COMPSCI4062&5063: Cyber Security Fundamentals

Topic 3: Access Control

Overview

- Concept of Access Control
  - Definition
  - Access control in a broader context
  - Basic elements

访问控制的概念
- 定义
- 在更广泛的背景下进行访问控制
- 基本要素

- Access Control Policies
  - Discretionary access control (DAC)
  - Role-based access control (RBAM)
  - Attribute-based access control (ABAC)

访问控制政策
- 酌情访问控制(DAC)
- 基于角色的访问控制 (RBAM)
- 基于属性的访问控制(ABAC)

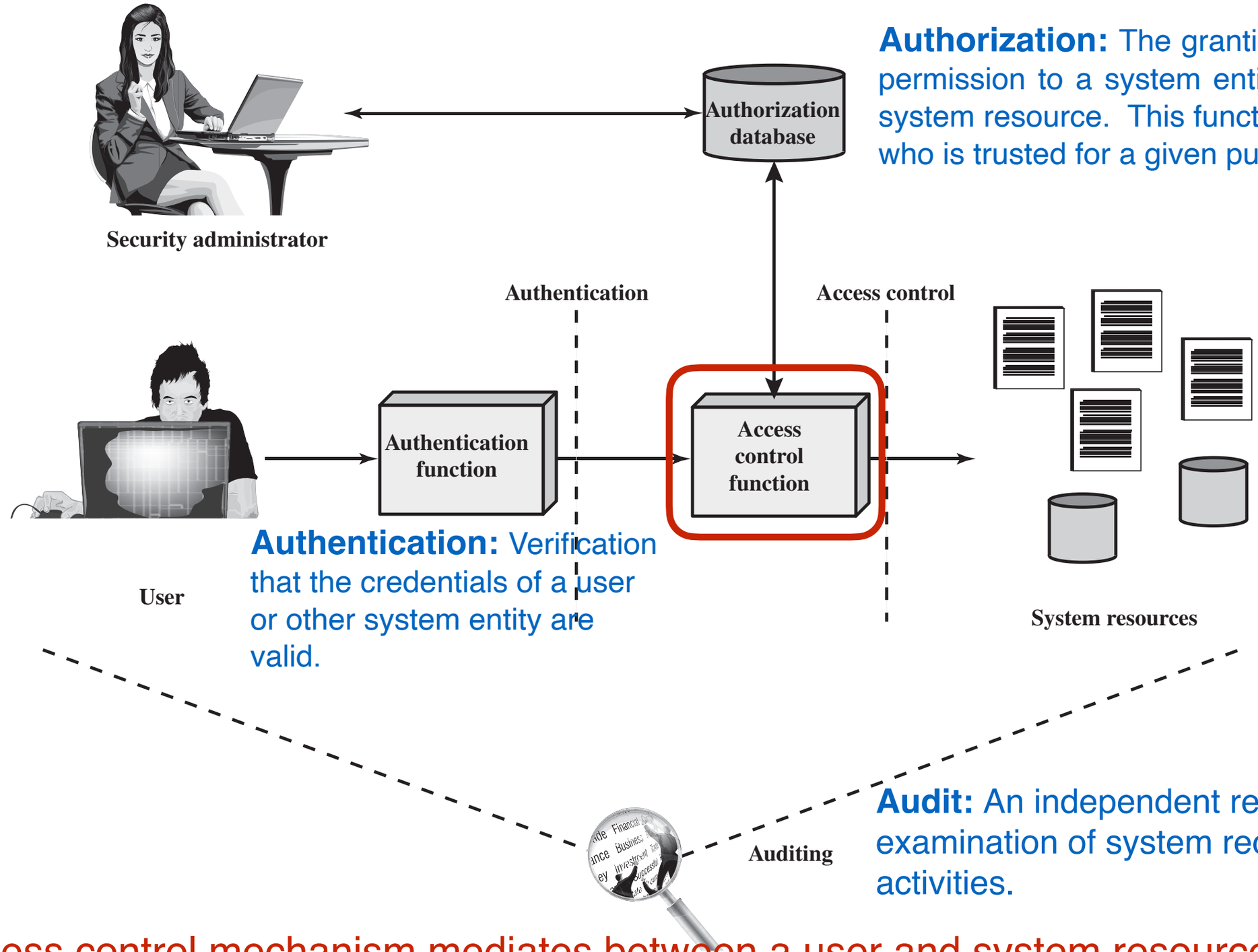
Reading: Chapter 4 Access Control in Book “Computer Security Principles and Practice (Third Edition)” by William Stallings and Lawrie Brown

Concept of Access Control

- Access control implements a security policy that specifies who or what (e.g., in the case of a process) may have access to each specific system resource and the type of access that is permitted in each instance.

  访问控制实现了一种安全策略，该策略指定谁或什么（例如，在进程的情况下）可以访问每个特定的系统资源，以及每个实例中允许的访问类型。

Access control in a broader context



Authorization: The granting of a right or permission to a system entity to access a system resource. This function determines who is trusted for a given purpose.

Authentication: Verification that the credentials of a user or other system entity are valid.

Audit: An independent review and examination of system records and activities.

An access control mechanism mediates between a user and system resources such as applications, operating systems, firewalls, routers, files, and database.

Basic Elements of Access Control

- **Object:** a resource to which access is controlled; an entity used to contain and/or receive information (e.g., pages, files, mailboxes…)
- **Subject:** an entity capable accessing objects
  - Owner (e.g., creator of a resource, system administrator)
  - Group (membership in the group is sufficient to exercise the access rights)
  - World (granted the least amount of access, not included in owner/group)
- **Access right:** describes the way in which a subject may access an object
  - Read: view information in a system resources, including ability to copy or print
  - Write: add, modify, or delete data; including read access
  - Execute: execute specified programs
  - Delete: delete certain system resources (e.g., files, records, programs)
  - Create: create new files, records, or fields
  - Search: list the files in a directory or otherwise search the directory

Access Control Policies

- Discretionary access control (DAC)
- Mandatory access control (MAC)
- Role-based access control (RBAM)
- Attribute-based access control (ABAC)

Not mutually exclusive.
An access control mechanism can employ two or even all three of these policies to cover different classes of system resources.

Discretionary Access Control (DAC)

- DAC controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do.

- An entity may be granted access rights that permit the entity, by its own volition, to enable another entity to access some resources.

DAC: Access Matrix

Dimension 1: Identified subjects that may attempt data access to the resources

确定可能尝试数据访问资源的对象

Dimension 2: objects that may be accessed

可以被访问的对象

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

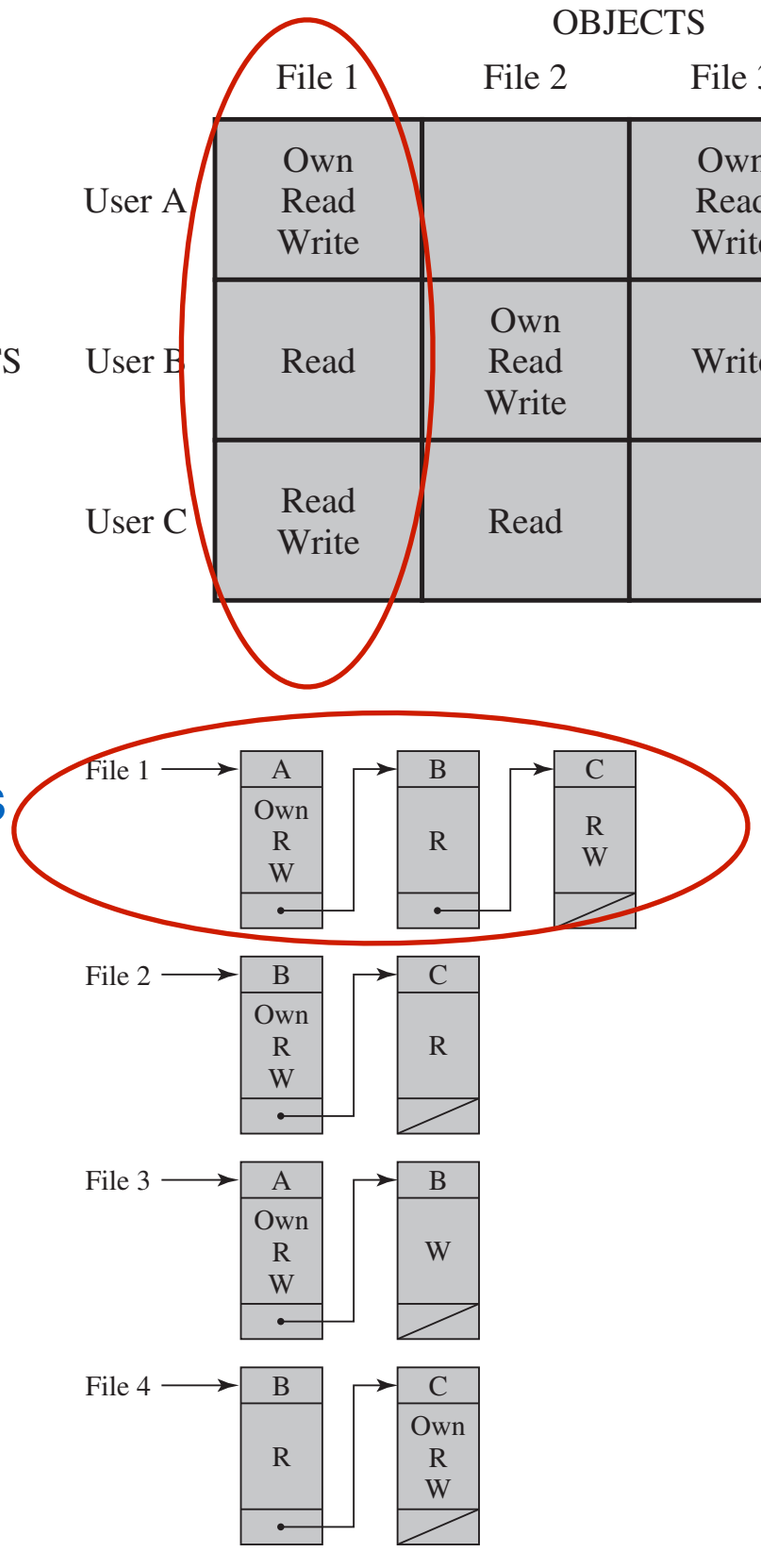
Access Matrix (sparse)

DAC: Access Matrix to ACL

Access Matrix

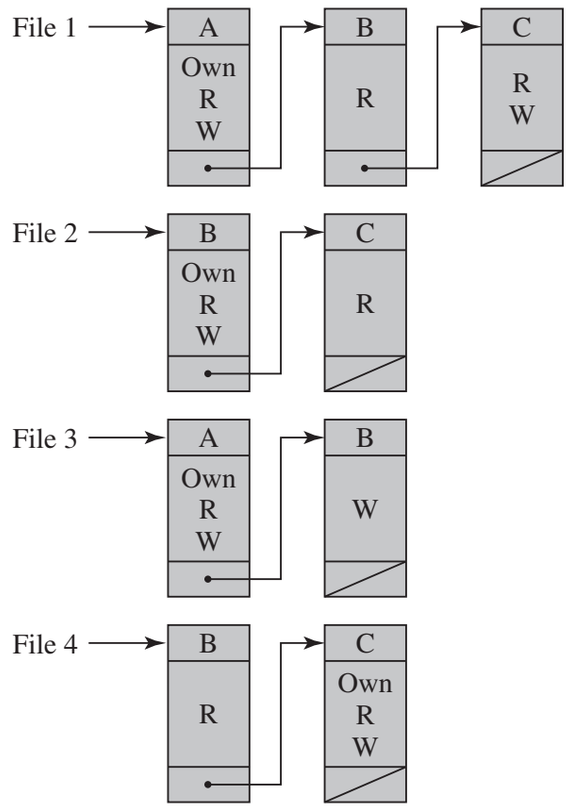
SUBJECTS	OBJECTS: File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

Access control lists (Columns)



ACL lists users and their permitted access rights

DAC: Access Control List



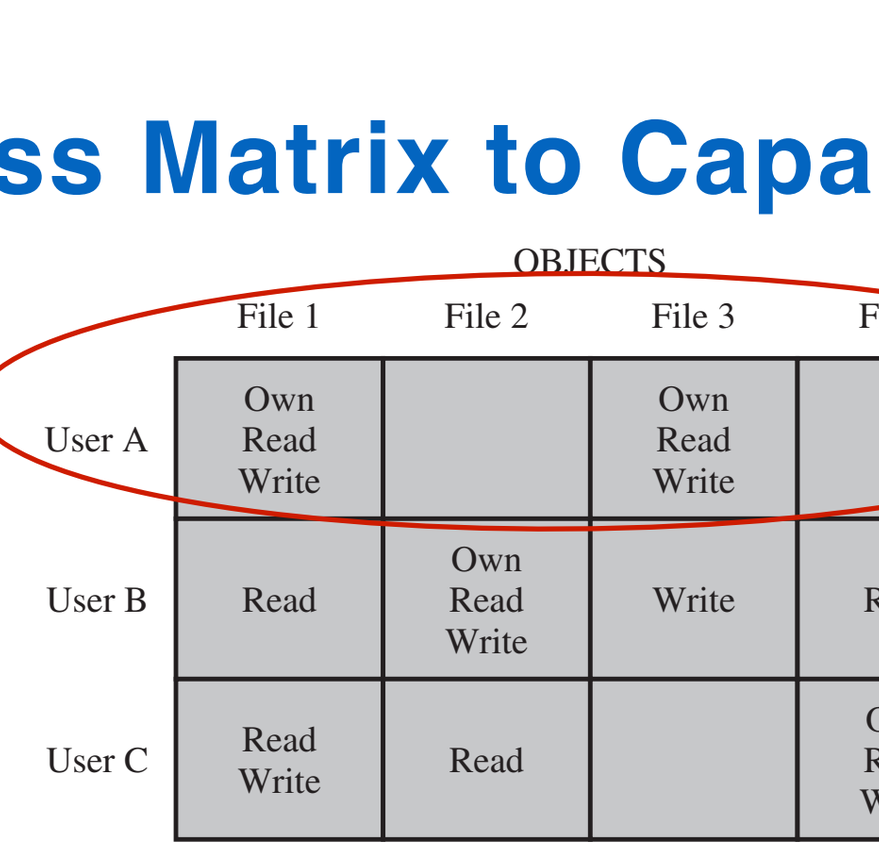
- Advantage
  - Contain a default, or public, entry (e.g. read-only access) that users are not explicitly listed
  - For a given resources, it is convenient for determining which subjects have which rights
- Disadvantage
  - Not convenient for determining the access rights available to a specific user

DAC: Access Matrix to Capability Tickets

Access Matrix

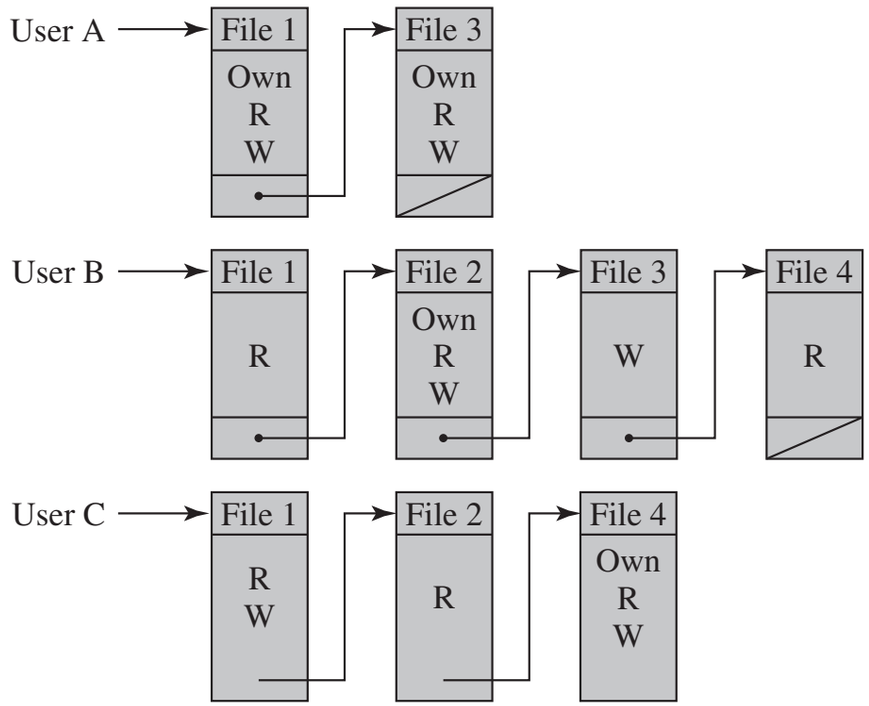
SUBJECTS	OBJECTS: File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

Capability tickets/lists (Rows)



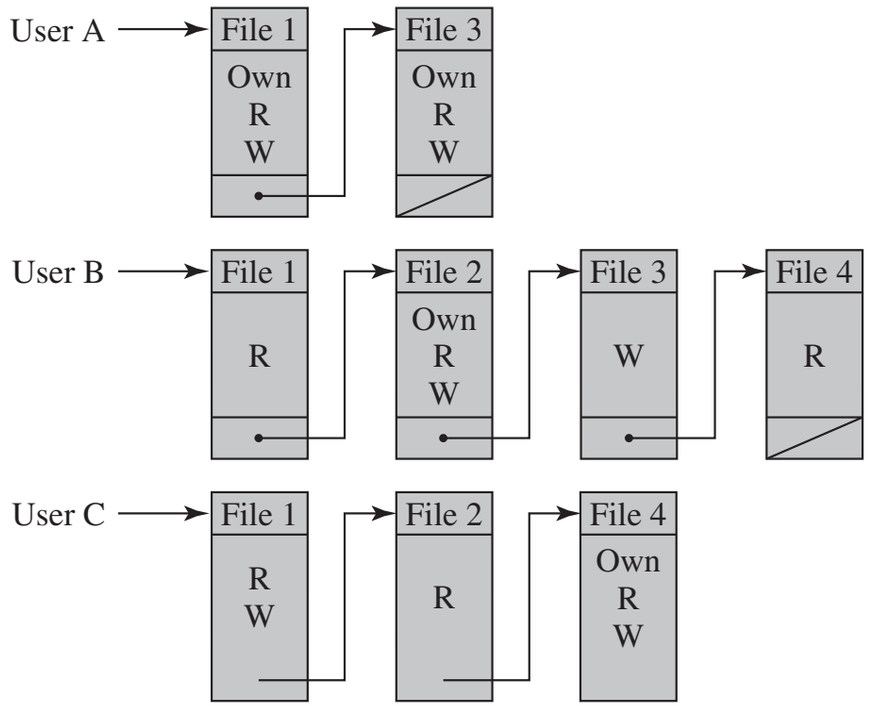
Capability tickets specifies authorized objects and operations for a particular user

DAC: Capability Tickets



- Advantages
  - Given a user, it is easy to determine the set of access rights
- Disadvantages
  - Given a specific resource, it is difficult to determine the list of users with specific access rights
  - Tickets may be authorized to loan or given to others, dispersed around the system —> security problem

DAC: Capability Tickets



- Advantages
  - Given a user, it is easy to determine the set of access rights
- Disadvantages
  - Given a specific resource, it is difficult to determine the list of users with specific access rights
  - Tickets may be authorized to loan or given to others, dispersed around the system —> security problem

Solution

Key idea: Make the ticket protected, guaranteed, and unforgeable

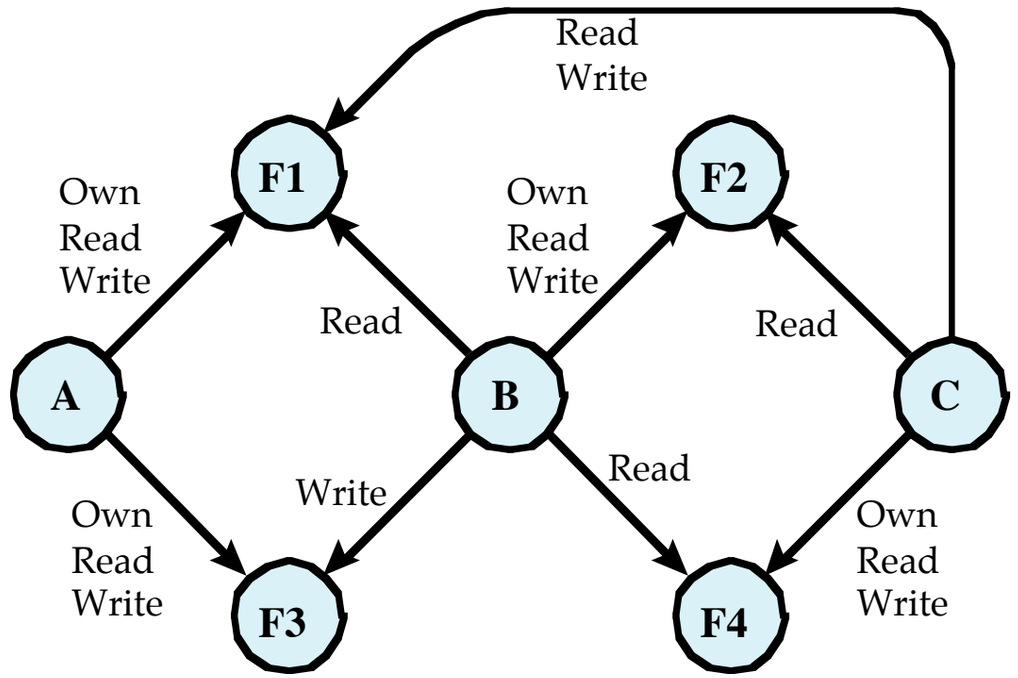
Solution 1: The operating system hold all tickets on behalf of users, but in a region of memory inaccessible to users

Solution 2: Include an unforgeable token (e.g., a large random pass word, or a cryptographic message authentication code) in the capability

DAC: Other Forms

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

Authorization Table



Directed Graph

DAC: Model

- Subjects, Objects, Rules
- Protection state: set of information, at a given point in time, that specifies the access rights for each subject with respect to each object.
- Three requirements
  - Representing the protection state
  - Enforcing access rights
  - Allowing subjects to alter the protection state in certain ways.

DAC: Model

- How to represent the protection state?
  - Extended access control matrix 扩展的访问控制矩阵

SUBJECTS	OBJECTS: Subjects S_1	S_2	S_3	Files F_1	F_2	Processes P_1	P_2	Disk drives 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			

Extended

* = copy flag set

-Processes: The ability to delete, stop(block), and wake up a process

-Devices: The ability to read/write the devices, to control its operation, and to block/unblock the device for use

-Memory locations/regions: The ability to read/write certain regions of memory that are protected such that the default is to disallow access

-Subjects: The ability to grant or delete rights of that subject to other objects

DAC: Model

OBJECTS

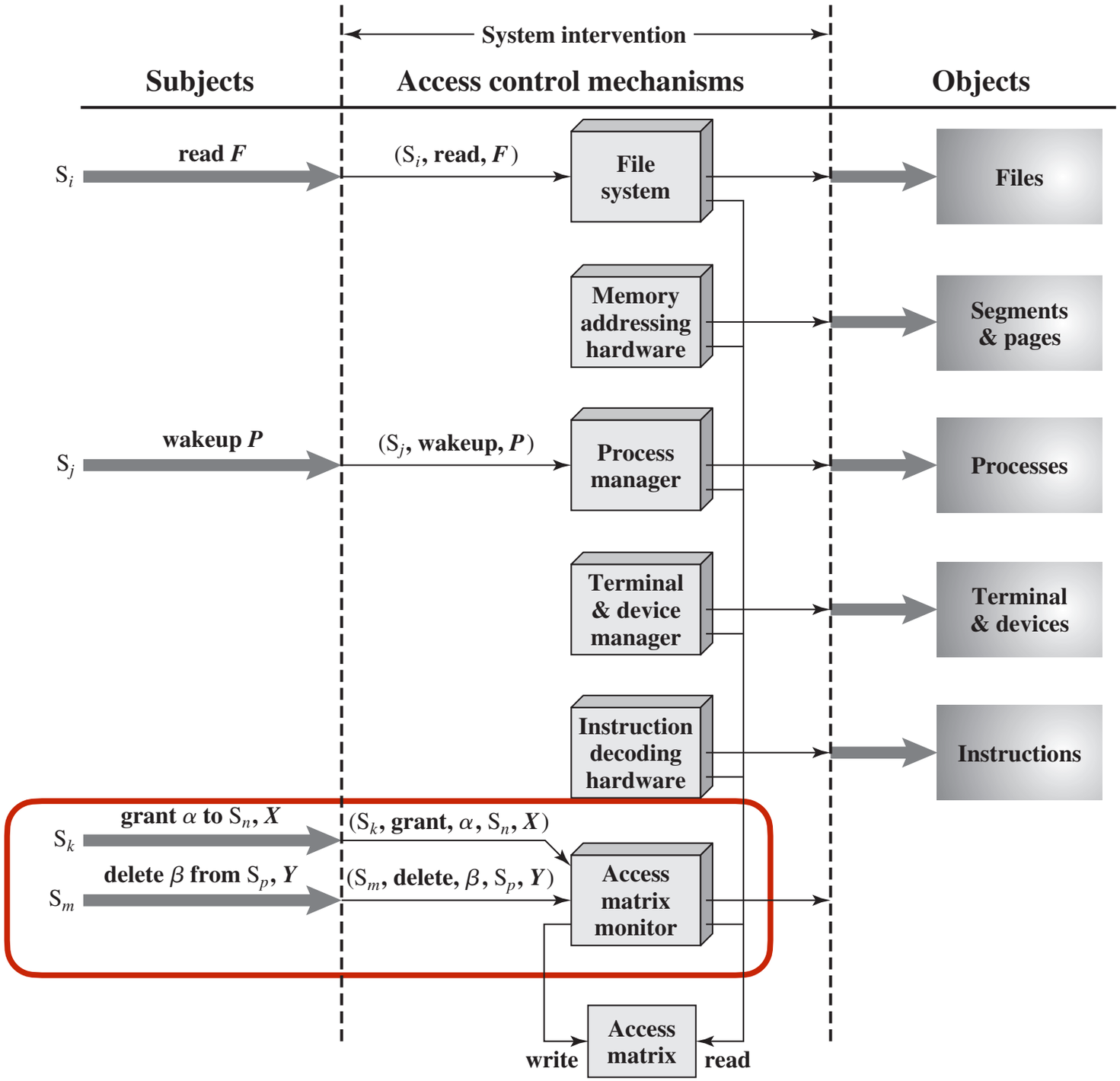
SUBJECTS	Subjects			Files		Processes		Disk drives	
	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read∗	read owner	wakeup	wakeup	seek	owner
S_2		control		write∗	execute			owner	seek∗
S_3			control		write	stop			

∗ = copy flag set

For an access control matrix A, each entry $A[S, X]$ contains strings, called access attributes, that specify the access rights of subject S to object X.

Example: $A[S_1, F_1]$. —> ‘read’ , that is S1 may read file F1.

DAC: Model



An Organization of the Access Control Function

DAC: Model

- A separate access control module is associated with each the of objects
- The module evaluates each request by the following steps
  该模块通过以下步骤评估每个请求

1. A subject S_0 issues a request of type α for object X.
2. The request causes the system (the operating system or an access control interface module of some sort) to generate a message of the form (S_0, α, X) to the controller for X.
3. The controller interrogates the access matrix A to determine if α is in $A[S_0, X]$. If so, the access is allowed; if not, the access is denied and a protection violation occurs. The violation should trigger a warning and appropriate action.

DAC: Model

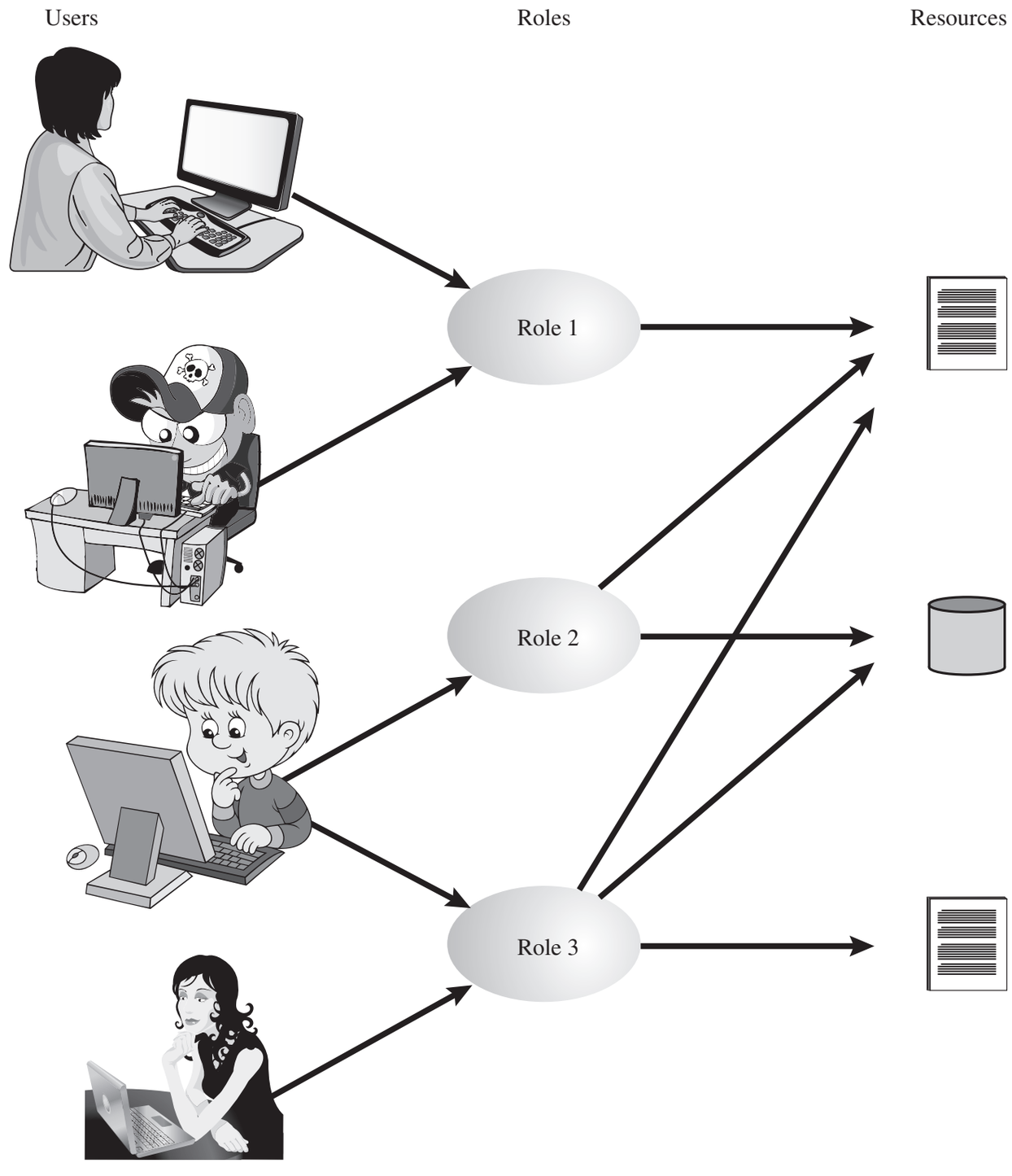
- How to modify the access matrix?
  - Access control system commands

Rule	Command (by S_0)	Authorization	Operation
R1	**transfer** $\left\{\begin{matrix}\alpha^* \\ \alpha\end{matrix}\right\}$ **to** S, X	'α^*' in $A[S_0, X]$	store $\left\{\begin{matrix}\alpha^* \\ \alpha\end{matrix}\right\}$ in $A[S, X]$
R2	**grant** $\left\{\begin{matrix}\alpha^* \\ \alpha\end{matrix}\right\}$ **to** S, X	'owner' in $A[S_0, X]$	store $\left\{\begin{matrix}\alpha^* \\ \alpha\end{matrix}\right\}$ 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

-R1: With a copy flag α^*, S_0 can transfer this right with/without copy flag to another subject.

-R4: Permits a subject to read that portion of the matrix that it owns or controls

Role-Based Access Control (RBAC)

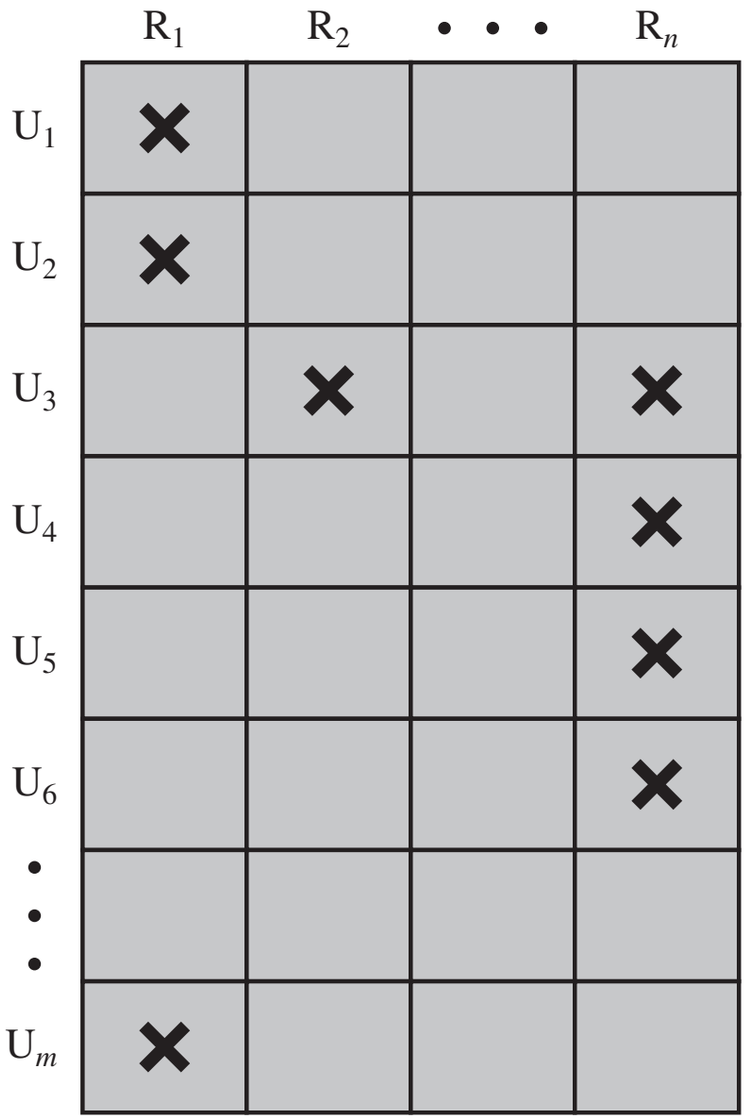


Role-Based Access Control (RBAC)

- RBAC controls access based on the **roles** that users have within the system and on rules stating what accesses are allowed to users in given roles (rather than user's identity in DAC).
- Users are assigned to different roles, either statically or dynamically, according to their responsibilities.
- The relationship of users to roles is many to many, as is the relationship of roles to resources, or system objects.

RBAC is active in commercial use and research :)

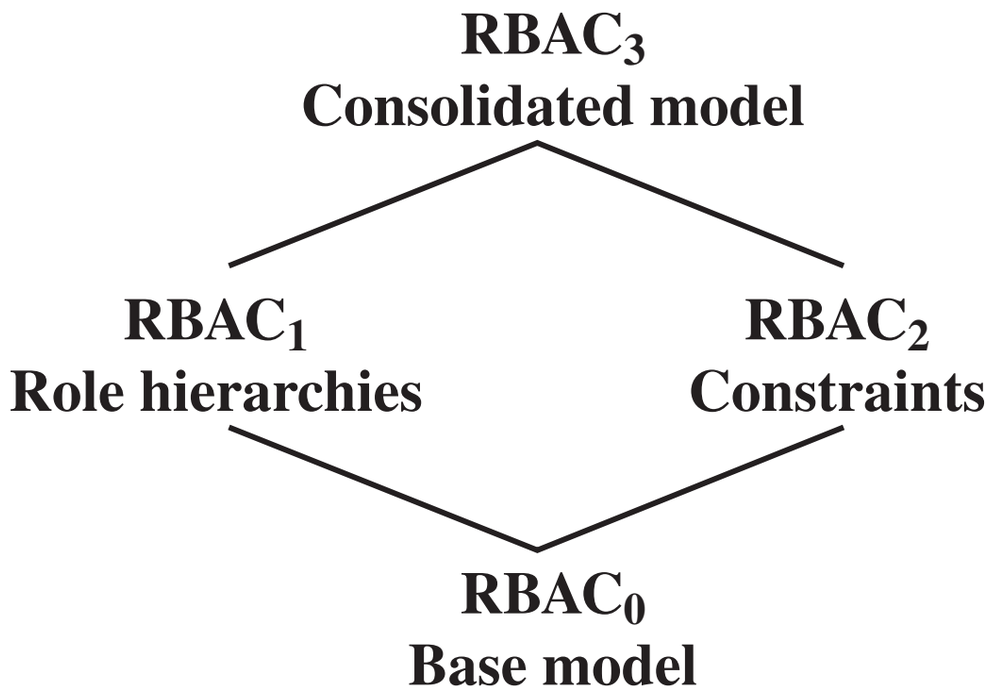
RBAC: Access Control Matrix



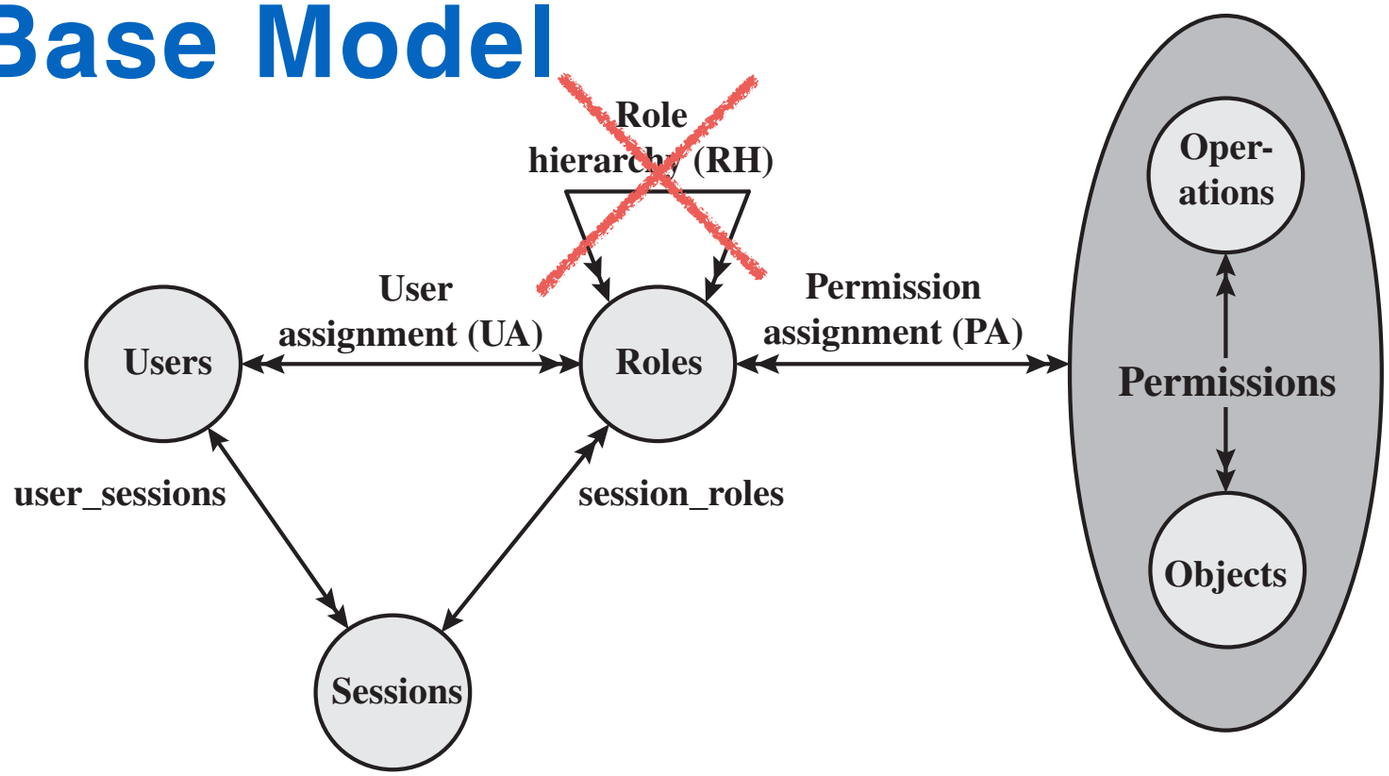
OBJECTS

ROLES	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			

RBAC: Models

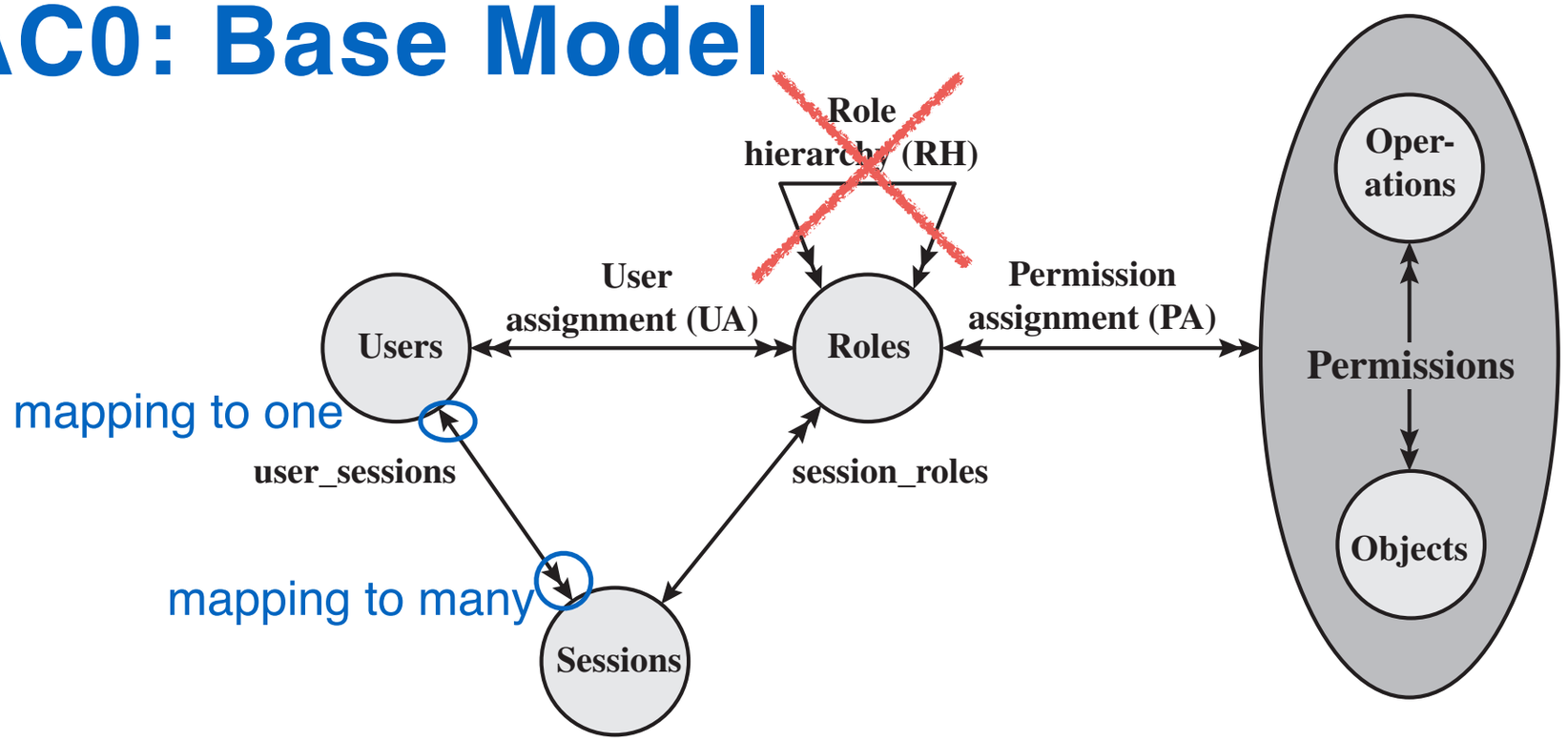

RBAC3
Consolidated model
RBAC1
Role hierarchies
RBAC2
Constraints
RBAC0
Base model

RBAC0: Base Model



- User: An individual that has access to this computer system. Each individual has an associated user ID.
- Role: A name of job function within the organization that controls this computer system. Typically, associated with each role is a description if the authority and responsibility conferred on this role, and on any user who assumes this role.
- Permission: An approval of a particular mode of access to one or more objects. Equivalent terms are access right, privilege, and authorization.
- Session: A mapping between a user and an activated subset of the set of roles to which the user is assigned.

RBAC0: Base Model



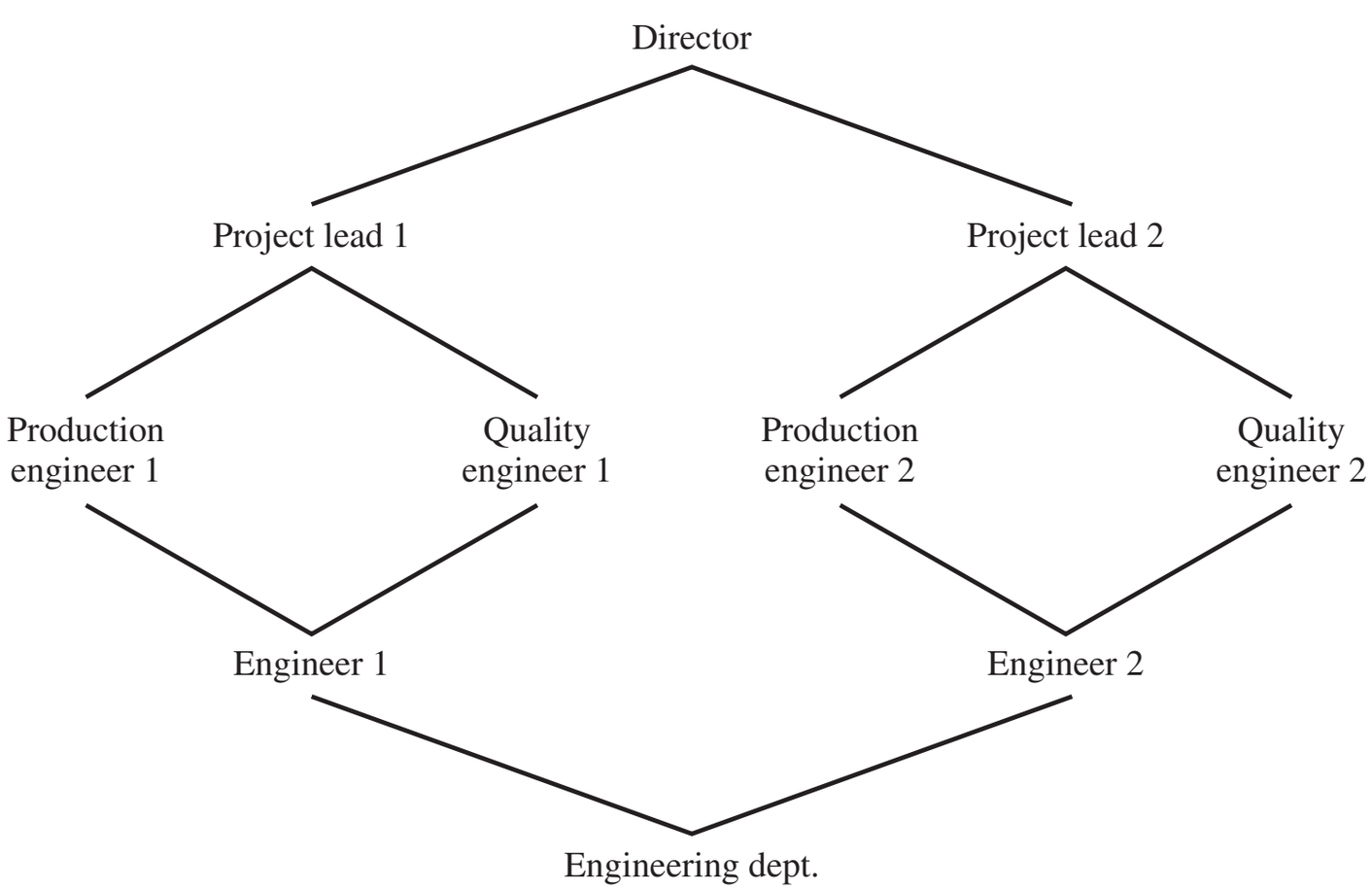
- Many-to-many between users and roles
- Many-to-many between roles and permissions

Flexibility and Granularity

Without flexibility and granularity, there is a greater risk that a user may be granted more access to resources than is needed because of the limited control over the types of access that can be allowed.

RBAC1: Role Hierarchies

- Job functions with greater responsibility have greater authority to access resources
- Role hierarchies make use of the concept of inheritance to enable one role to implicitly include access rights associated with a subordinate role



RBAC2: Constraints

- Constraints provide a means of adapting RBAC to the specifics of administrative and security policies in an organization.
  - Mutually Exclusive Roles
  - Cardinality
  - Prerequisite Roles

RBAC2: Constraints

- Mutually Exclusive Roles

  Separation of duties and capabilities within an organization

  - A user can only be assigned to one role in the set
  - Any permission (access right) can be granted to only one role in the set

Purpose: To increase difficulty of collusion among individuals of different skills or divergent job functions to thwart security policies

RBAC2: Constraints

- Cardinality

  Set a maximum number with respect to roles

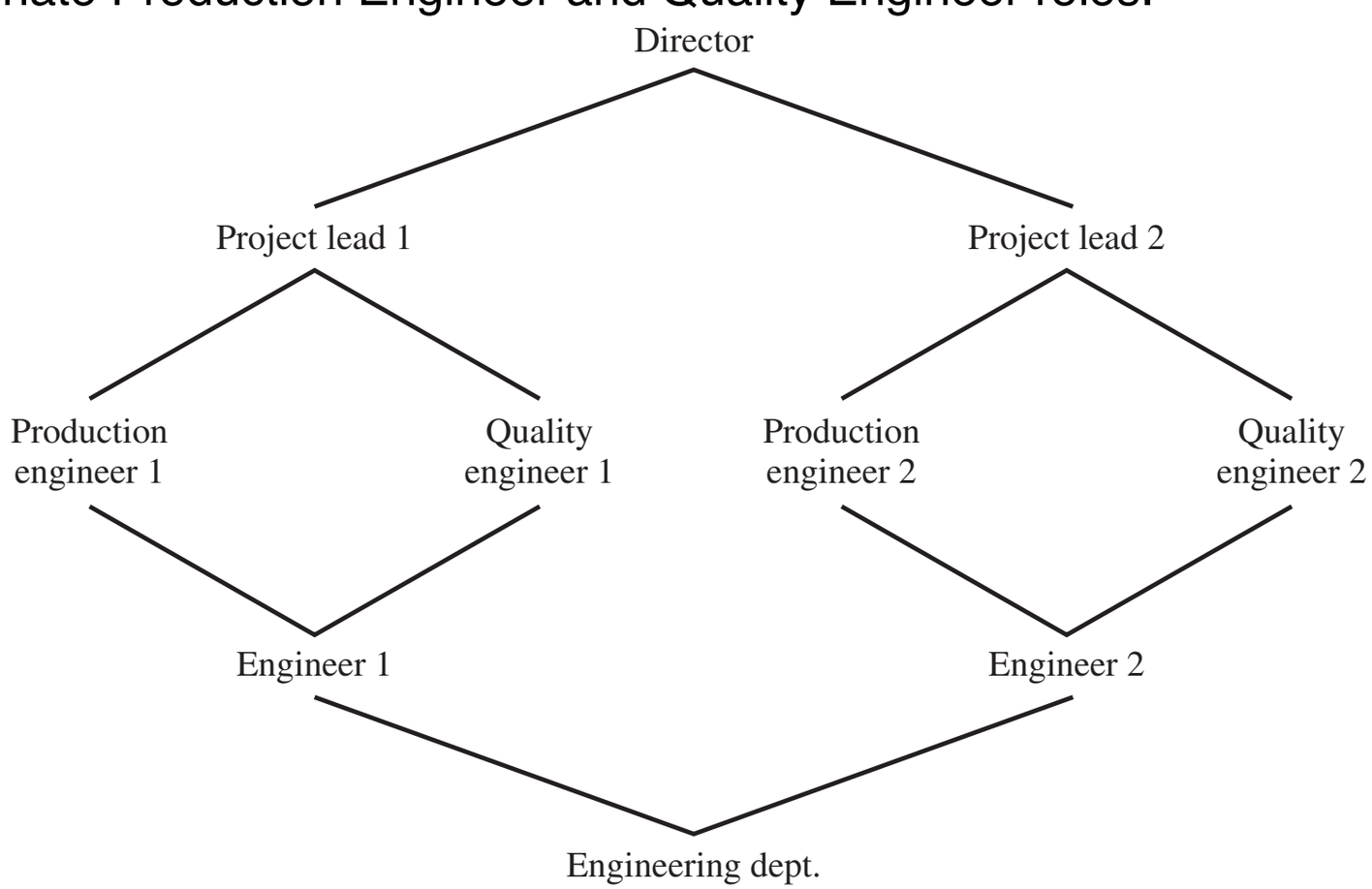
  - Set a maximum number of users that can be assigned to a given role
  - Constraint on the number of roles that is a user assigned to
  - Set a maximum number of roles that can be granted a particular permission

RBAC2: Constraints

- Prerequisite role

  A user can only be assigned to a particular role if it is already assigned to some other specified roles

  Example: In a hierarchy, a user assigned to a Project Lead role must also be assigned to at the subordinate Production Engineer and Quality Engineer roles.



Attribute-Based Access Control (ABAC)

- ABAC controls access based on attributes of the users, the resources to be accessed, and current environmental conditions.
- Flexibility and expressive power

- Three key elements
  - Attributes
  - Architecture Model
  - Policies

ABAC: Attributes

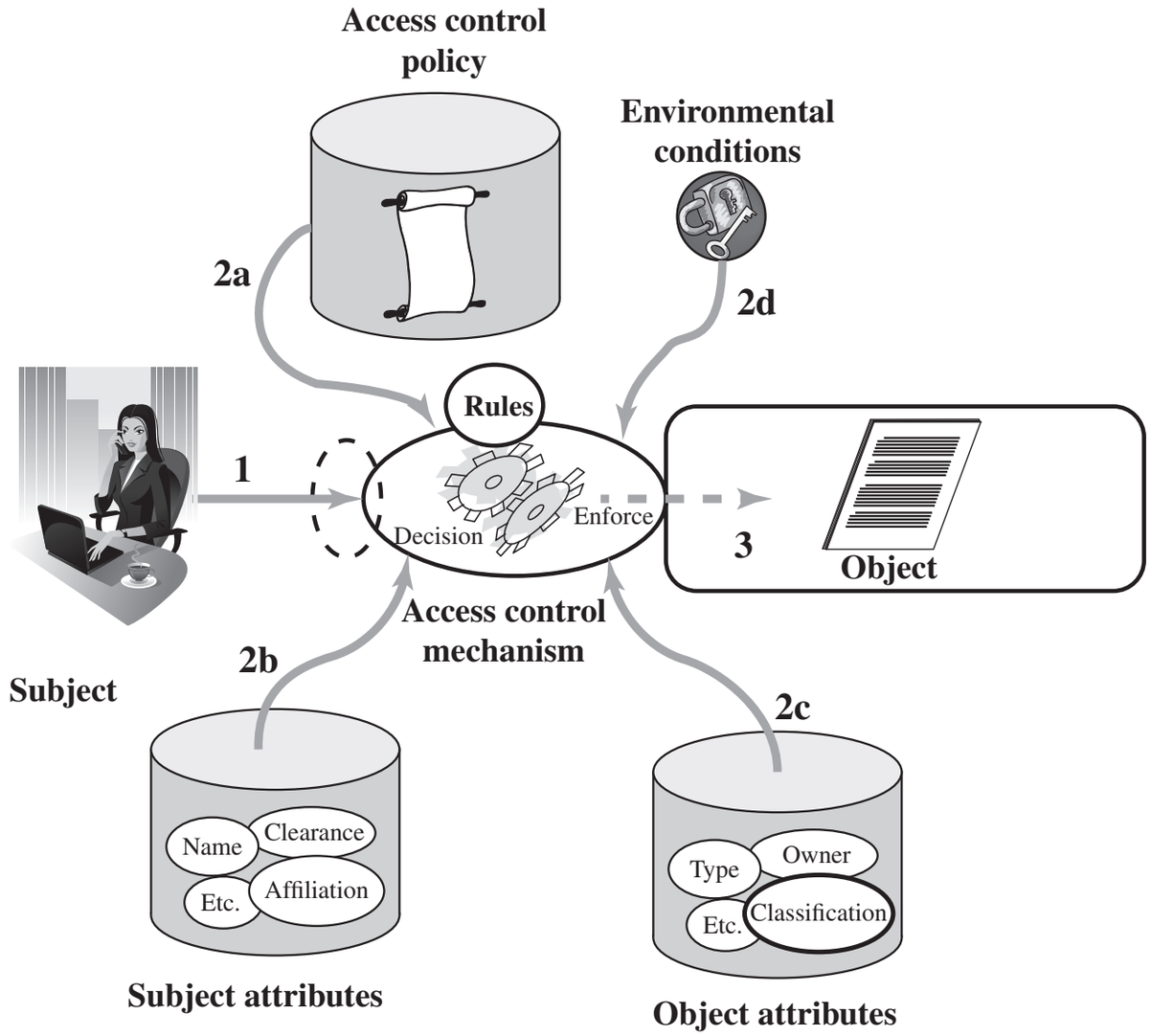
It defines specific aspects of the subject, object, environment conditions, and/or requested operations that are predefined or preassigned by an authority.

- Subject attributes: Define the identity and characteristics of the subject (e.g., the subject's identifier, name, organization, job titles…)
- Object attributes: Can be extracted from the metadata of the object and leveraged to make access control decisions (e.g. title, date, author of a Microsoft Word document)
- Environment attributes (ignored in most access control policies): Describe the operational, technical, and even situational environment or context in which the information access occurs; not associated with a particular subject nor a object/resource (e.g., current data and time, virus/hacker activities, and the network's security level)

ABAC: Logical Architecture

- Use four independent sources for the access control decision — powerful and flexible

Complexity and Performance Tradeoff



ABAC: Policies

1. S, O, and E are subjects, objects, and environments, respectively;
2. SA_k $(1 \leq k \leq K)$, OA_m $(1 \leq m \leq M)$, and EA_n $(1 \leq n \leq N)$ are the pre-defined attributes for subjects, objects, and environments, respectively;
3. ATTR(s), ATTR(o), and ATTR(e) are attribute assignment relations for subject s, object o, and environment e, respectively:

$$\texttt{ATTR}(s) \subseteq \texttt{SA}_1 \times \texttt{SA}_2 \times \ldots \times \texttt{SA}_K$$
$$\texttt{ATTR}(o) \subseteq \texttt{OA}_1 \times \texttt{OA}_2 \times \ldots \times \texttt{OA}_M$$
$$\texttt{ATTR}(e) \subseteq \texttt{EA}_1 \times \texttt{EA}_2 \times \ldots \times \texttt{EA}_N$$

4. In the most general form, a Policy Rule, which decides on whether a subject s can access an object o in a particular environment e, is a Boolean function of the attributes of s, o, and e:

   $\texttt{Rule: can\_access } (s, o, e) \leftarrow f(\texttt{ATTR}(s), \texttt{ATTR}(o), \texttt{ATTR}(e))$

   Given all the attribute assignments of s, o, and e, if the function's evaluation is true, then the access to the resource is granted; otherwise the access is denied.
5. A policy rule base or policy store may consist of a number of policy rules, covering many subjects and objects within a security domain. The access control decision process in essence amounts to the evaluation of applicable policy rules in the policy store.

ABAC: Example

- An online entertainment store enforces the following access control policy based on the user's age and the movie content rating:

Movie Rating	Users Allowed Access
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

-RBAC: Three roles (Adult, Juvenile, Child), Tree Permissions (can view R-rated movies, can view PG-13-rated movies, and can view G-rated movies). User-to-role assignment and the permission-to-tole assignment are manual admin tasks.

-ABAC: Without explicitly defining roles! Whether a user u can access a movie m (in a security environment e which is ignored here) would be resolved by evaluating a policy rule below.

$$
\begin{aligned}
&\text{R1:}\mathtt{can\_access}(u,\ m,\ e) \leftarrow \\
&\quad (\mathtt{Age}(u) \geq 17 \wedge \mathtt{Rating}(m) \in \{\mathtt{R},\ \mathtt{PG\text{-}13},\ \mathtt{G}\}) \vee \\
&\quad (\mathtt{Age}(u) \geq 13 \wedge \mathtt{Age}(u) < 17 \wedge \mathtt{Rating}(\mathtt{m}) \in \{\mathtt{PG\text{-}13},\ \mathtt{G}\}) \vee \\
&\quad (\mathtt{Age}(u) < 13 \wedge \mathtt{Rating}(m) \in \{\mathtt{G}\})
\end{aligned}
$$

ABAC: Example

- Suppose movies are classified as either New Release or Old Release, based on release date compared to the current date, and users are classified as Premium User and Regular User, based on the fee they pay.

R2:can_access(u, m, e) ←
(MembershipType(u) = Premium)
(MembershipType(u) = Regular ∧ MovieType(m) = OldRelease)

R3:can_access(u, m, e) ← R1 ∧ R2

Summary

- Concept of Access Control
  - Definition
  - Access control in a broader context
  - Basic elements: object, subject, access right
- Access Control Policies
  - Discretionary access control (DAC)
    - Access matrix -> access control list/capability tickets
    - Model: protection state (extend access matrix)
  - Role-based access control (RBAM)
    - RBAC0, RBAC1 (hierarchies), RBAC2 (constraints)
  - Attribute-based access control (ABAC)
    - Subject/Object Attribute and Environment Attribute