COMPUTER SECURITY

PRINCIPLES AND PRACTICE

SECOND EDITION

William Stallings | Lawrie Brown

Chapter 4

Access Control

Access Control

ITU-T Recommendation X.800 defines access control as follows:

"The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner."

Access Control Principles

RFC 2828 defines computer security as:

"Measures that implement and assure security services in a computer system, particularly those that assure access control service".

Relationship Among Access Control and Other Security Functions

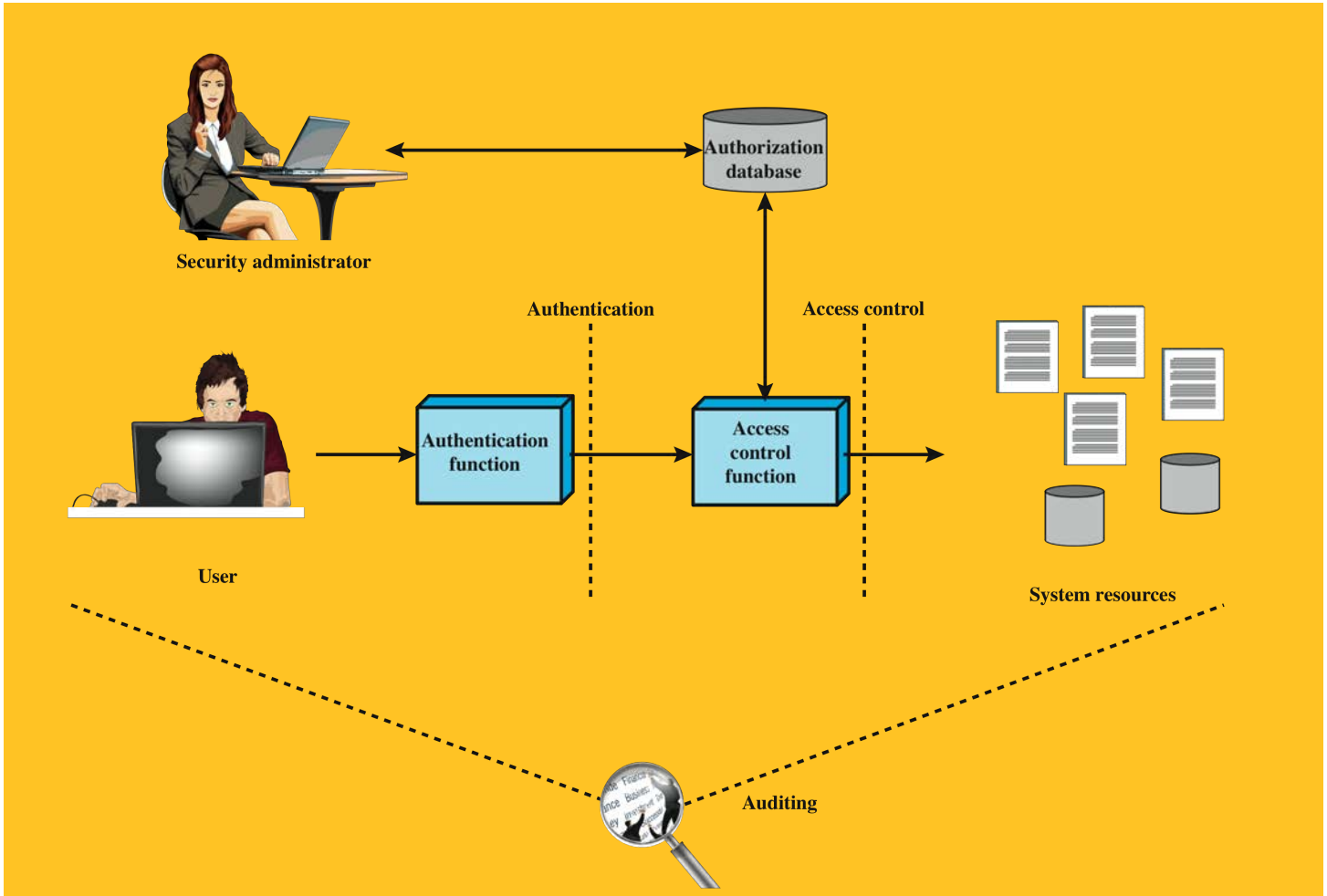


Figure 4.1 Relationship Among Access Control and Other Security Functions

Access Control Policies

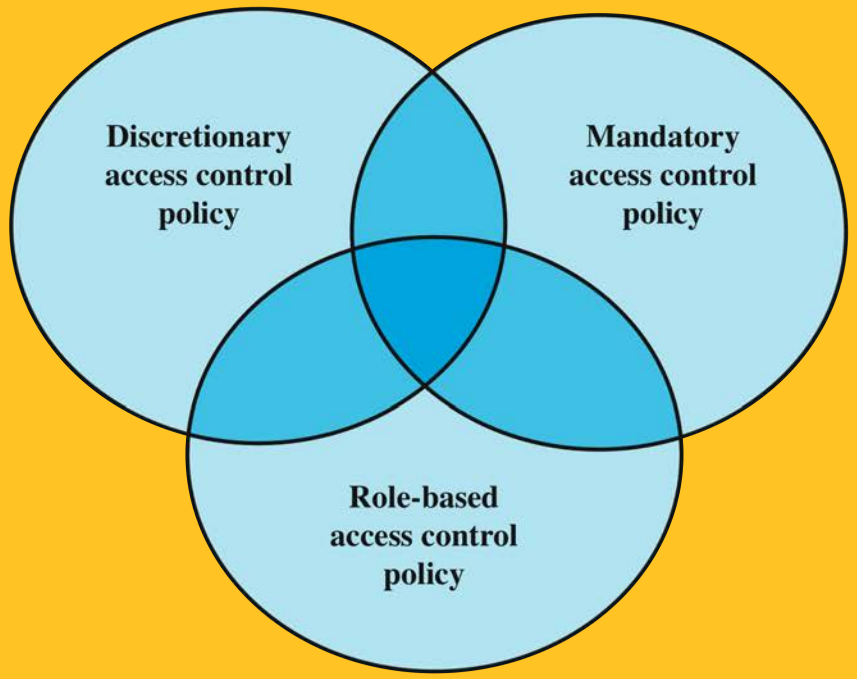
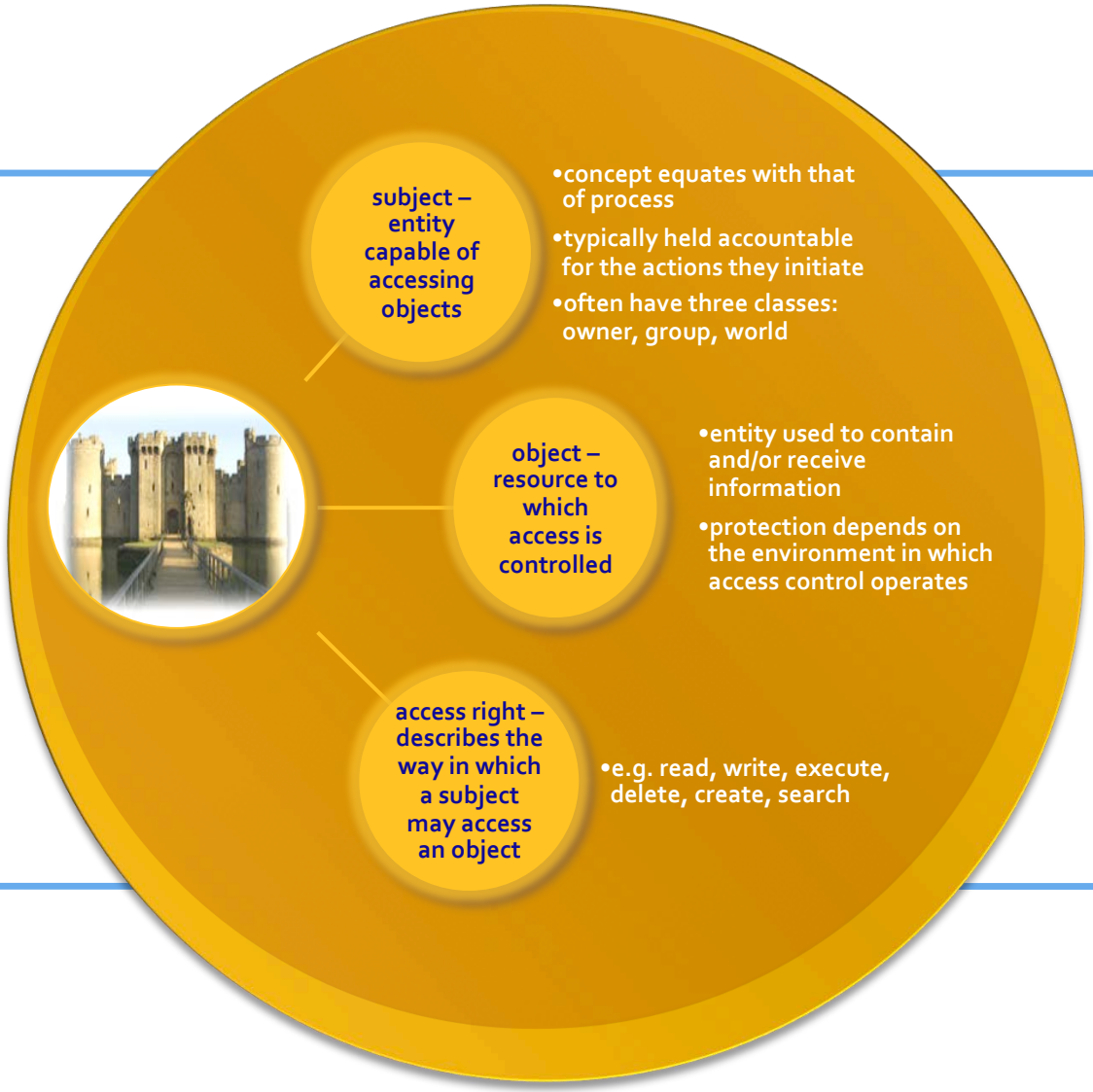


Figure 4.2 Multiple Access Control Policies. DAC, MAC, and RBAC are not mutually exclusive. A system may implement two or even three of these policies for some or all types of access. [SAND94]

Access Control Requirements

- reliable input
- support for fine and coarse specifications
- least privilege
- separation of duty
- open and closed policies
- policy combinations and conflict resolution
- administrative policies
- dual control

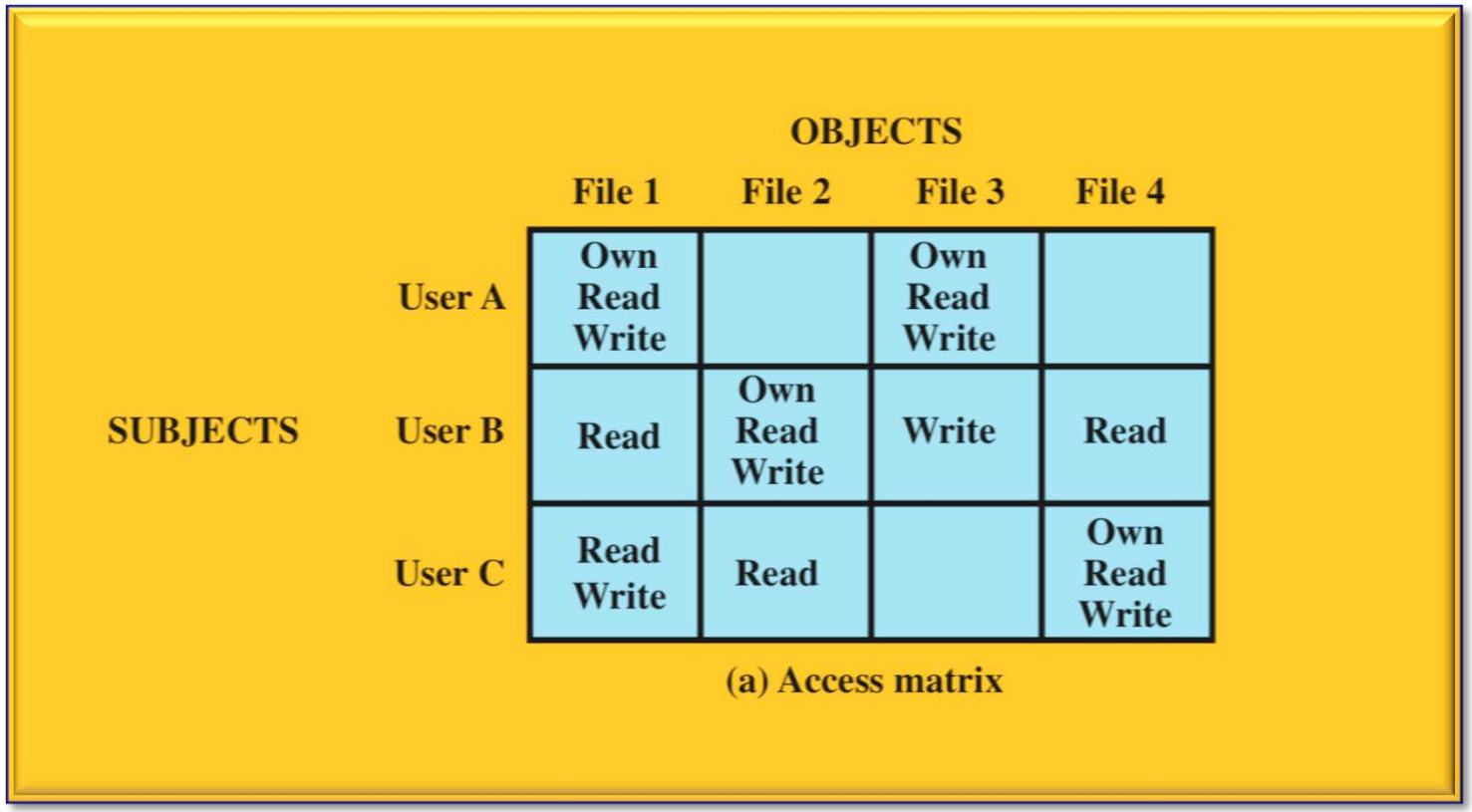
Access Control Basic Elements



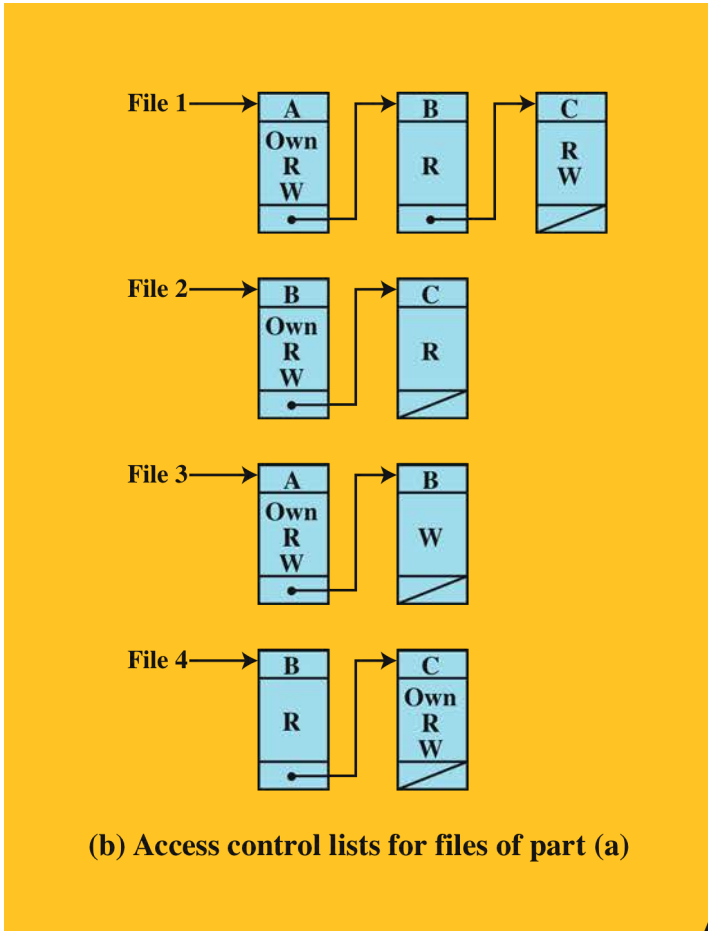
Discretionary Access Control (DAC)

- scheme in which an entity may enable another entity to access some resource
- often provided using an access matrix
  - one dimension consists of identified subjects that may attempt data access to the resources
  - the other dimension lists the objects that may be accessed
- each entry in the matrix indicates the access rights of a particular subject for a particular object

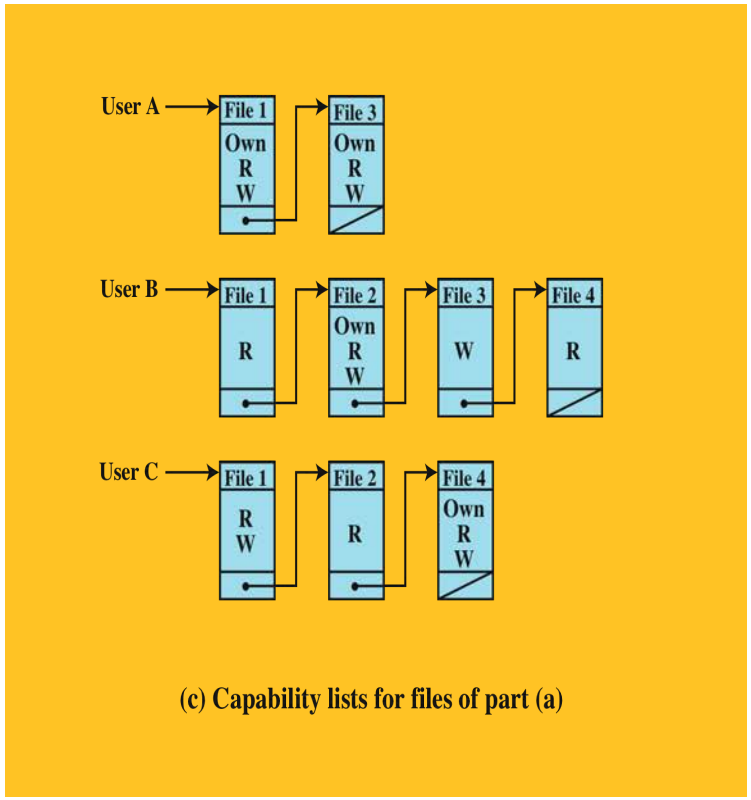
Figure 4.3a
Access Matrix



Figures 4.3b and c
Example of Access Control Structures



(b) Access control lists for files of part (a)



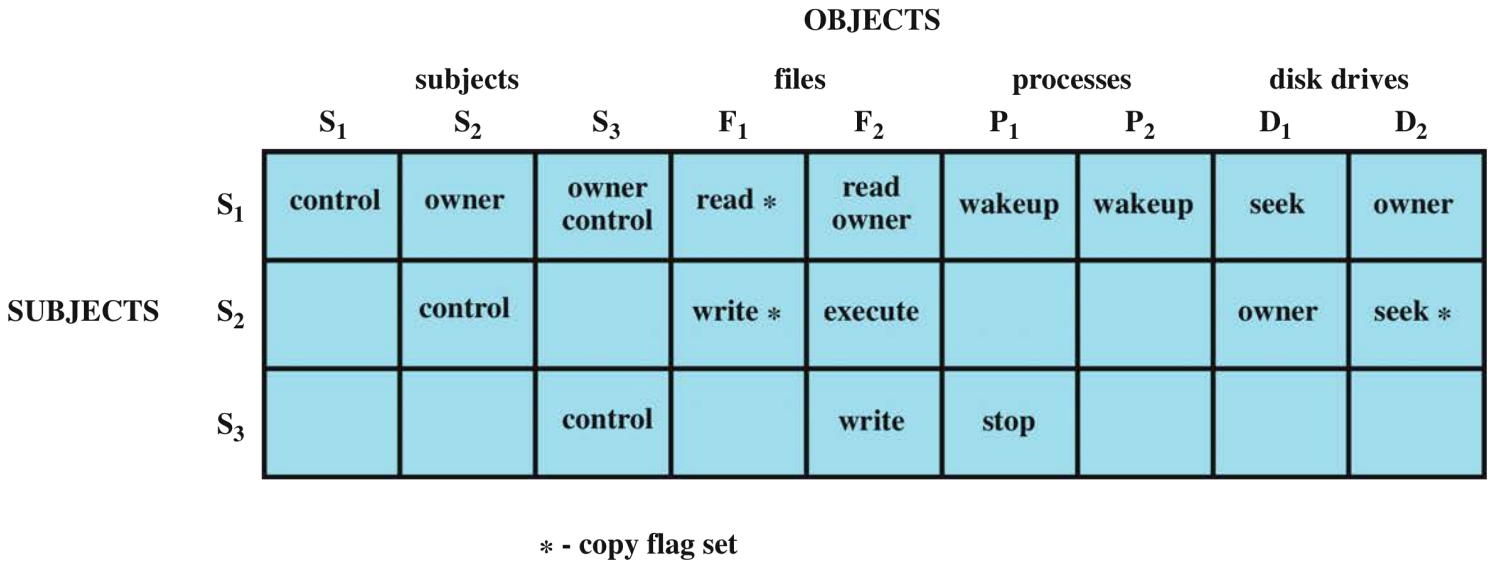
(c) Capability lists for files of part (a)

Table 4.1

Authorization Table for Files in Figure 4.3

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

Figure 4.4
Extended Access Control Matrix



		OBJECTS								
		subjects			files		processes		disk drives	
		S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
SUBJECTS	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			

* - copy flag set

Figure 4.4 Extended Access Control Matrix

Figure 4.5

Access Control Function

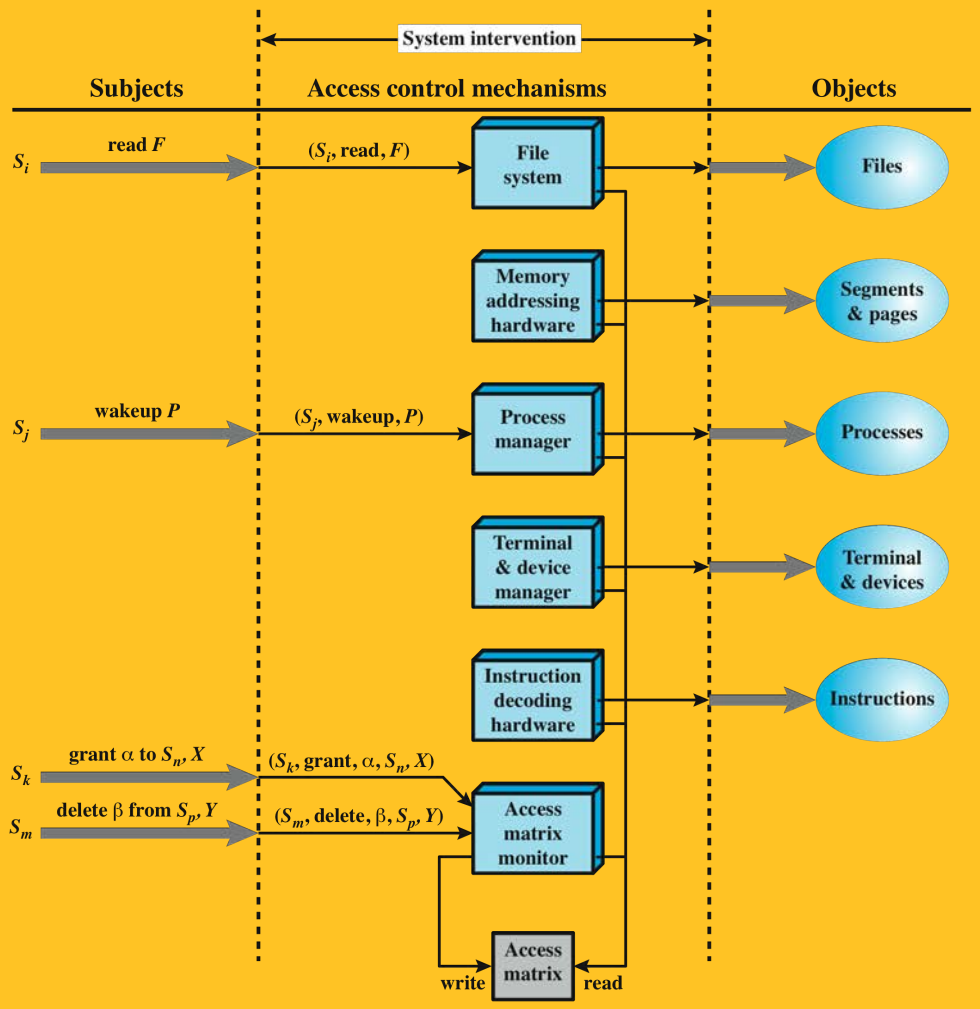
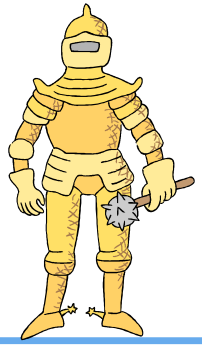


Figure 4.5 An Organization of the Access Control Function

Rule	Command (by S_0)	Authorization	Operation
R1	**transfer** $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ **to** S, X	'α*' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R2	**grant** $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ **to** S, X	'owner' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R3	**delete** α **from** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	delete α from $A[S, X]$
R4	$w \leftarrow$ **read** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	copy $A[S, X]$ into w
R5	**create object** X	None	add column for X to A; store 'owner' in $A[S_0, X]$
R6	**destroy object** X	'owner' in $A[S_0, X]$	delete column for X from A
R7	**create subject** S	none	add row for S to A; execute **create object** S; store 'control' in $A[S, S]$
R8	**destroy subject** S	'owner' in $A[S_0, S]$	delete row for S from A; execute **destroy object** S

Table 4.2

Access Control System Commands



Protection Domains

- set of objects together with access rights to those objects
- more flexibility when associating capabilities with protection domains
- in terms of the access matrix, a row defines a protection domain
- user can spawn processes with a subset of the access rights of the user
- association between a process and a domain can be static or dynamic
- in user mode certain areas of memory are protected from use and certain instructions may not be executed
- in kernel mode privileged instructions may be executed and protected areas of memory may be accessed

UNIX File Access Control

UNIX files are administered using inodes (index nodes)

- control structures with key information needed for a particular file
- several file names may be associated with a single inode
- an active inode is associated with exactly one file
- file attributes, permissions and control information are sorted in the inode
- on the disk there is an inode table, or inode list, that contains the inodes of all the files in the file system
- when a file is opened its inode is brought into main memory and stored in a memory resident inode table

directories are structured in a hierarchical tree

- may contain files and/or other directories
- contains file names plus pointers to associated inodes

UNIX
File Access Control

- unique user identification number (user ID)
- member of a primary group identified by a group ID
- belongs to a specific group
- 12 protection bits
  - specify read, write, and execute permission for the owner of the file, members of the group and all other users
- the owner ID, group ID, and protection bits are part of the file's inode

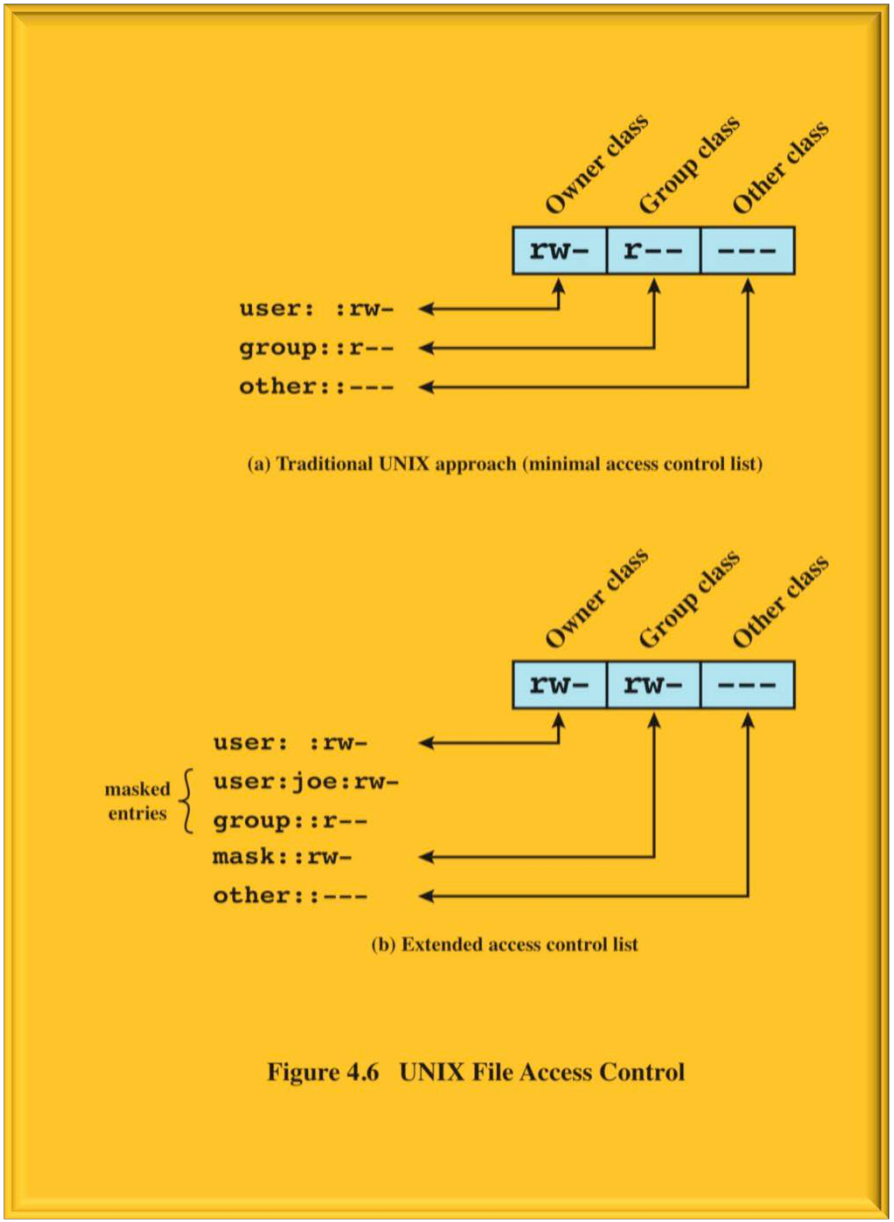


Figure 4.6 UNIX File Access Control

Traditional UNIX File Access Control

- "set user ID"(SetUID)
- "set group ID"(SetGID)
  - system temporarily uses rights of the file owner / group in addition to the real user's rights when making access control decisions
  - enables privileged programs to access files / resources not generally accessible
- sticky bit
  - when applied to a directory it specifies that only the owner of any file in the directory can rename, move, or delete that file
- superuser
  - is exempt from usual access control restrictions
  - has system-wide access

Access Control Lists (ACLs) in UNIX

- modern UNIX systems support ACLs
  - FreeBSD, OpenBSD, Linux, Solaris
- FreeBSD
  - Setfacl command assigns a list of UNIX user IDs and groups
  - any number of users and groups can be associated with a file
  - read, write, execute protection bits
  - a file does not need to have an ACL
  - includes an additional protection bit that indicates whether the file has an extended ACL
- when a process requests access to a file system object two steps are performed:
  - step 1 selects the most appropriate ACL
    - owner, named users, owning / named groups, others
  - step 2 checks if the matching entry contains sufficient permissions

Figure 4.7

Role-Based Access Control (RBAC)

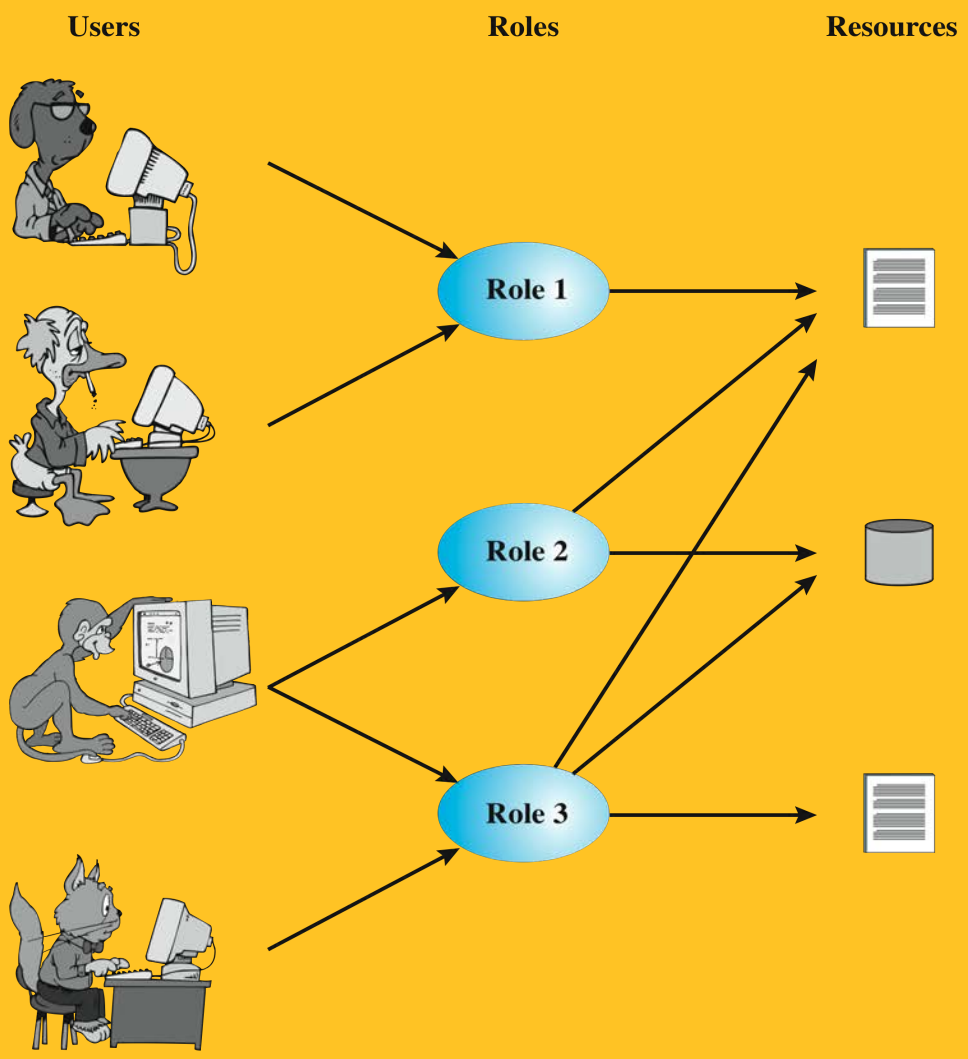
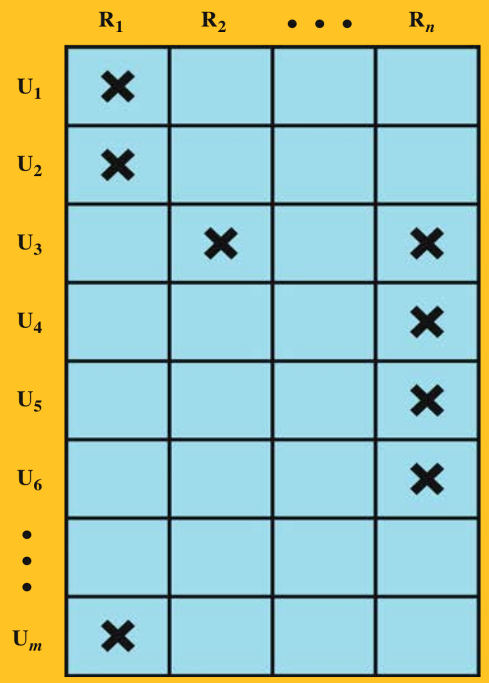


Figure 4.7 Users, Roles, and Resources



OBJECTS

ROLES	R_1	R_2	R_n	F_1	F_1	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			

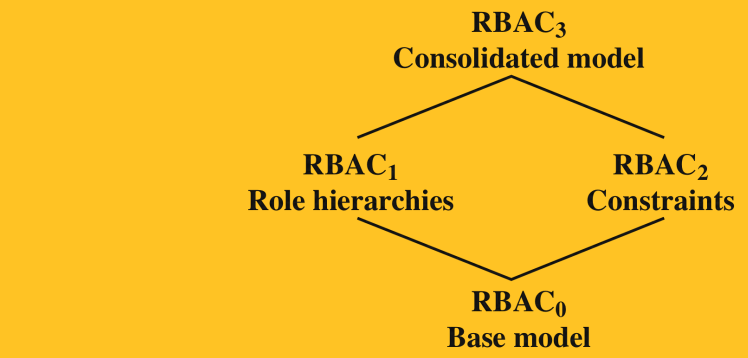
Figure 4.8 Access Control Matrix Representation of RBAC

Figure 4.8

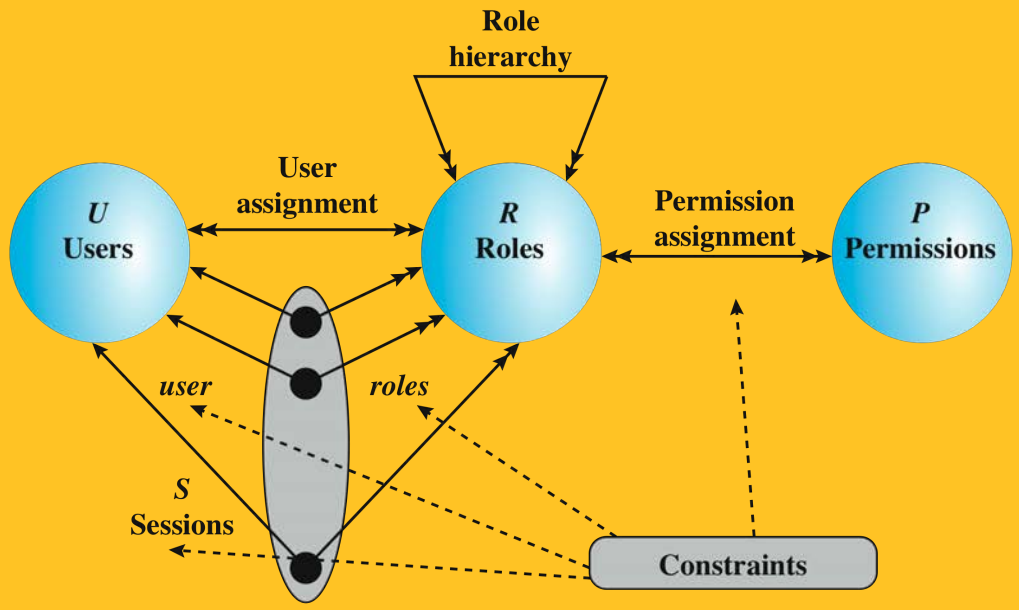
Access Control Matrix

Figure 4.9

Role-Based Access Control Models



(a) Relationship among RBAC models



(b) RBAC models

Figure 4.9 A Family of Role-Based Access Control Models. $RBAC_0$ is the minimum requirement for an RBAC system. RBAC1 adds role hierarchies and $RBAC_2$ adds constraints. RBAC3 includes $RBAC_1$ and $RBAC_2$. [SAND96]

Table 4.3
Scope RBAC Models

Models	Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

Example of Role Hierarchy

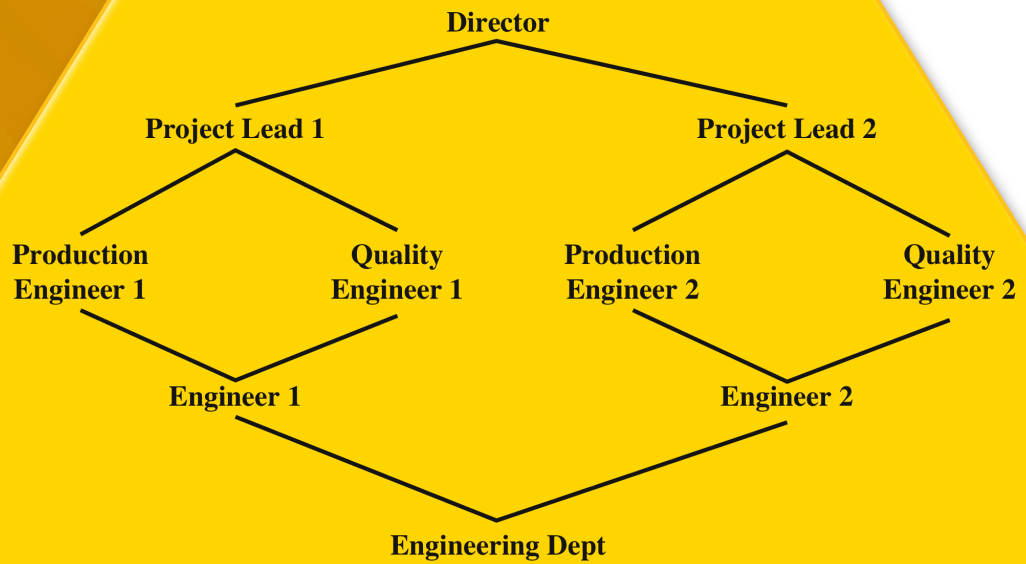


Figure 4.10 Example of Role Hierarchy

Constraints - RBAC

- **provide a means of adapting RBAC to the specifics of administrative and security policies of an organization**
- **a defined relationship among roles or a condition related to roles**
- **types:**

mutually exclusive roles

- a user can only be assigned to one role in the set (either during a session or statically)
- any permission (access right) can be granted to only one role in the set

cardinality

- setting a maximum number with respect to roles

prerequisite roles

- dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role

RBAC System and Administrative Functional Specification

administrative functions

- provide the capability to create, delete, and maintain RBAC elements and relations

supporting system functions

- provide functions for session management and for making access control decisions

review functions

- provide the capability to perform query operations on RBAC elements and relations

Figure 4.11
NIST RBAC Model

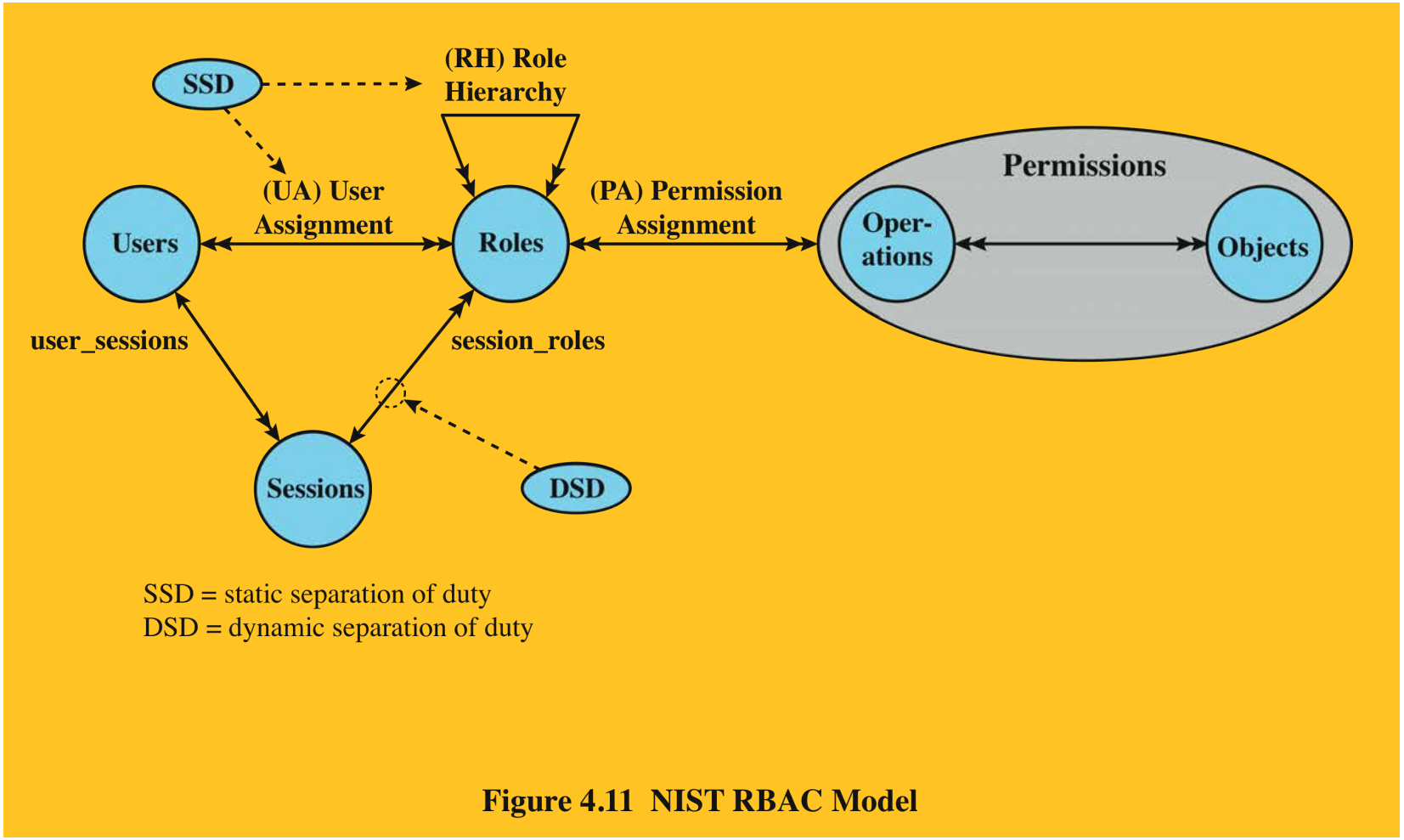


Figure 4.11 NIST RBAC Model

Basic Definitions

- object
  - any system resource subject to access control, such as a file, printer, terminal, database record
- operation
  - an executable image of a program, which upon invocation executes some function for the user
- permission
  - an approval to perform an operation on one or more RBAC protected objects

Core RBAC

administrative functions

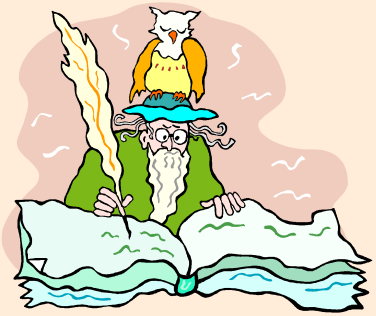
- add and delete users from the set of users
- add and delete roles from the set of roles
- create and delete instances of user-to-role assignment
- create and delete instances of permission-to-role assignment

supporting system functions

- create a user session with a default set of active roles
- add an active role to a session
- delete a role from a session
- check if the session subject has permission to perform a request operation on an object

review functions

- enable an administrator to view but not modify all the elements of the model and their relations

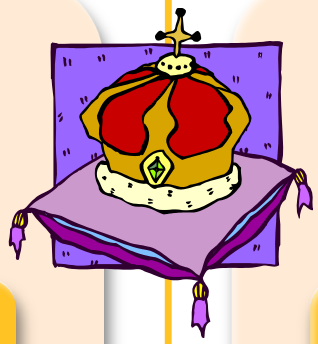


Hierarchical RBAC

general role hierarchies

allow an arbitrary partial ordering of the role hierarchy

supports multiple inheritance, in which a role may inherit permissions from multiple subordinate roles and more than one role can inherit from the same subordinate role



limited role hierarchies

impose restrictions resulting in a simpler tree structure

role may have one or more immediate ascendants but is restricted to a single immediate descendant

Static Separation of Duty Relations (SSD)

- enables the definition of a set of mutually exclusive roles, such that if a user is assigned to one role in the set, the user may not be assigned to any other role in the set
- can place a cardinality constraint on a set of roles
- defined as a pair (*role set*, *n*) where no user is assigned to *n* or more roles from the role set
- includes administrative functions for creating and deleting role sets and adding and deleting role members
- includes review functions for viewing the properties of existing SSD sets

Dynamic Separation of Duty Relations (DSD)

- limit the permissions available to a user
- places constraints on the roles that can be activated within or across a user's sessions
- define constraints as a pair *(role set, n)*, where *n* is a natural number $n \leq 2$, with the property that no user session may activate *n* or more roles from the role set
- enables the administrator to specify certain capabilities for a user at different, non-overlapping spans of time
- includes administrative and review functions for defining and viewing DSD relations

Functions and Roles for Banking Example

Table 4.4

(a) Functions and Official Positions

Role	Function	Official Position
A	financial analyst	Clerk
B	financial analyst	Group Manager
C	financial analyst	Head of Division
D	financial analyst	Junior
E	financial analyst	Senior
F	financial analyst	Specialist
G	financial analyst	Assistant
•••	•••	•••
X	share technician	Clerk
Y	support e-commerce	Junior
Z	office banking	Head of Division

Functions and Roles for Banking Example

Table 4.4
(b) Permission Assignments

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	1, 2, 3, 4, 7
	derivatives trading	1, 2, 3, 7, 10, 12, 14
	interest instruments	1, 4, 8, 12, 14, 16
	private consumer instruments	1, 2, 4, 7
• • •	• • •	• • •

Functions and Roles for Banking Example

Table 4.4
(c) PA with Inheritance

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	7
	derivatives trading	14
	private consumer instruments	1, 2, 4, 7
• • •	• • •	• • •

Figure 4.12
Example of Access Control Administration

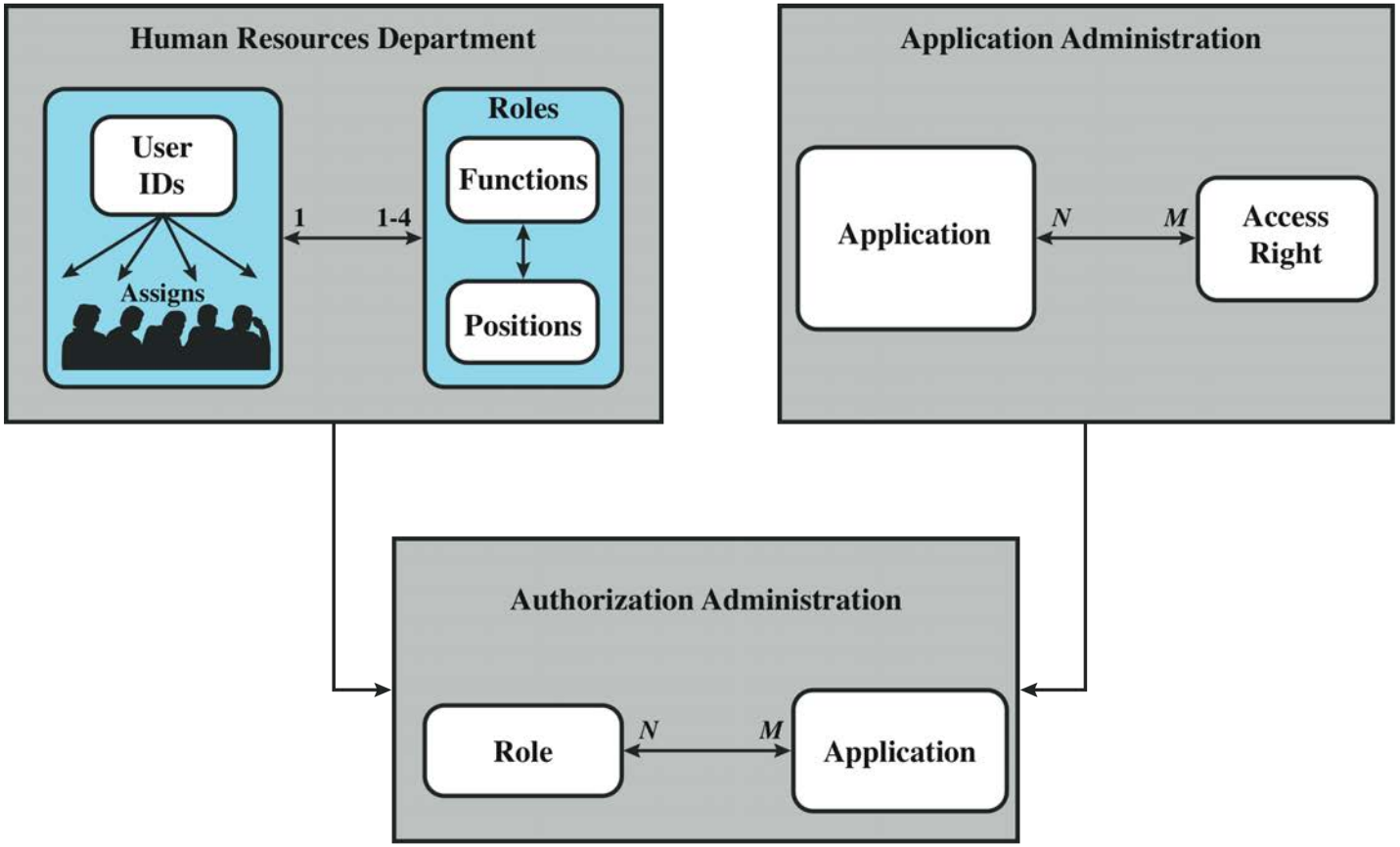


Figure 4.12 Example of Access Control Administration

Summary

- access control
  - prevent unauthorized users from gaining access to resources
  - prevent legitimate users from accessing resources in an unauthorized manner
  - enable legitimate users to access resources in an authorized manner
  - subjects, objects, access rights
  - authentication, authorization, audit
- discretionary access controls (DAC)
  - controls access based on identity
- mandatory access control (MAC)
  - controls access based on security labels
- role-based access control (RBAC)
  - controls access based on roles