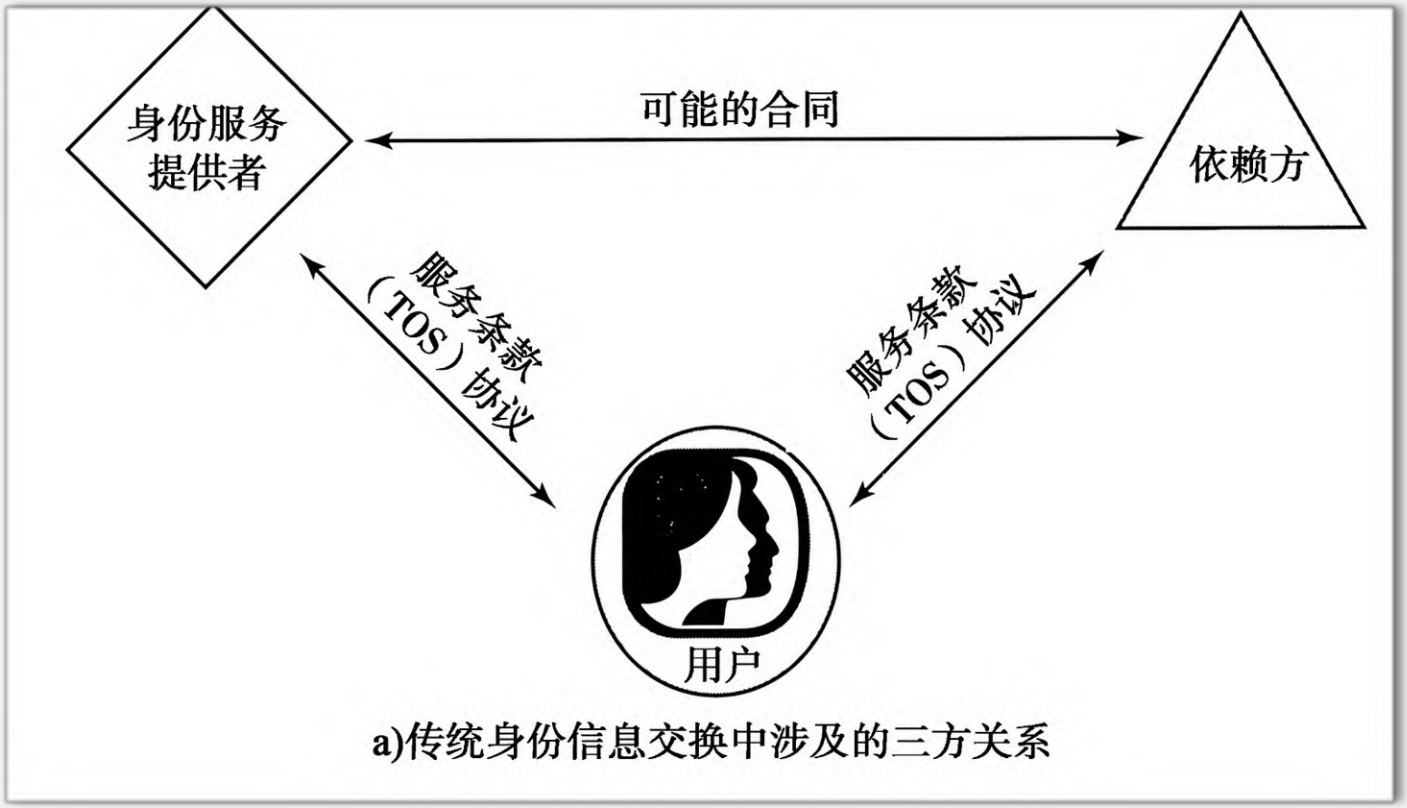
4.7.4 身份联合

- “身份联合”这个术语用来描述允许一个组织信任由另一个组织创建和发布的数字身份、身份属性与凭证的技术、标准、策略和过程。
- 身份联合解决两个问题：
  ① 你如何信任需要访问自己系统且来自外部组织的个体的身份？
  ② 当贵组织中的个体需要与外部组织合作时，你如何保证他们的身份？

4.8 信任框架

- 信任、身份、属性等相互联系的概念已经成为互联网企业、网络服务提供者和大型企业关注的核心问题。
- 这可以从电子商务建立过程明显地看出来，为了提高效率、保护隐私并简化法律程序，交易各方一般采用“须知”准则：为了和某人进行交易，你需要知道他哪些信息？
- 个体的属性必须被知晓并且验证后才能允许进行交易，而这些属性依赖于情境。

4.8.1 传统的身份交换方法



a)传统身份信息交换中涉及的三方关系

上图描述了传统的身份信息交换技术。该技术包括，用户开发与身份服务提供者（identity service provider）采购数字身份和凭证的协议，与最终用户服务与应用提供者以及愿意依赖身份服务提供者生成的身份与凭证信息的各方签订协议。

4.8.2 开放的身份信任框架

OpenID

- 这是一个开放性标准，允许用户通过合作网站（称为依赖方）提供的第三方服务实现身份认证，从而不需网站管理员自己提供相关服务，且允许用户整合自己的数字身份。用户可以选择自己喜欢的OpenID身份提供者创建账户，然后使用这些账户登录任何接受OpenID认证的网站。

OIDF

- OpenID基金会（OpenID Foundation）是由承诺使用、推广和保护OpenID技术的个人和公司组成的非营利国际组织。OIDF提供OpenID所需的基础设施，并且帮助促进和支持OpenID的推广使用。

ICF

- 信息卡基金会（Information Card Foundation）是由致力于共同发展信息卡生态系统的公司和个人组成的非营利团体。信息卡是人们在网上使用的个人数字身份，是身份元系统的重要组成部分。每张信息卡看起来都具有卡片形状的图片和与之关联的卡名，使得人们能够组织自己的所有数字身份并且很便捷地选择某种交互环境中想用的数字身份。

OITF

- 开放身份信任框架（Open Identity Trust Framework）是一个标准化的、开放性的用于身份和属性交换的信任框架规范，由OIDF和ICF联合开发。

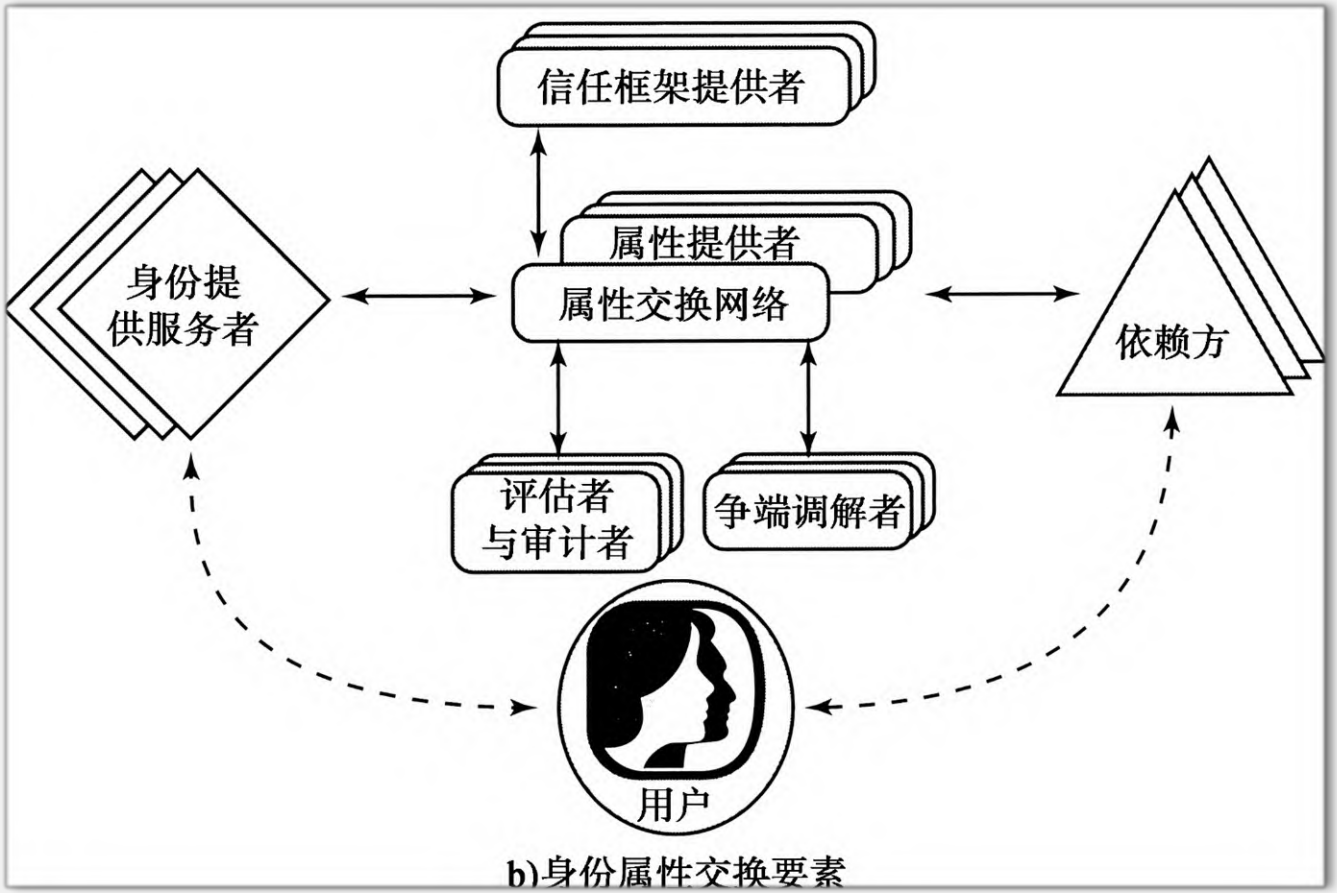
OIX

- 开放身份交换公司（Open Identity Exchange Corporation）是一个独立且中立的遵循开放身份信任框架模型的国际证书信任框架提供者。

AXN

- 属性交换网络（Attribute Exchange Network）是一个因特网范围的在线网关，使身份服务提供者和依赖方能够以可负担的成本实现高效访问用户声称、许可且已验证的大量在线身份属性。

4.8.2 开放的身份信任框架



上图描述了OITF所包含的要素。箭头实线表示与信任框架提供者之间关于实现技术、运行和法律要求达成的协议。箭头虚线表示受这些需求潜在影响的其他协议。

4.9 案例学习：银行的RBAC系统

- Dresdner银行实现了一个RBAC系统，可以作为有用的实际例子[SCHA01]。
- 银行使用多种计算机应用。其中很多应用最初是为大型机环境开发的，这些老应用中有一些现在支持客户-服务器网络，其它的保留在大型机上。服务器上也有新应用。
- 1990年以前，在每台服务器和大型机上采用简单的DAC系统。管理员在每台主机上维护本地访问控制文件，为每台主机上的每个应用的每个用户定义访问权。系统非常繁琐，非常耗时，易于出错。为了改进系统，银行引进了RBAC方案。RBAC是系统级的，它将访问权的确定划分为三个不同的管理单元，以获得更高的安全性。

4.9 案例学习：银行的RBAC系统

表 4-4　银行业实例的职责与角色

a）职责与职位

角　色	职　责	职　位	角　色	职　责	职　位
A	金融分析师	职员	G	金融分析师	助理
B	金融分析师	团队经理	…	…	…
C	金融分析师	部门总监	X	股票技师	职员
D	金融分析师	低级	Y	电子商务支持	低级
E	金融分析师	高级	Z	银行业务	部门总监
F	金融分析师	专家			

b）许可分配

角　色	应　用	访　问　权
A	货币市场工具	1, 2, 3, 4
	衍生贸易	1, 2, 3, 7, 10, 12
	利息工具	1, 4, 8, 12, 14, 16
B	货币市场工具	1, 2, 3, 4, 7
	衍生贸易	1, 2, 3, 7, 10, 12, 14
	利息工具	1, 4, 8, 12, 14, 16
	私人消费者工具	1, 2, 4, 7
…	…	…

c）具有继承的许可分配

角　色	应　用	访　问　权
A	货币市场工具	1, 2, 3, 4
	衍生贸易	1, 2, 3, 7, 10, 12
	利息工具	1, 4, 8, 12, 14, 16
B	货币市场工具	7
	衍生贸易	14
	私人消费者工具	1, 2, 4, 7
…	…	…

4.9 案例学习：银行的RBAC系统

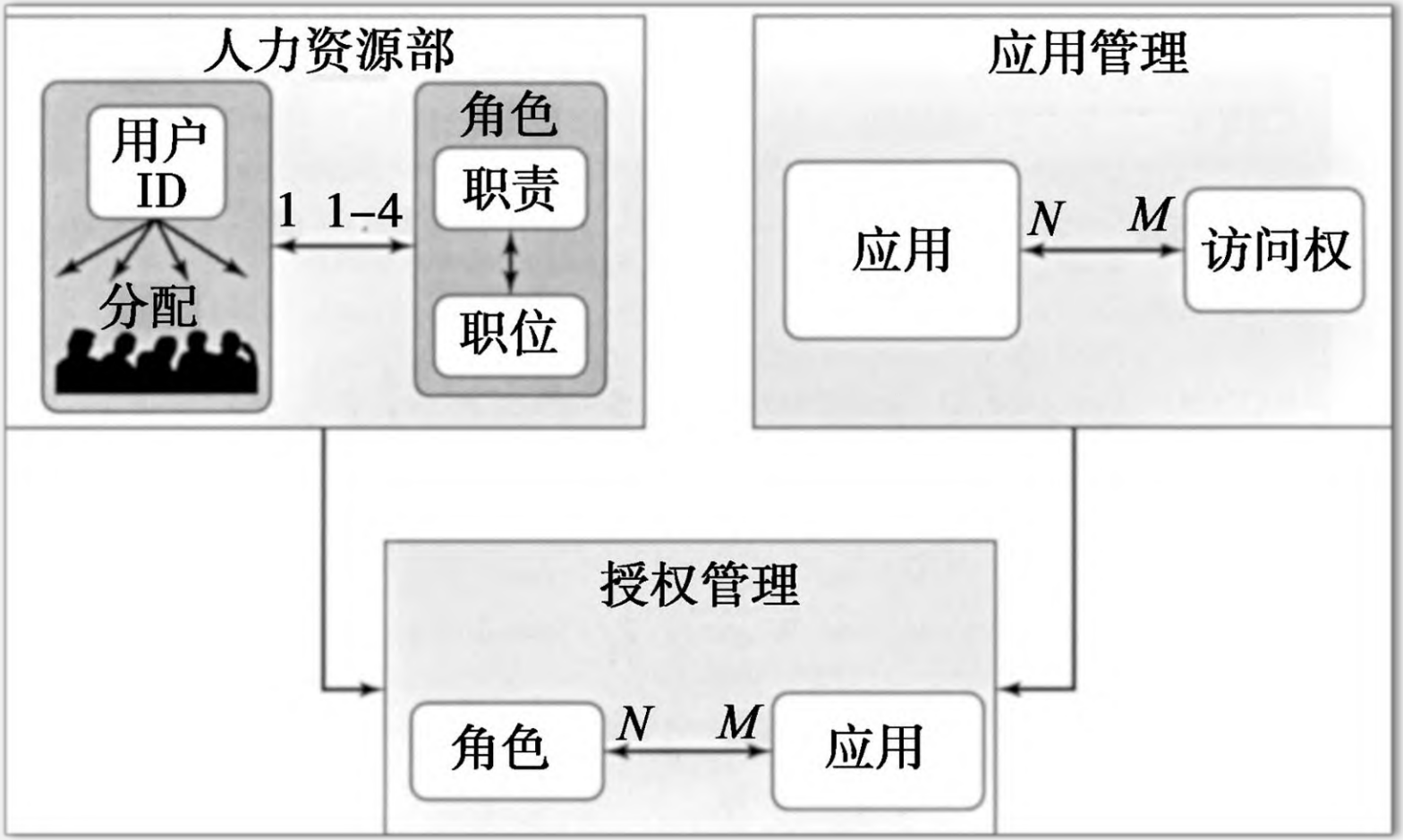


图4.14 访问控制管理实例