

CS442 – Fall 2008

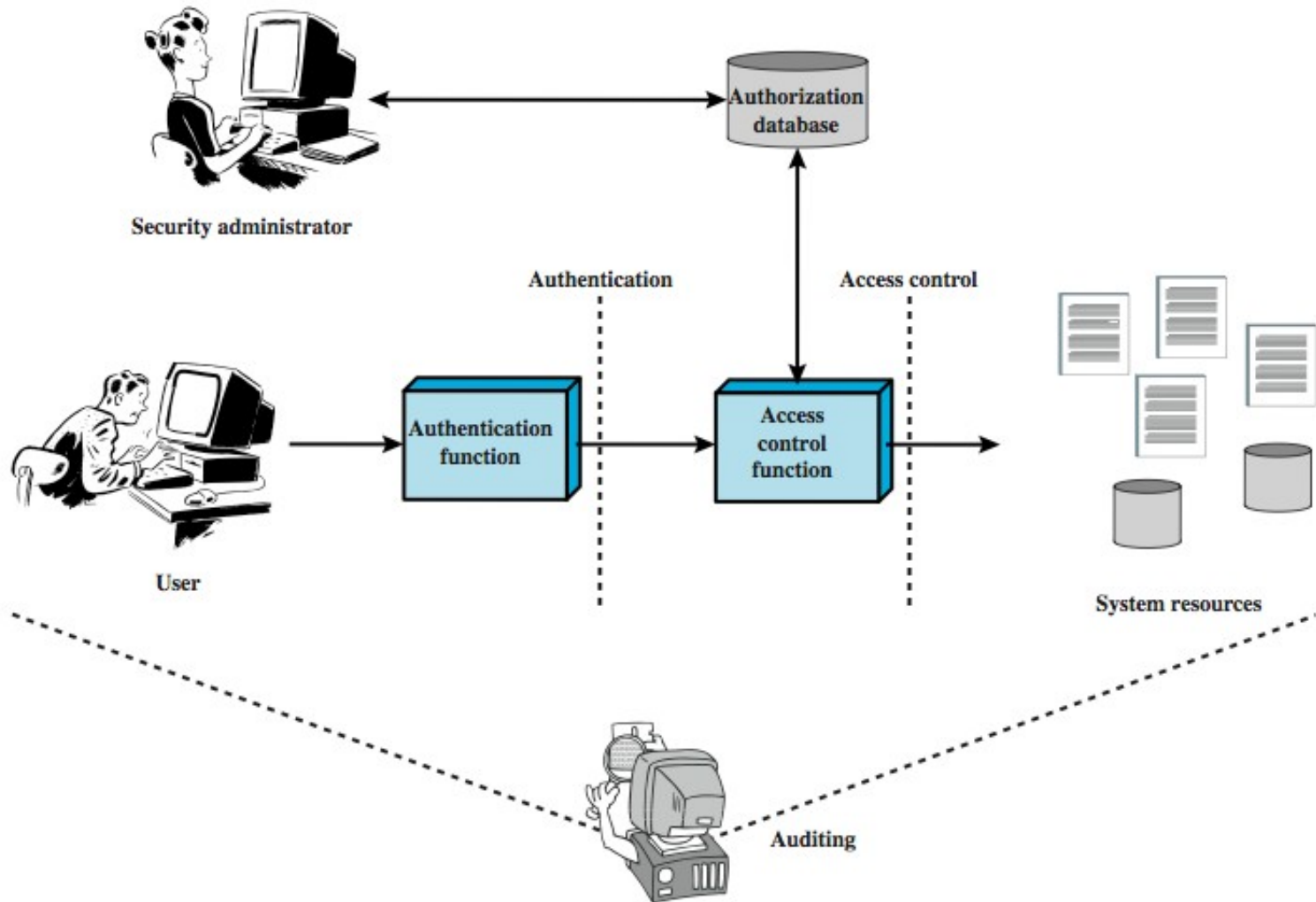
Intro. to Computer Security

Vinod Ganapathy
Lecture 9

Access Control

- “The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner“
- central element of computer security
- assume have users and groups
 - authenticate to system
 - assigned access rights to certain resources on system

Access Control Principles



Access Control Requirements

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

Access Control Elements

- subject - entity that can access objects
 - a process representing user/application
 - often have 3 classes: owner, group, world
- object - access controlled resource
 - e.g. files, directories, records, programs etc
 - number/type depend on environment
- access right - way in which subject accesses an object
 - e.g. read, write, execute, delete, create, search

Access Control: Overview

- Protection state of system
 - Describes current settings, values of system relevant to protection
- Access control matrix
 - Describes protection state precisely
 - Matrix describing rights of subjects
 - State transitions change elements of matrix

Discretionary Access Control

- often provided using an access matrix
 - lists subjects in one dimension (rows)
 - lists objects in the other dimension (columns)
 - each entry specifies access rights of the specified subject to that object

Description

objects (entities)

	o_1	...	o_m	s_1	...	s_n
s_1						
s_2						
...						
s_n						

subjects

- Subjects $S = \{ s_1, \dots, s_n \}$
- Objects $O = \{ o_1, \dots, o_m \}$
- Rights $R = \{ r_1, \dots, r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \dots, r_y \}$
means subject s_i has
rights r_x, \dots, r_y over
object o_j

Example 1

- Processes p, q
- Files f, g
- Rights r, w, x, a, o

$f \quad g \quad p \quad q$

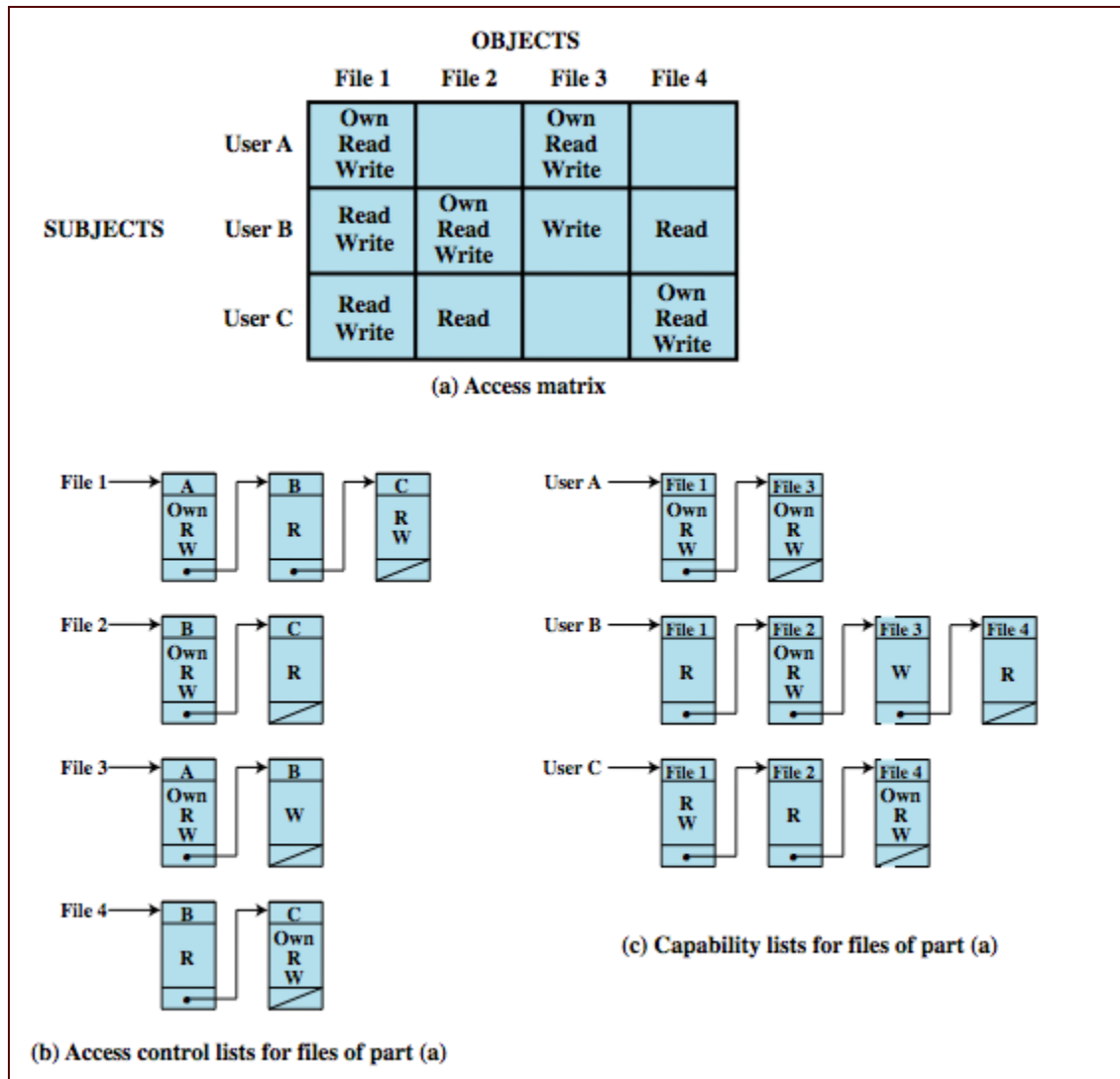
$p \quad rwo \quad r \quad rwxow$

$q \quad a \quad ro \quad r \quad rwxo$

Access control structures

- access matrix is often sparse
- can decompose by either row or column

Access Control Structures

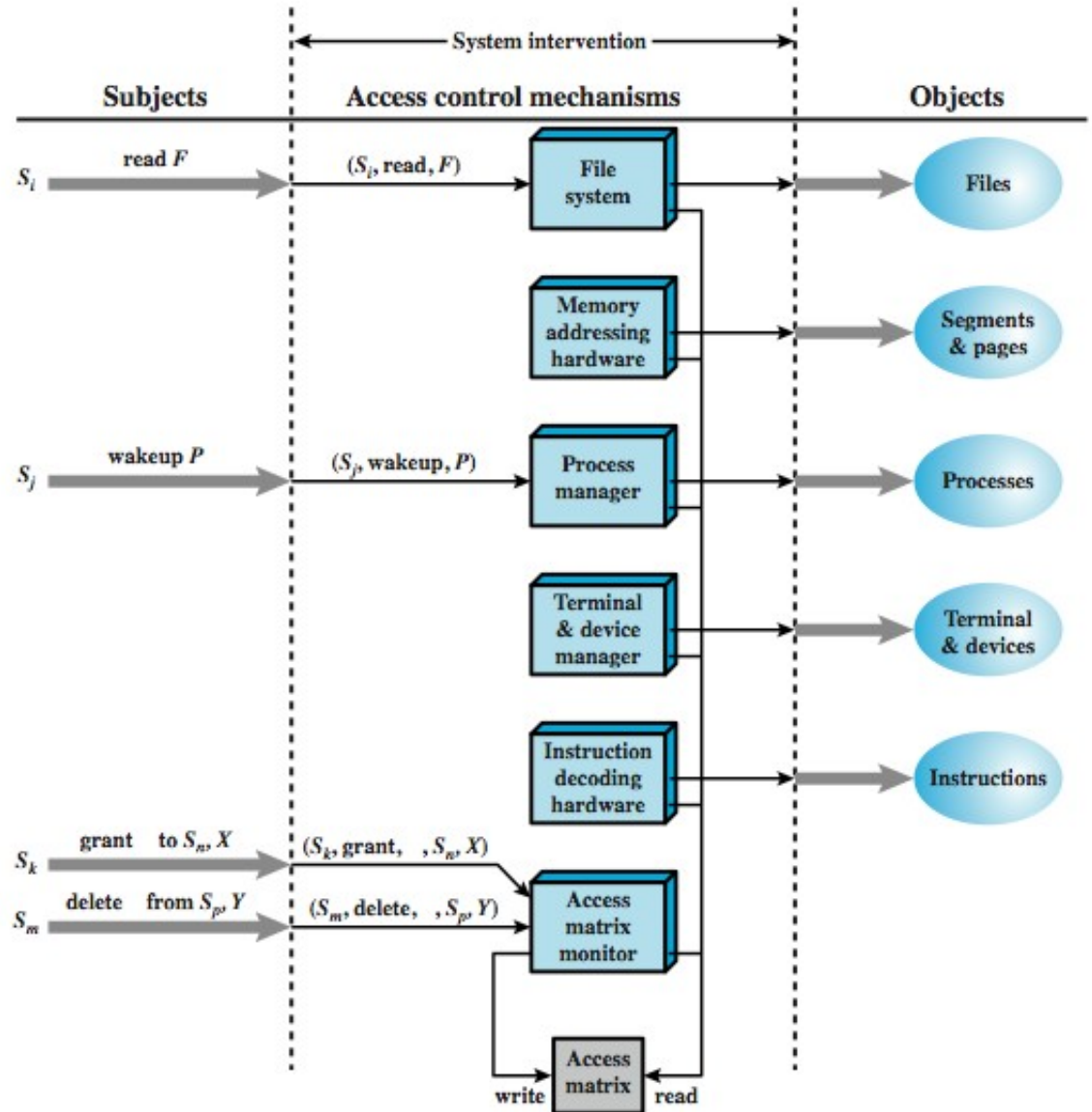


Access Control Model

		OBJECTS								
		subjects			files		processes		disk drives	
		S ₁	S ₂	S ₃	F ₁	F ₁	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

Access Control Function



Primitive Operations

- **create subject s ; create object o**
 - Creates new row, column in ACM; creates new column in ACM
- **destroy subject s ; destroy object o**
 - Deletes row, column from ACM; deletes column from ACM
- **enter r into $A[s, o]$**
 - Adds r rights for subject s over object o
- **delete r from $A[s, o]$**
 - Removes r rights from subject s over object o

Creating File

- Process p creates file f with r and w permission

```
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
```

Mono-Operational Commands

- Make process p the owner of file g
command *make.owner(p, g)*
enter *own* into $A[p, g]$;
end
- Mono-operational command
 - Single primitive operation in this command

Conditional Commands

- Let p give q r rights over f , if p owns f
command `grant.read.file.1(p, f, q)`
if `own in A[p, f]`
then
 enter r into `A[q, f]`;
end
- Mono-conditional command
 - Single condition in this command

Multiple Conditions

- Let p give q r and w rights over f , if p owns f and p has c rights over q

```
command grant.read.file.2( $p, f, q$ )  
  if  $own$  in  $A[p, f]$  and  $c$  in  $A[p, q]$   
  then  
    enter  $r$  into  $A[q, f]$ ;  
    enter  $w$  into  $A[q, f]$ ;  
end
```

Key Points

- Access control matrix simplest abstraction mechanism for representing protection state
- Transitions alter protection state
- Primitive operations alter matrix
 - Transitions can be expressed as commands composed of these operations and, possibly, conditions

What Is “Secure”?

- Adding a generic right r where there was not one is “leaking”
- If a system S , beginning in initial state s_0 , cannot leak right r , it is *safe with respect to the right r* .

Safety Question

- Does there exist an algorithm for determining whether a protection system S with initial state s_0 is safe with respect to a generic right r ?
 - Here, “safe” = “secure” for an abstract model
- Answer: **No**. Seminal result due to Harrison, Ruzzo and Ullman (1976).

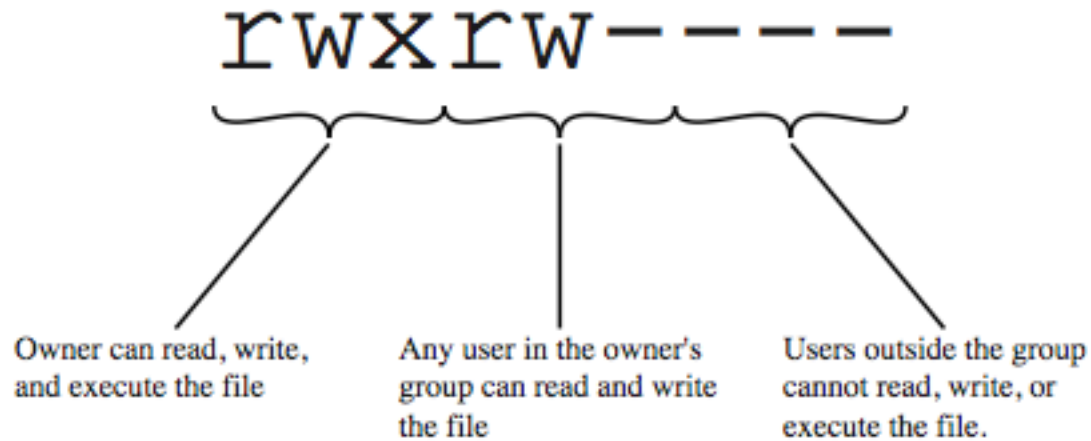
Protection Domains

- set of objects with associated access rights
- in access matrix view, each row defines a protection domain
 - but not necessarily just a user
 - may be a limited subset of user's rights
 - applied to a more restricted process
- may be static or dynamic

UNIX File Concepts

- UNIX files administered using inodes
 - control structure with key info on file
 - attributes, permissions of a single file
 - may have several names for same inode
 - have inode table / list for all files on a disk
 - copied to memory when disk mounted
- directories form a hierarchical tree
 - may contain files or other directories
 - are a file of names and inode numbers

UNIX File Access Control



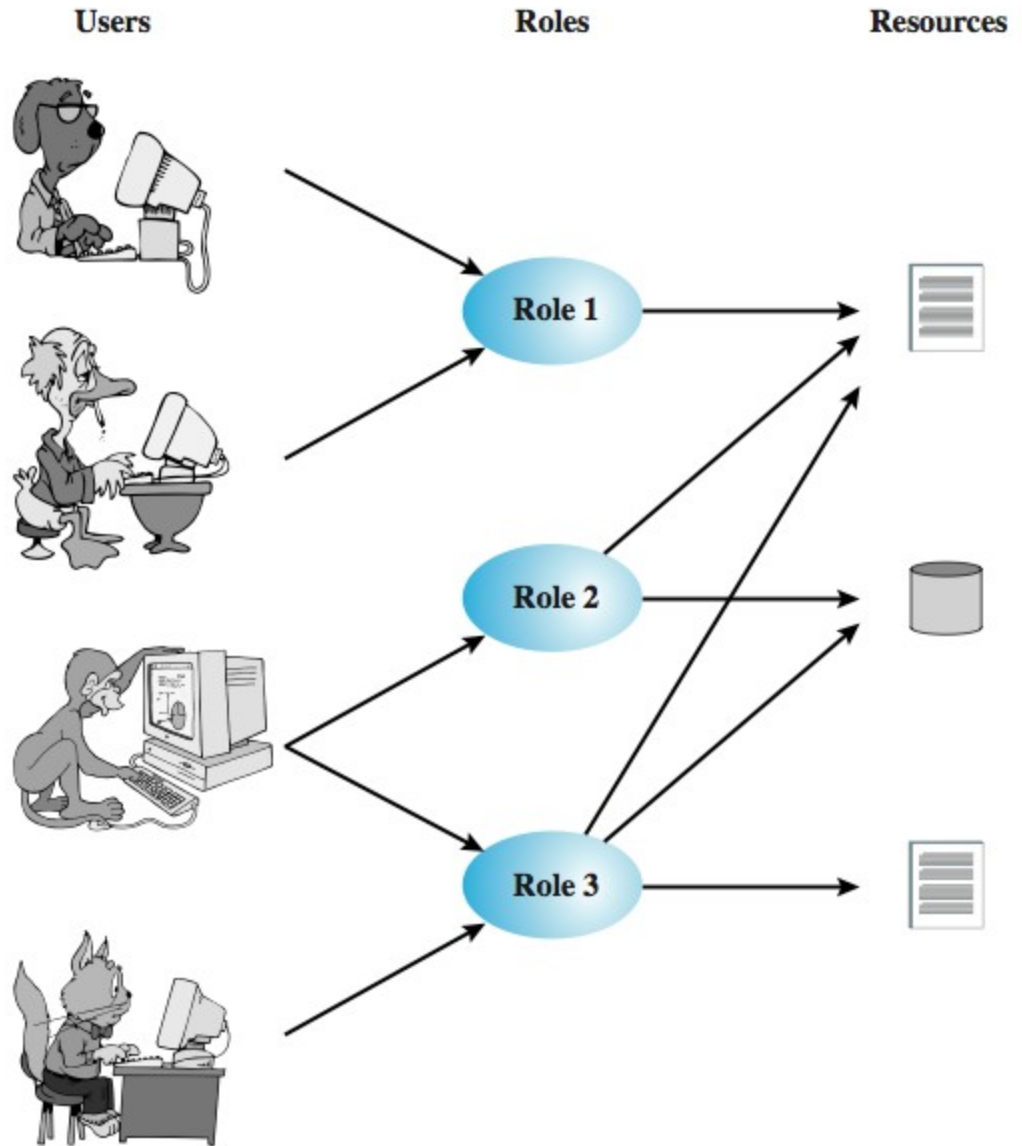
UNIX File Access Control

- “set user ID”(SetUID) or “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
 - on directory limits rename/move/delete to owner
- superuser
 - is exempt from usual access control restrictions

UNIX Access Control Lists

- modern UNIX systems support ACLs
- can specify any number of additional users / groups and associated rwx permissions
- ACLs are optional extensions to std perms
- group perms also set max ACL perms
- when access is required
 - select most appropriate ACL
 - owner, named users, owning / named groups, others
 - check if have sufficient permissions for access

Role-Based Access Control

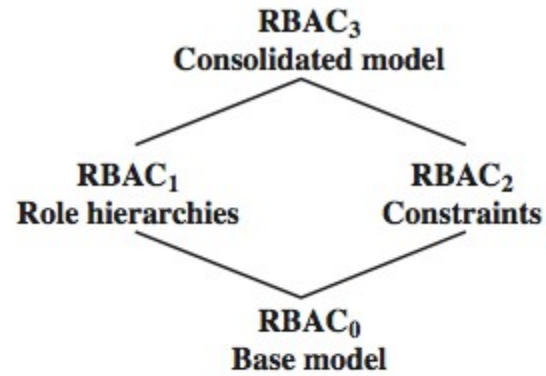


Role-Based Access Control

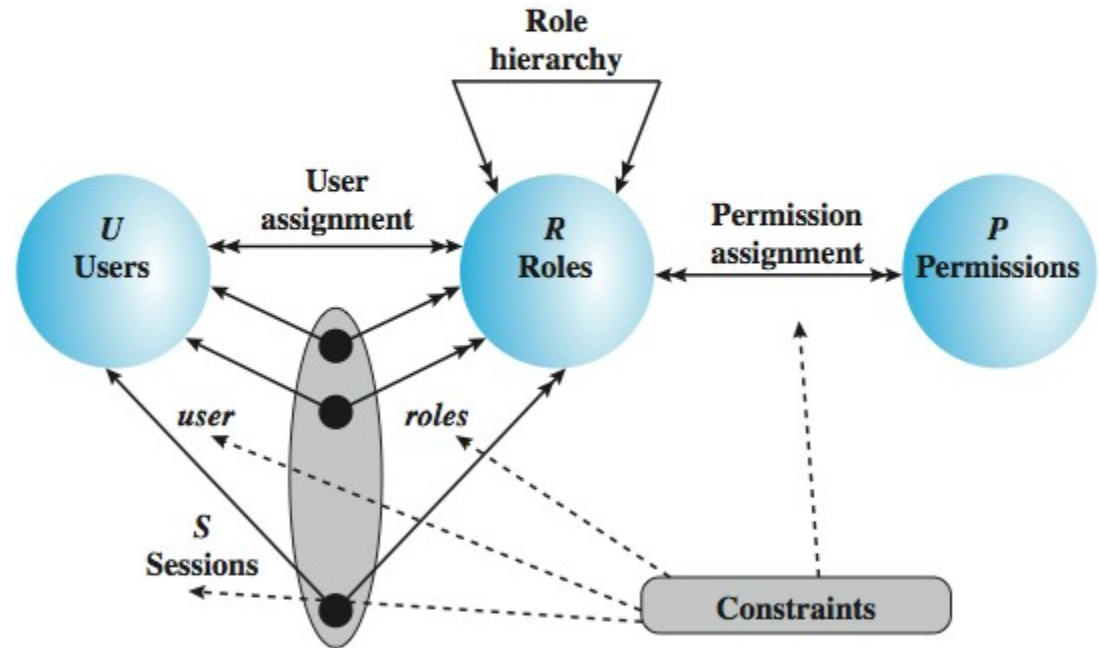
	R_1	R_2	...	R_n
U_1	×			
U_2	×			
U_3		×		×
U_4				×
U_5				×
U_6				×
•				
•				
U_m	×			

	OBJECTS								
	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			

Role- Based Access Control

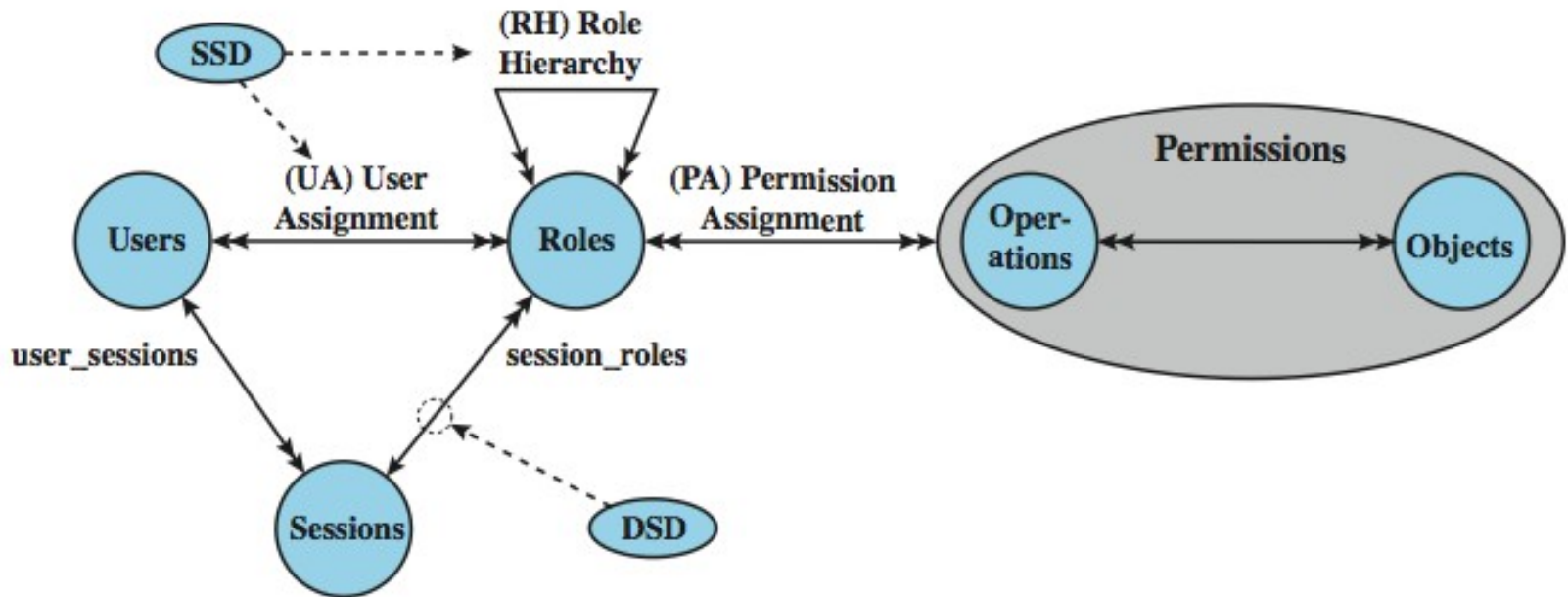


(a) Relationship among RBAC models



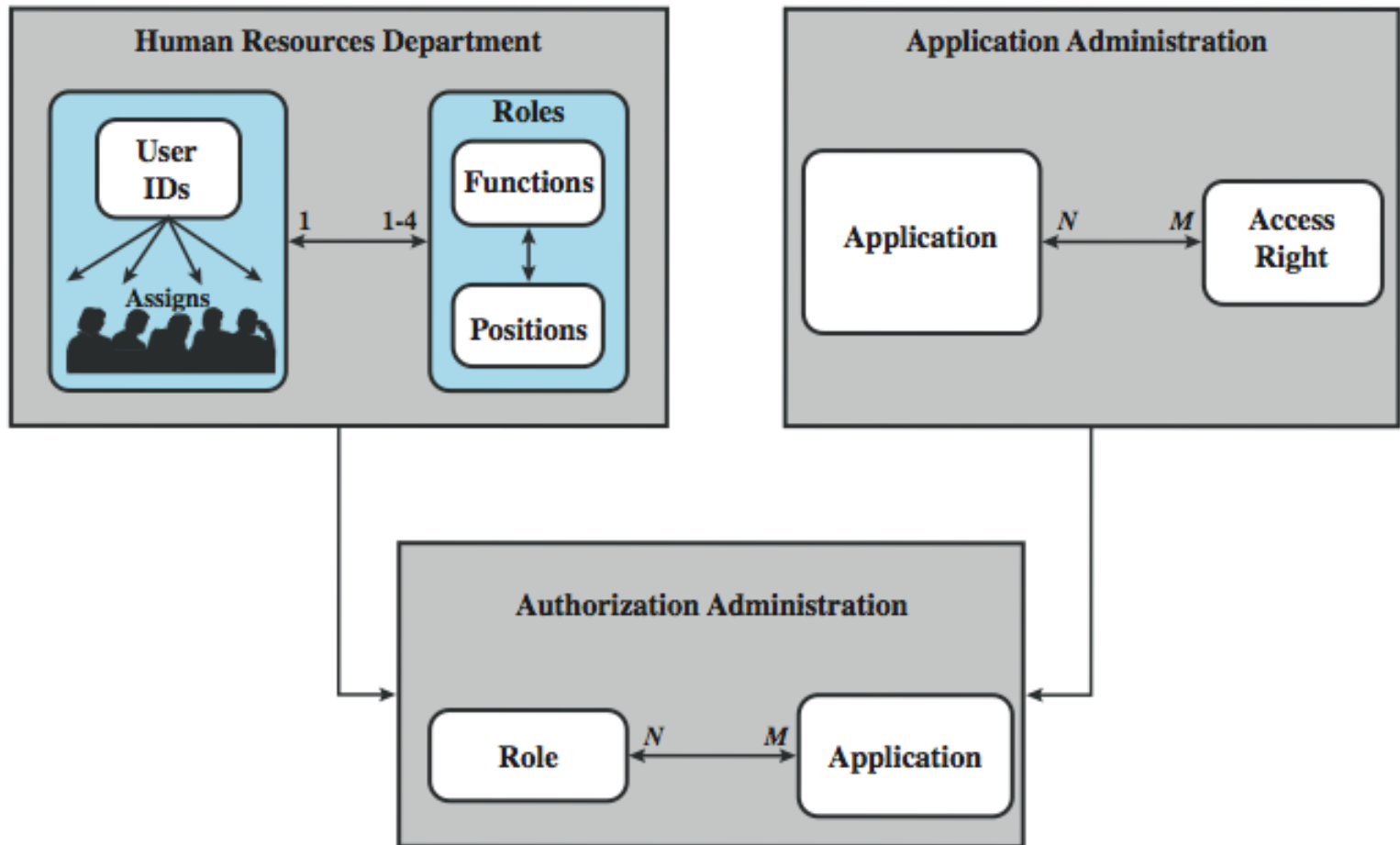
(b) RBAC models

NIST RBAC Model



SSD = static separation of duty
DSD = dynamic separation of duty

RBAC For a Bank



Security Policy

- Policy partitions system states into:
 - Authorized (secure)
 - These are states the system can enter
 - Unauthorized (nonsecure)
 - If the system enters any of these states, it's a security violation
- Secure system
 - Starts in authorized state
 - Never enters unauthorized state

Confidentiality

- X set of entities, I information
- I has *confidentiality* property with respect to X if no $x \in X$ can obtain information from I
- I can be disclosed to others
- Example:
 - X set of students
 - I final exam answer key
 - I is confidential with respect to X if students cannot obtain final exam answer key

Integrity

- X set of entities, I information
- I has *integrity* property with respect to X if all $x \in X$ trust information in I
- Types of integrity:
 - trust I , its conveyance and protection (data integrity)
 - I information about origin of something or an identity (origin integrity, authentication)
 - I resource: means resource functions as it should (assurance)

Availability

- X set of entities, I resource
- I has *availability* property with respect to X if all $x \in X$ can access I
- Types of availability:
 - traditional: x gets access or not
 - quality of service: promised a level of access (for example, a specific level of bandwidth) and not meet it, even though some access is achieved

Policy Models

- Abstract description of a policy or class of policies
- Focus on points of interest in policies
 - Security levels in multilevel security models
 - Separation of duty in Clark-Wilson model
 - Conflict of interest in Chinese Wall model

Types of Security Policies

- Military (governmental) security policy
 - Policy primarily protecting confidentiality
- Commercial security policy
 - Policy primarily protecting integrity
- Confidentiality policy
 - Policy protecting only confidentiality
- Integrity policy
 - Policy protecting only integrity

Integrity and Transactions

- Begin in consistent state
 - “Consistent” defined by specification
- Perform series of actions (*transaction*)
 - Actions cannot be interrupted
 - If actions complete, system in consistent state
 - If actions do not complete, system reverts to beginning (consistent) state