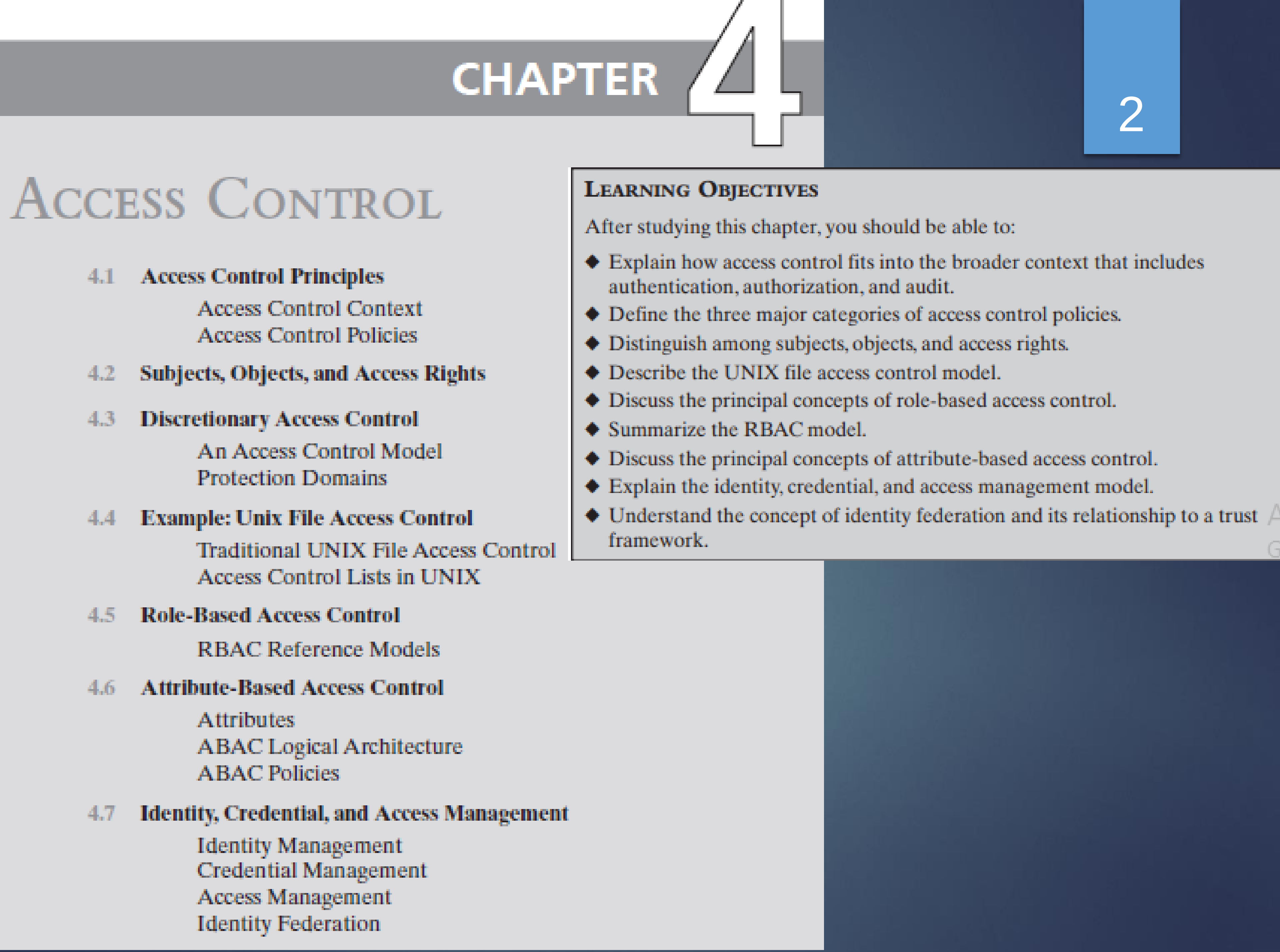
# Information Security Fall 2022

**WEEK # 8**

**LECTURE # 19, 20 AND 21**

**DR. AQSA ASLAM**



CHAPTER 4

# Access Control

## Learning Objectives

After studying this chapter, you should be able to:

- Explain how access control fits into the broader context that includes authentication, authorization, and audit.
- Define the three major categories of access control policies.
- Distinguish among subjects, objects, and access rights.
- Describe the UNIX file access control model.
- Discuss the principal concepts of role-based access control.
- Summarize the RBAC model.
- Discuss the principal concepts of attribute-based access control.
- Explain the identity, credential, and access management model.
- Understand the concept of identity federation and its relationship to a trust framework.

## Access Control Definitions

**Two definitions of access control are useful in understanding its scope**

NISTIR 7298 defines access control as:

**“the process of granting or denying specific requests to:(1) obtain and use information and related information processing services; and (2) enter specific physical facilities”**

## Access Control Definitions

RFC 4949 defines access control as:

"a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy"

## Access Control Principles

- In a broad sense, all of computer security is concerned with access control

- RFC 4949 defines computer security as:

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

## Access Control Context

- Access control context involves the following entities and functions:
  - **Authentication:**
    - Verification that the credentials of a user or other system entity are valid.
  - **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.
  - **Audit:**
    - An independent review and examination of system records and activities in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures, to detect breaches in security, and to recommend any indicated changes in control, policy, and procedures.

# Access Control Context

- An access control mechanism mediates between
  - a user (or a process executing on behalf of a user) and system resources
  - such as applications, operating systems, firewalls, routers, files, and databases

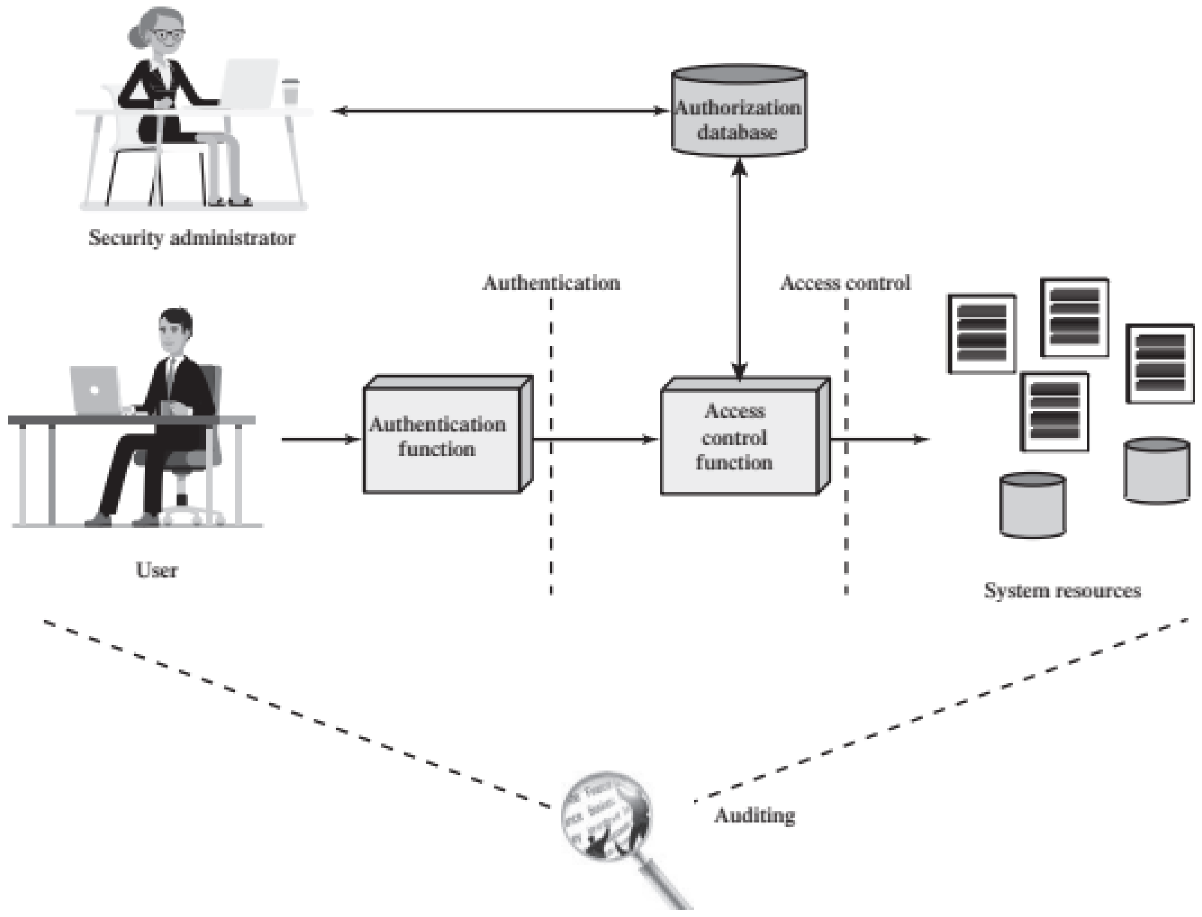


**Figure 4.1 Relationship Among Access Control and Other Security Functions**
*Source:* Based on [SAND94].

## Access Control Policies

- **Discretionary access control (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
- **Mandatory access control (MAC)**
  - Controls access based on comparing security labels with security clearances
- **Role-based access control (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
- **Attribute-based access control (ABAC)**
  - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

## Subjects, Objects, and Access Rights

### Subject

An entity capable of accessing objects

Three classes
- Owner
- Group
- World

### Object

A resource to which access is controlled

Entity used to contain and/or receive information

### Access right

Describes the way in which a subject may access an object

Could include:
- Read
- Write
- Execute
- Delete
- Create
- Search

## Subjects, Objects, and Access Rights

- **Subject**
  - **Owner:** This may be the creator of a resource, such as a file. For system resources, ownership may belong to a system administrator. For project resources, a project administrator or leader may be assigned ownership.
  - **Group:** In addition to the privileges assigned to an owner, a named group of users may also be granted access rights, such that membership in the group is sufficient to exercise these access rights. In most schemes, a user may belong to multiple groups
  - **World:** The least amount of access is granted to users who are able to access the system but are not included in the categories owner and group for this resource
- **Object**
  - **Examples** include records, blocks, pages, segments, files, portions of files, directories, directory trees, mailboxes, messages, and programs. Some access control systems also encompass, bits, bytes, words, processors, communication ports, clocks, and network nodes.
- **Access rights**
  - **Example:** read, write, execute, delete, create, search

## Subjects, Objects, and Access Rights

- An access right describes the way in which a subject may access an object.
- Access rights could include the following:
  - **Read:** User may view information in a system resource
    - (e.g., a file, selected records in a file, selected fields within a record, or some combination). Read access includes the ability to copy or print.
  - **Write:** User may add, modify, or delete data in system resource
    - (e.g., files, records, programs). Write access includes read access.
  - **Execute:** User may execute specified programs.
  - **Delete:** User may delete certain system resources, such as files or records.
  - **Create:** User may create new files, records, or fields.
  - **Search:** User may list the files in a directory or otherwise search the directory.

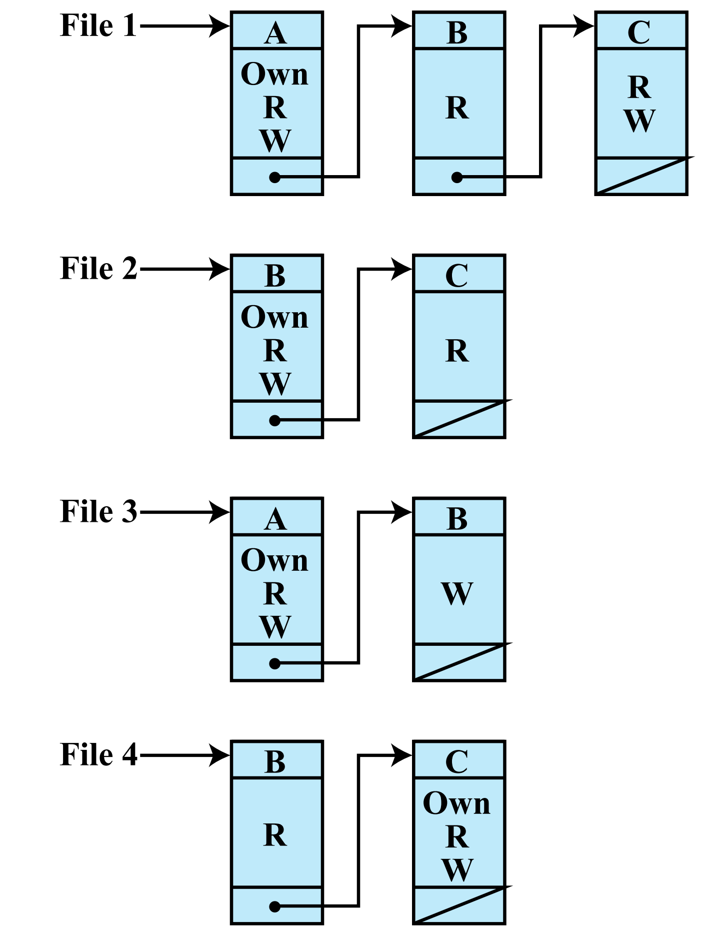
## Discretionary Access Control (DAC)

- Scheme in which an entity may be granted access rights that permit the entity, by its own violation, to 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

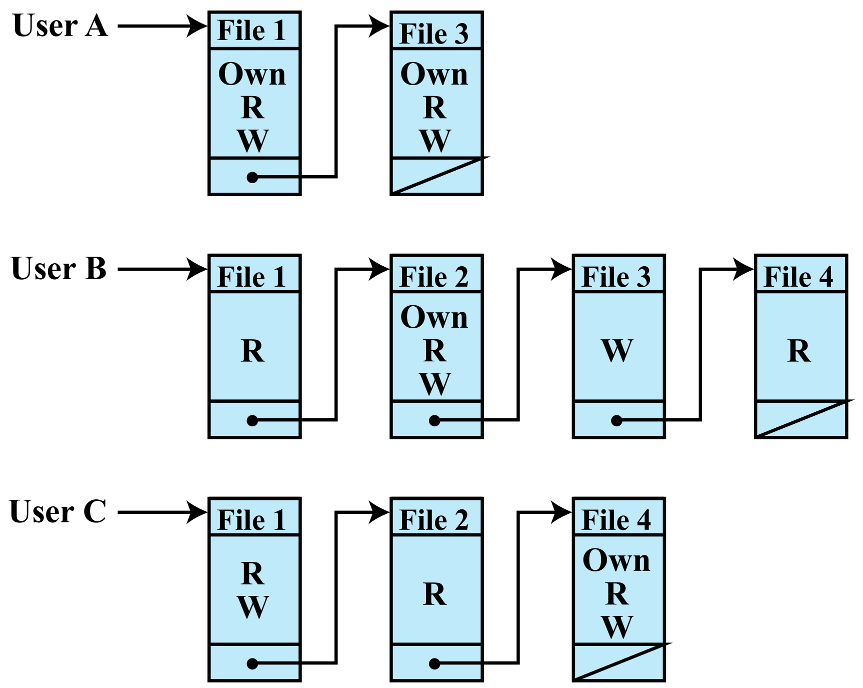
| | | OBJECTS | | | |
|---|---|---|---|---|---|
| | | File 1 | File 2 | File 3 | File 4 |
| | User A | Own Read Write | | Own Read Write | |
| SUBJECTS | User B | Read | Own Read Write | Write | Read |
| | User C | Read Write | Read | | Own Read Write |

(a) Access matrix

**Figure 4.2 Example of Access Control Structures**



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



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

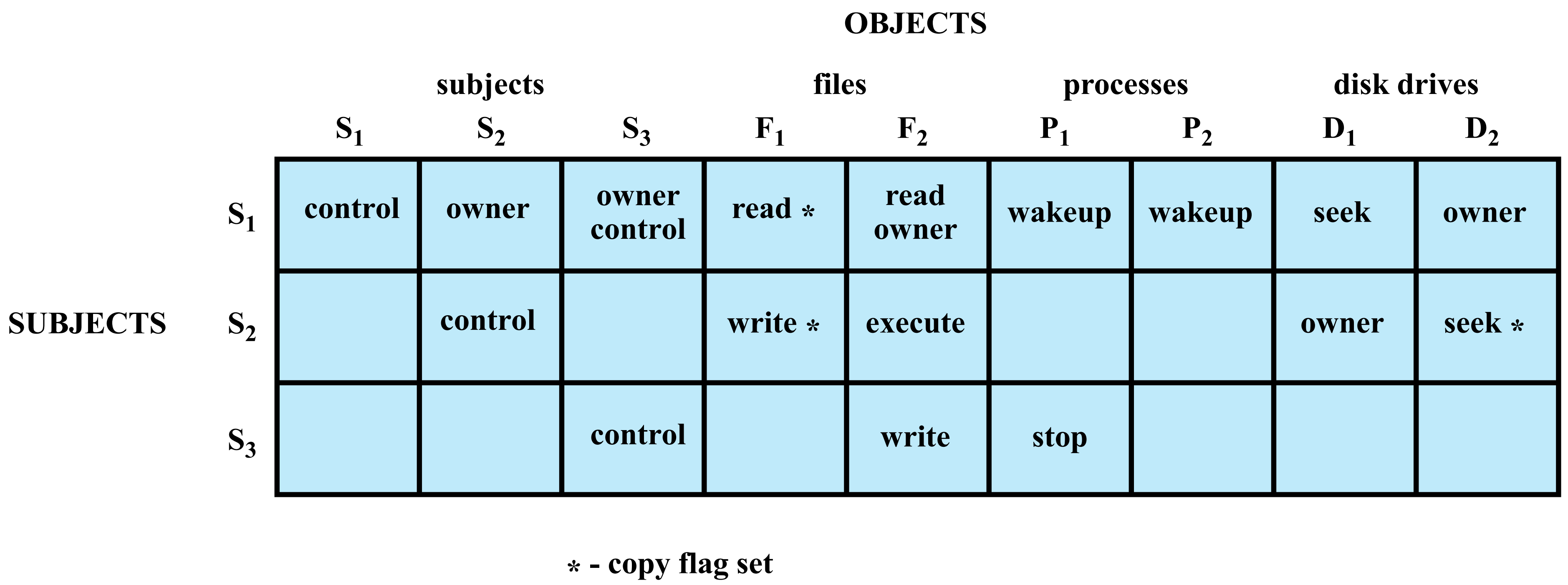
## Figure 4.2 Example of Access Control Structures

## Table 4.2

## Authorization Table for Files in Figure 4.2

| 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 |

(Table is on page 113 in the textbook)



| | | 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 4.3 Extended Access Control Matrix

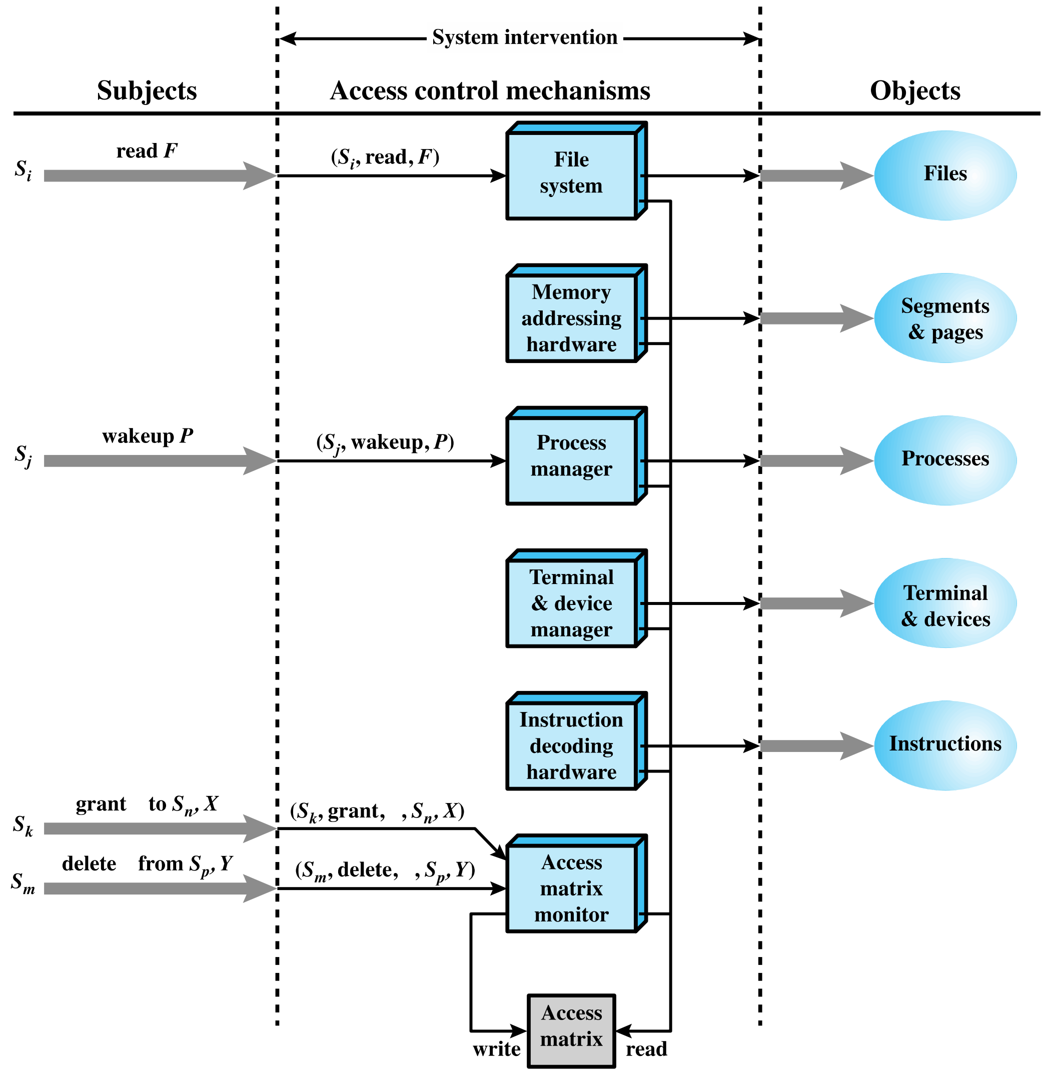


Figure 4.4 An Organization of the Access Control Function

## 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

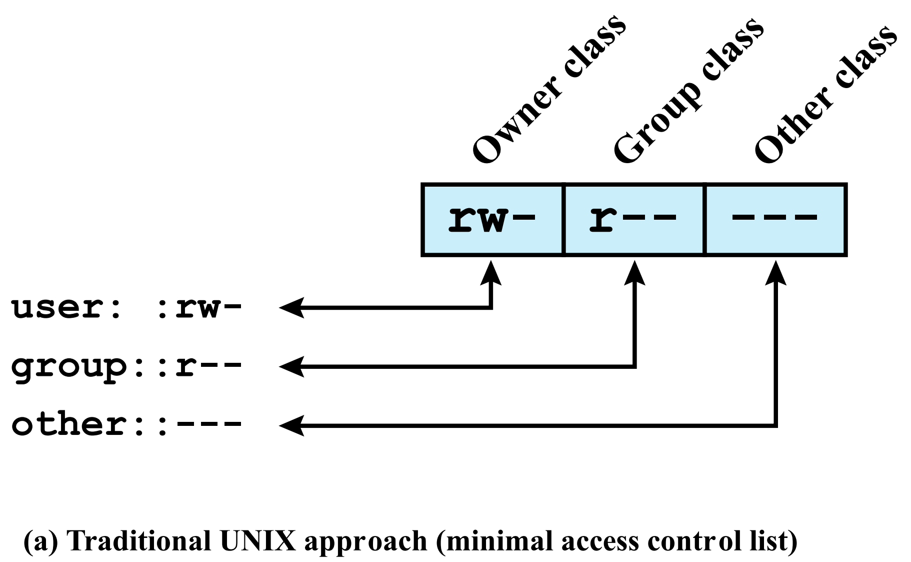


Figure 4.5 UNIX File Access Control

## 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**

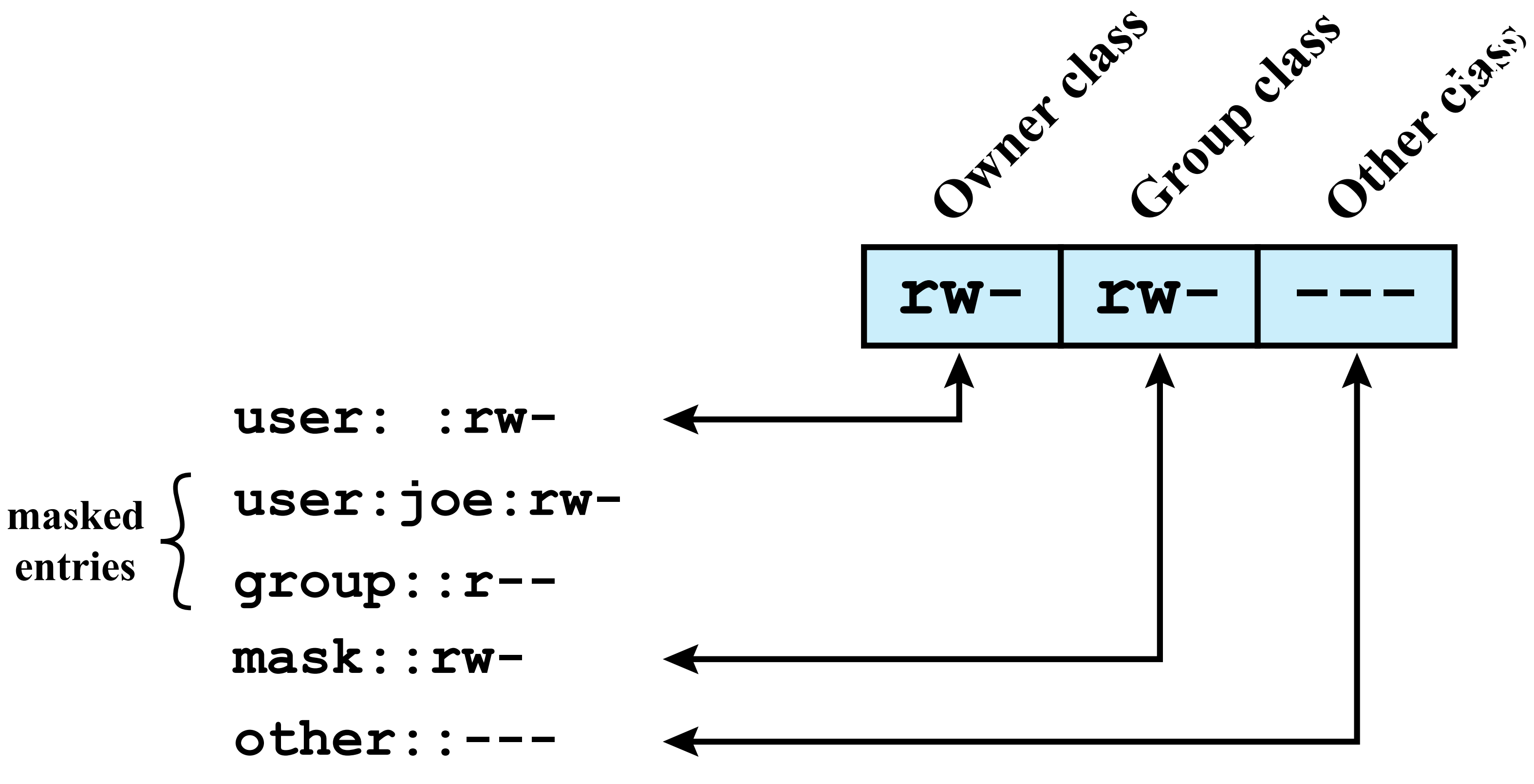
- FreeBSD, OpenBSD, Linux, Solaris

**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
- Step 2 checks if the matching entry contains sufficient permissions



(b) Extended access control list

Figure 4.5 UNIX File Access Control

## Role-Based Access Control (RBAC)

- RBAC is based on the roles that users assume in a system rather than the user's identity.
- RBAC models define a role as a job function within an organization.
- RBAC systems assign access rights to roles instead of individual users
  - In turn, 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 (see Figure 4.6).
- The set of users changes, in some environments frequently, and the assignment of a user to one or more roles may also be dynamic.
- The set of roles in the system in most environments is relatively static, with only occasional additions or deletions.
- Each role will have specific access rights to one or more resources.
- The set of resources and the specific access rights associated with a particular role are also likely to change infrequently.

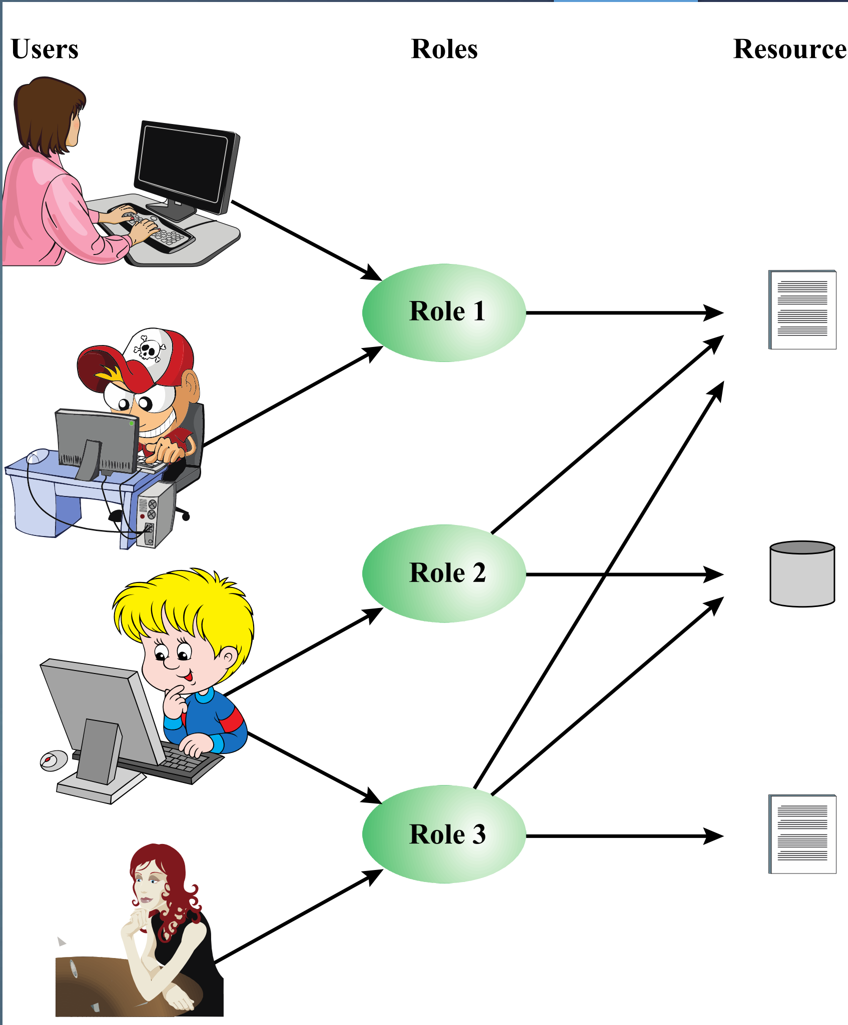
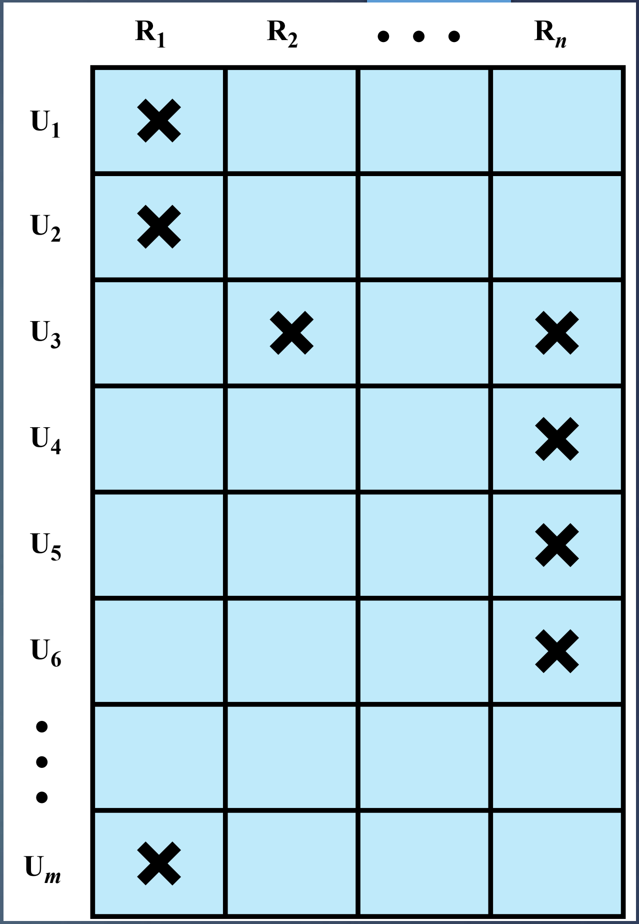


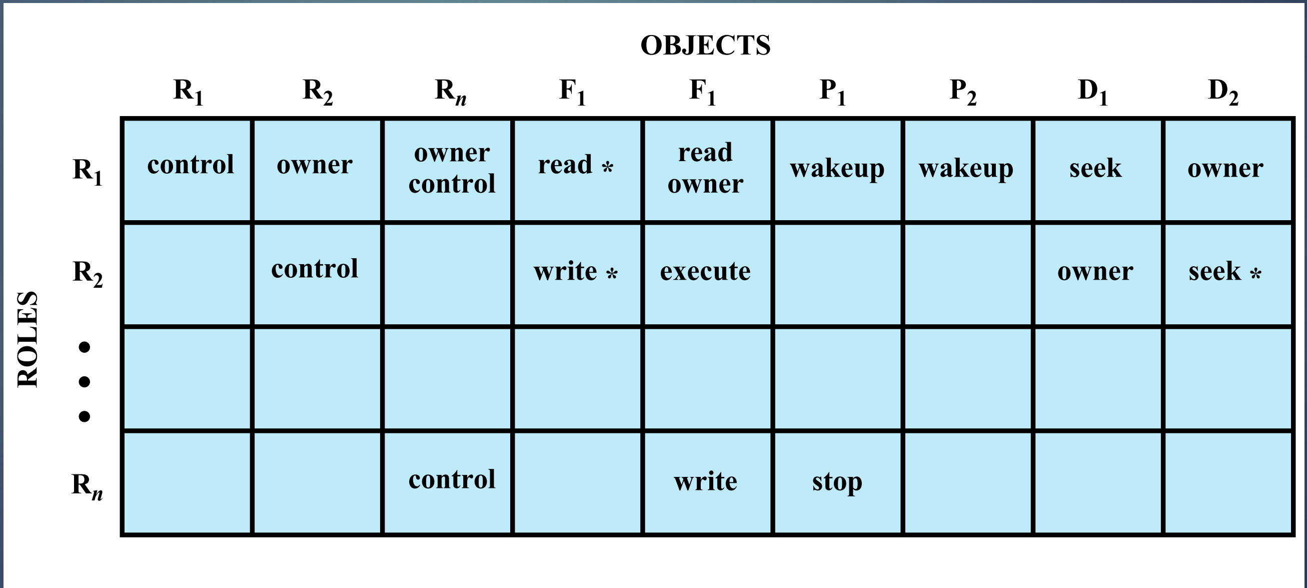
**Figure 4.6 Users, Roles, and Resources**

- We can use the access matrix representation to depict the key elements of an RBAC system in simple terms, as shown in Figure 4.7.
- The upper matrix relates individual users to roles.
- Typically there are many more users than roles.
- Each matrix entry is either blank or marked, the latter indicating that this user is assigned to this role.
  - Note a single user may be assigned multiple roles (more than one mark in a row) and multiple users may be assigned to a single role (more than one mark in a column



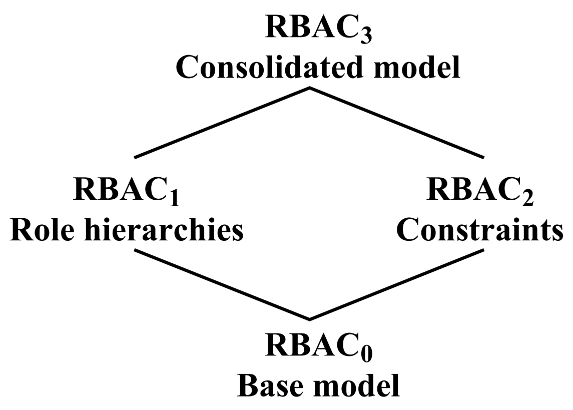
| | $R_1$ | $R_2$ | • • • | $R_n$ |
|---|---|---|---|---|
| $U_1$ | ✗ | | | |
| $U_2$ | ✗ | | | |
| $U_3$ | | ✗ | | ✗ |
| $U_4$ | | | | ✗ |
| $U_5$ | | | | ✗ |
| $U_6$ | | | | ✗ |
| ⋮ | | | | |
| $U_m$ | ✗ | | | |

- The lower matrix has the same structure as the DAC access control matrix, with roles as subjects.
- Typically, there are few roles and many objects, or resources.
- In this matrix, the entries are the specific access rights enjoyed by the roles.
  - Note a role can be treated as an object, allowing the definition of role hierarchies.

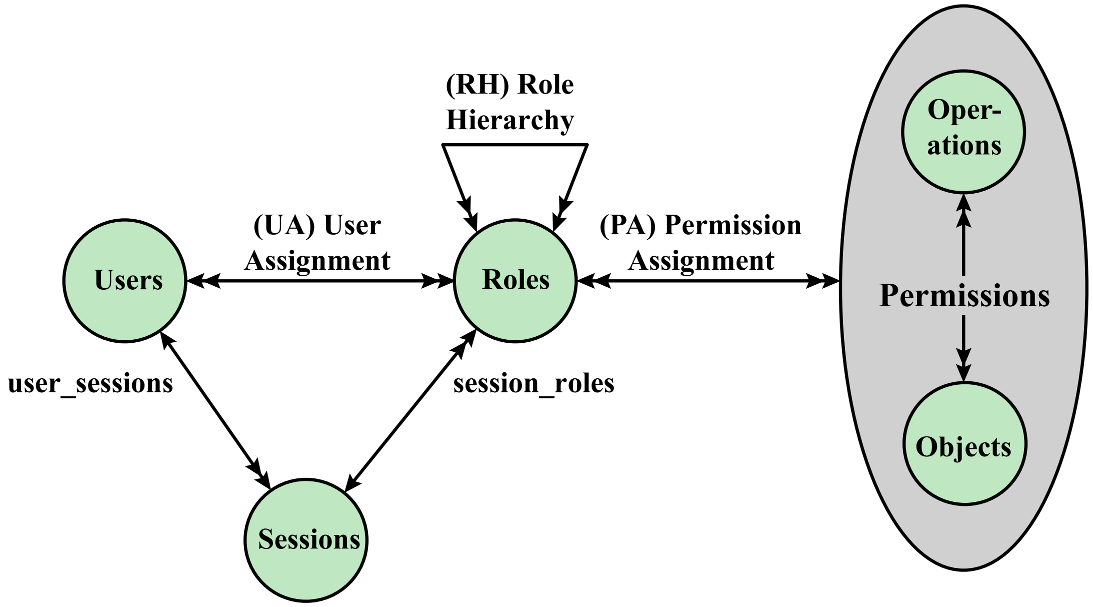


| | | OBJECTS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $R_1$ | $R_2$ | $R_n$ | $F_1$ | $F_1$ | $P_1$ | $P_2$ | $D_1$ | $D_2$ |
| ROLES | $R_1$ | control | owner | owner control | read * | read owner | wakeup | wakeup | seek | owner |
| | $R_2$ | | control | | write * | execute | | | owner | seek * |
| | • • • | | | | | | | | | |
| | $R_n$ | | | control | | write | stop | | | |

**Figure 4.7 Access Control Matrix Representation of RBAC**



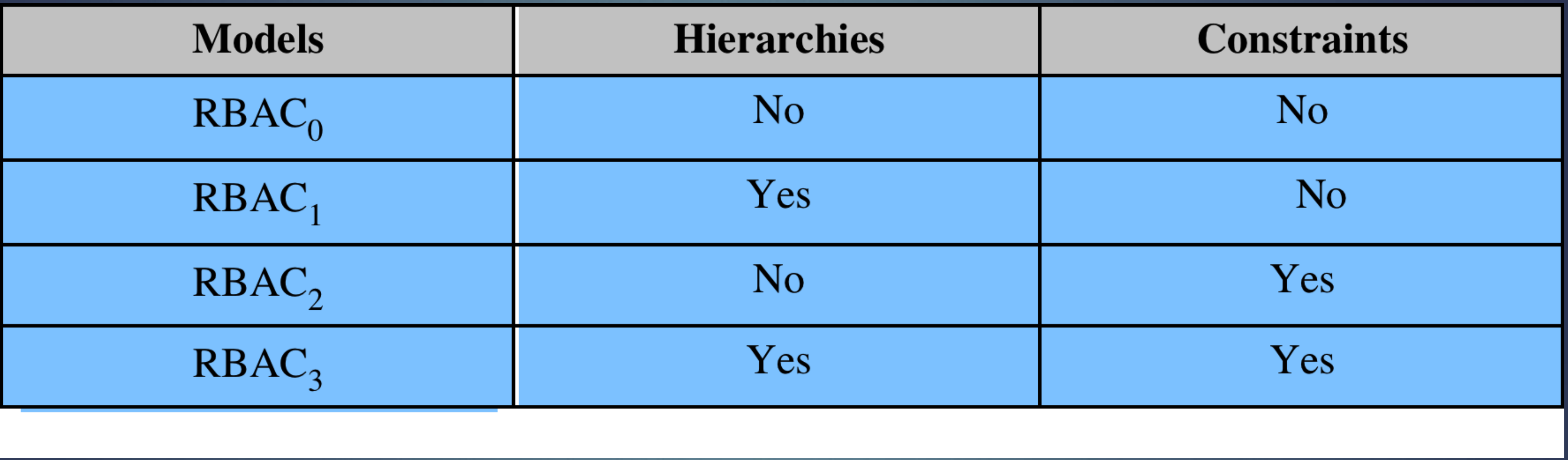
(a) Relationship among RBAC models



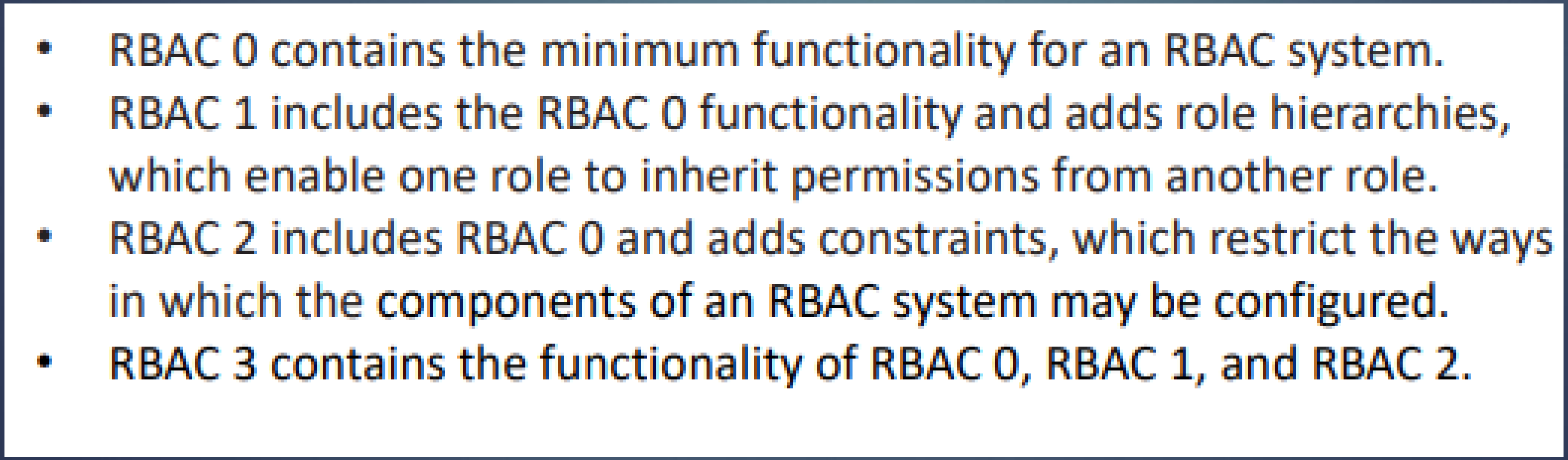
(b) RBAC models

**Figure 4.8 A Family of Role-Based Access Control Models.**

# Table 4.4 Scope RBAC Models



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



- RBAC 0 contains the minimum functionality for an RBAC system.
- RBAC 1 includes the RBAC 0 functionality and adds role hierarchies, which enable one role to inherit permissions from another role.
- RBAC 2 includes RBAC 0 and adds constraints, which restrict the ways in which the components of an RBAC system may be configured.
- RBAC 3 contains the functionality of RBAC 0, RBAC 1, and RBAC 2.

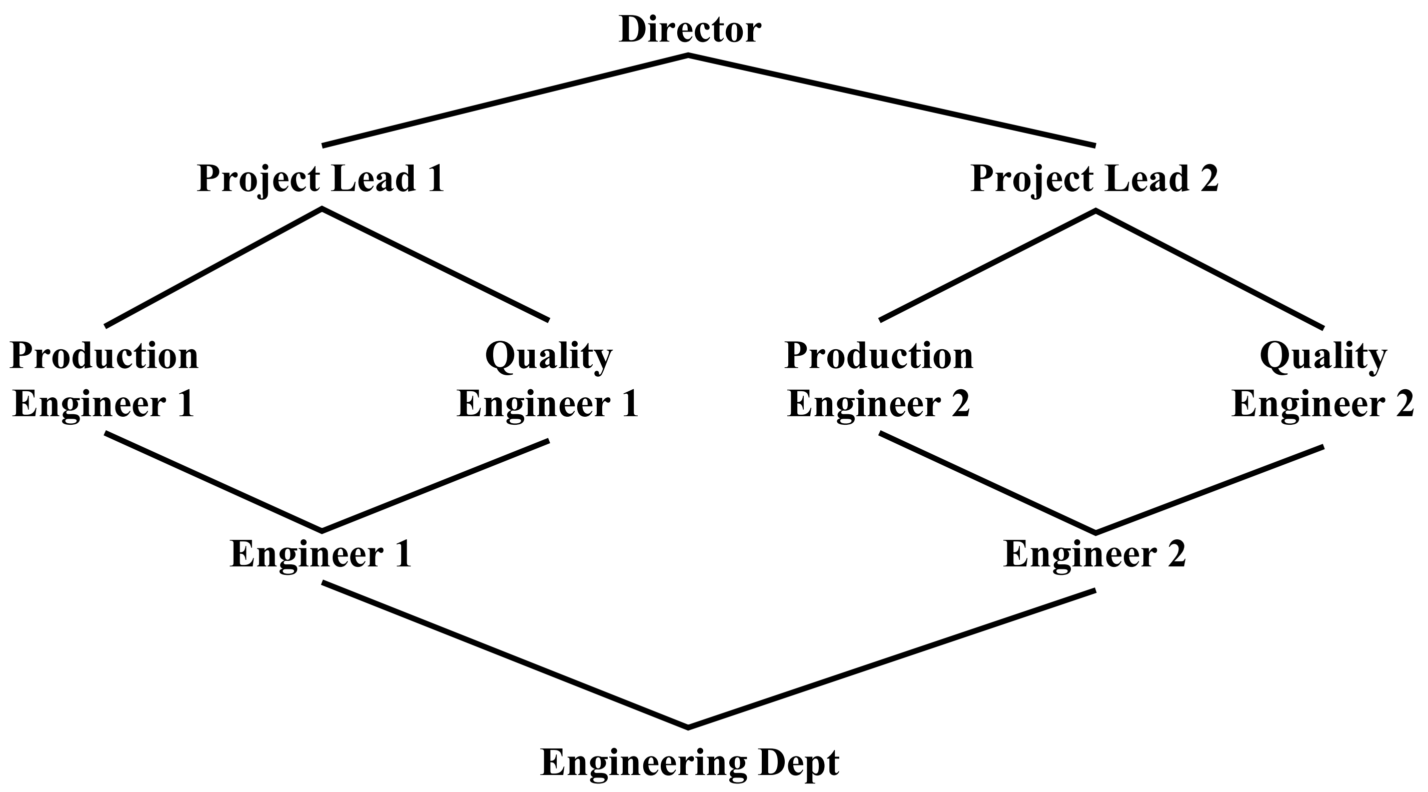


**Figure 4.9 Example of Role Hierarchy**

# 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 | **Cardinality** | **Prerequisite roles** |
|---|---|---|
| • A user can only be assigned to one role in the set (either during a session or statically)<br>• Any permission (access right) can be granted to only one role in the set | • Setting a maximum number with respect to roles | • Dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role |

## Attribute-Based Access Control (ABAC)

Can define authorizations that express conditions on properties of both the resource and the subject

Strength is its flexibility and expressive power

Main obstacle to its adoption in real systems has been concern about the performance impact of evaluating predicates on both resource and user properties for each access

Web services have been pioneering technologies through the introduction of the eXtensible Access Control Markup Language (XAMCL)

There is considerable interest in applying the model to cloud services

# ABAC Model: Attributes

## Subject attributes

- A subject is an active entity(e.g., a user, an application, a process, or a device) that causes information to flow among objects or changes the system state
- Attributes define the identity and characteristics of the subject

## Object attributes

- An object (or resource) is a passive(in the context of the given request) information system-related entity (e.g., devices,files, records, tables, processes, programs, networks, domains) containing or receiving information
- Objects have attributes that can be leverages to make access control decisions

## Environment attributes

- Describe the operational, technical, and even situational environment or context in which the information access occurs
- These attributes have so far been largely ignored in most access control policies

# ABAC

Distinguishable because it controls access to objects by evaluating rules against the attributes of entities, operations, and the environment relevant to a request

Relies upon the evaluation of attributes of the subject, attributes of the object, and a formal relationship or access control rule defining the allowable operations for subject-object attribute combinations in a given environment

Systems are capable of enforcing DAC, RBAC, and MAC concepts

Allows an unlimited number of attributes to be combined to satisfy any access control rule

Figure 4.10 illustrates in a logical architecture the essential components of an ABAC system. An access by a subject to an object proceeds according to the following steps:

1. A subject requests access to an object. This request is routed to an access control mechanism.
2. The access control mechanism is governed by a set of rules (2a) that are defined by a preconfigured access control policy. Based on these rules, the access control mechanism assesses the attributes of the subject (2b), object (2c), and current environmental conditions (2d) to determine authorization.
3. The access control mechanism grants the subject access to the object if access is authorized, and denies access if it is not authorized.

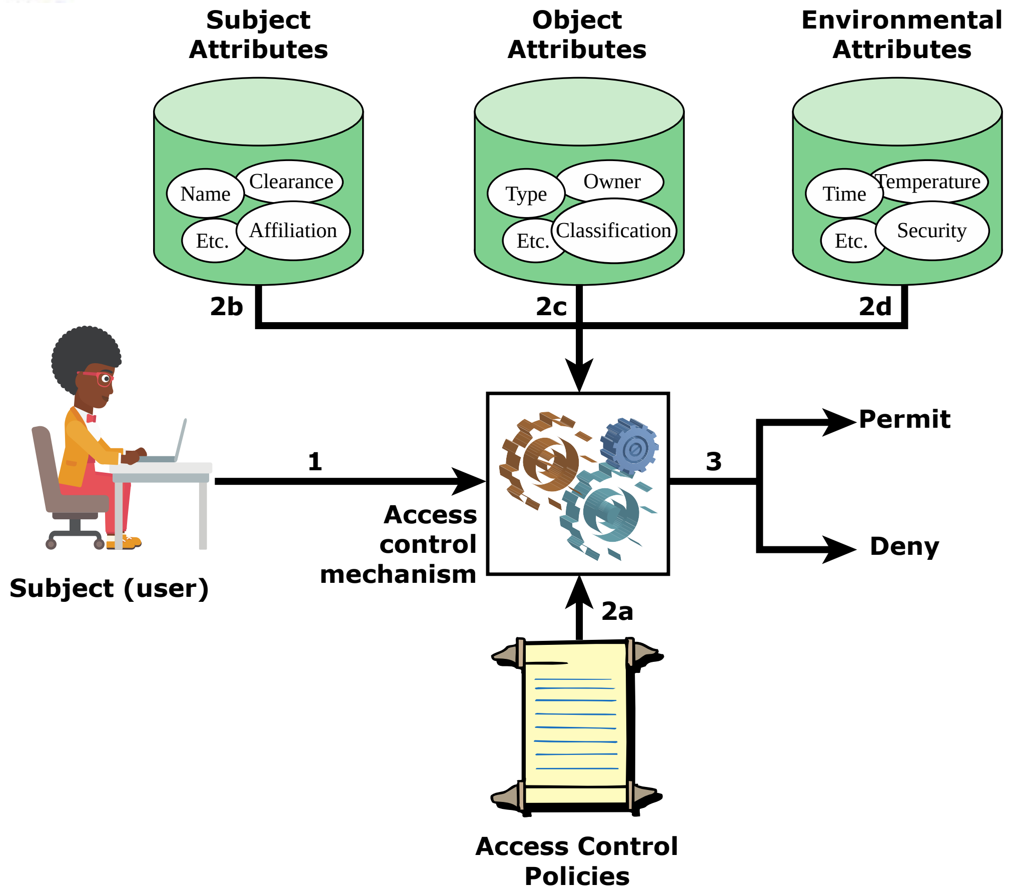


Figure 4.10 ABAC Scenario

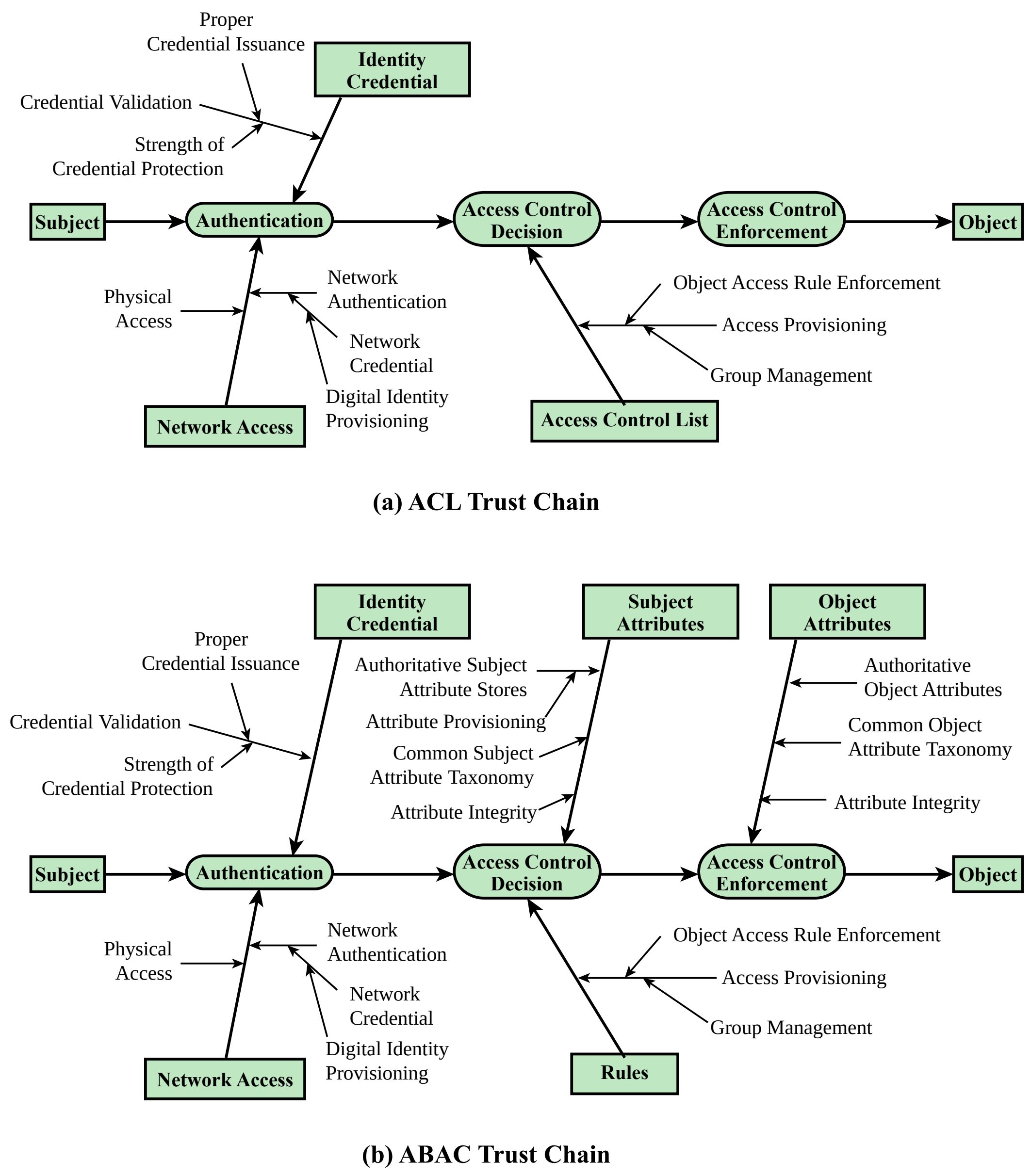


**Figure 4.11 ACL and ABAC Trust Relationships**

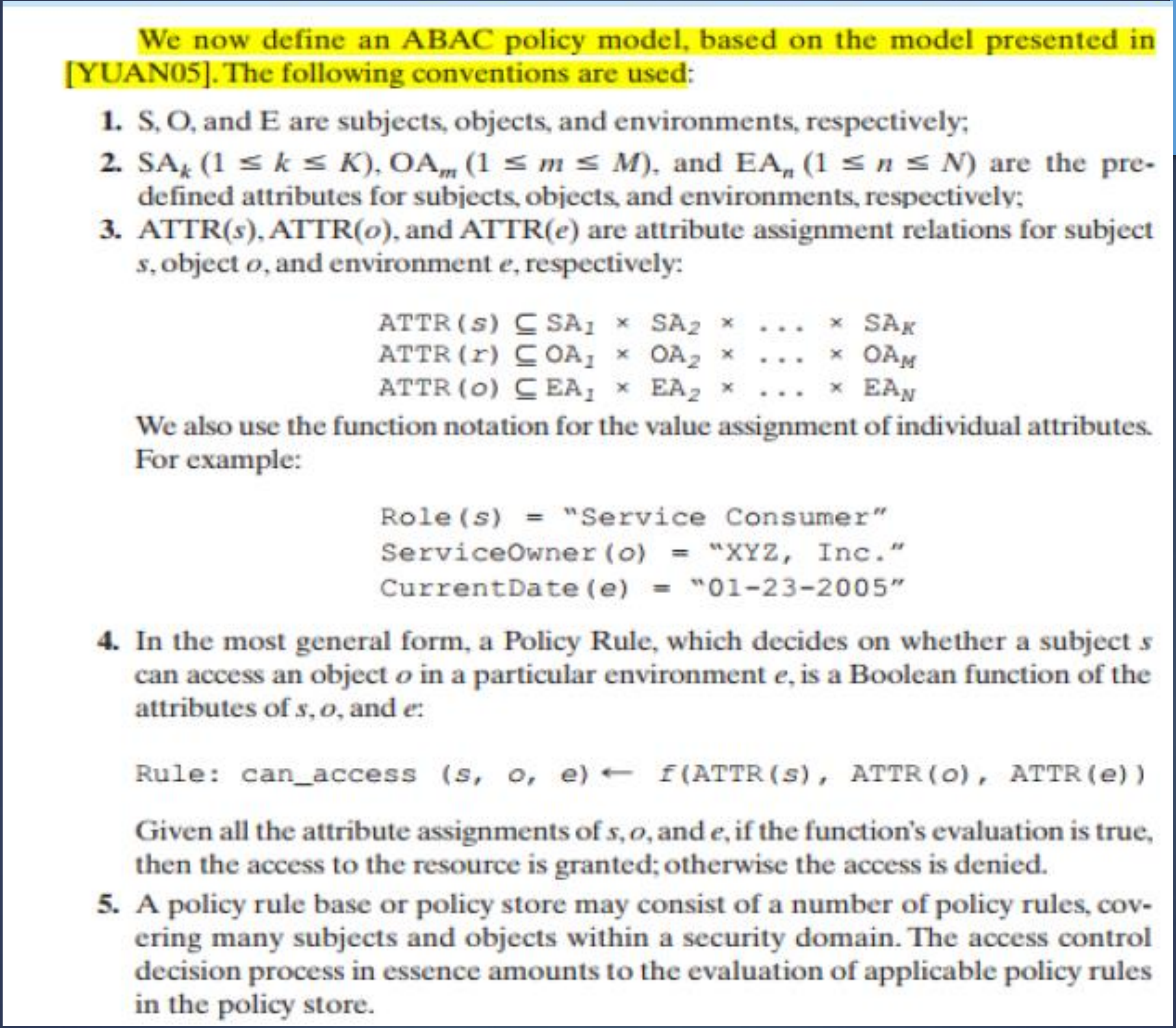
# ABAC Policies

A policy is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions

Typically written from the perspective of the object that needs protecting and the privileges available to subjects

Privileges represent the authorized behavior of a subject and are defined by an authority and embodied in a policy

Other terms commonly used instead of privileges are: rights, authorizations, and entitlements



We now define an ABAC policy model, based on the model presented in [YUAN05]. The following conventions are used:

1. S, O, and E are subjects, objects, and environments, respectively;
2. $SA_k$ ($1 \le k \le K$), $OA_m$ ($1 \le m \le M$), and $EA_n$ ($1 \le n \le N$) 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:

$$\text{ATTR}(s) \subseteq SA_1 \times SA_2 \times \ldots \times SA_K$$
$$\text{ATTR}(r) \subseteq OA_1 \times OA_2 \times \ldots \times OA_M$$
$$\text{ATTR}(o) \subseteq EA_1 \times EA_2 \times \ldots \times EA_N$$

   We also use the function notation for the value assignment of individual attributes. For example:

```
Role(s)  = "Service Consumer"
ServiceOwner(o)  = "XYZ, Inc."
CurrentDate(e)  = "01-23-2005"
```

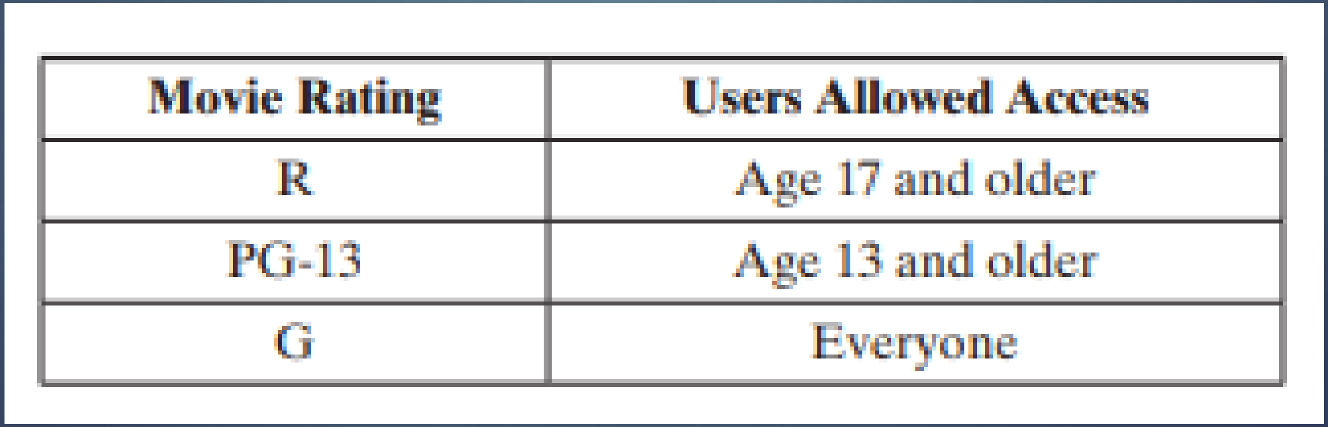
4. In the most general form, a Policy Rule, which decides on whether a subject $s$ can access an object $o$ in a particular environment $e$, is a Boolean function of the attributes of $s$, $o$, and $e$:

```
Rule: can_access (s, o, e) ← f(ATTR(s), ATTR(o), 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.

## Example of an Online Entertainment Store (RBAC VS ABAC)

- The example of an online entertainment store that streams movies to users for a flat monthly fee.
  - **We will use this example to contrast RBAC and ABAC approaches.**
- The store must enforce the following access control policy based on the user's age and the movie's content rating:



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

## Example of an Online Entertainment Store (RBAC VS ABAC)

- **In an RBAC model,** every user would be assigned one of three roles: Adult, Juvenile, or Child, possibly during registration.
- There would be three permissions created: Can view R-rated movies, Can view PG-13-rated movies, and Can view G-rated movies.
- The Adult role gets assigned with all three permissions; the Juvenile role gets Can view PG-13-rated movies and Can view G-rated movies permissions, and the Child role gets the Can view G-rated movies permission only.
- Both the user-to-role and permission-to-role assignments are manual administrative tasks.

## Example of an Online Entertainment Store (RBAC VS ABAC)

- The ABAC approach to this application does not need to explicitly define roles.
- Instead, whether a user **u** can access or view a movie **m** (in a security environment **e** which is ignored here) would be resolved by evaluating a policy rule such as the following:

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

where Age and Rating are the subject attribute and the object attribute, respectively.

- The advantage of the ABAC model shown here is that it eliminates the definition and management of static roles, hence eliminating the need for the administrative tasks for user-to-role assignment and permission-to-role assignment.

## Example of an Online Entertainment Store (RBAC VS ABAC)

- **For the RBAC model,** we would have to double the number of roles, to distinguish each user by age and fee, and we would have to double the number of separate permissions as well.
  - In general, if there are **K** subject attributes and **M** object attributes, and if for each attribute, Range() denotes the range of possible values it can take, then the respective number of roles and permissions required for an **RBAC model are**:

$$\prod_{k=1}^{K} Range\ (SA_k) \quad \text{and} \quad \prod_{m=1}^{M} Range\ (SA_m)$$

- Thus, we can see that as the number of attributes increases to accommodate finer-grained policies, the number of roles and permissions grows exponentially

# Example of an Online Entertainment Store (RBAC VS ABAC)

- In contrast, the ABAC model deals with additional attributes in an efficient way.
- For this example, the policy R1 defined previously still applies.
- We need two new rules:

$$
\begin{aligned}
&\mathtt{R2{:}can\_access}(u,\ m,\ e) \leftarrow \\
&\quad (\mathtt{MembershipType}(u) = \mathtt{Premium}) \vee \\
&\quad (\mathtt{MembershipType}(u) = \mathtt{Regular} \wedge \mathtt{MovieType}(m) = \mathtt{OldRelease}) \\
&\mathtt{R3{:}can\_access}(u,\ m,\ e) \leftarrow \mathtt{R1} \wedge \mathtt{R2}
\end{aligned}
$$

- With the ABAC model,
  - **It is also easy to add environmental attributes.**
- Suppose we wish to add a new policy rule that is expressed in words as follows:
  - **Regular users are allowed to view new releases in promotional periods.**
- This would be difficult to express in an RBAC model.
- In an ABAC model:
  - **We only need to add a conjunctive (AND) rule that checks to see the environmental attribute today's date falls in a promotional period**

# Identity, Credential, and Access Management (ICAM)

- A comprehensive approach to managing and implementing digital identities, credentials, and access control
- Developed by the U.S. government
- Designed to:
  - Create trusted digital identity representations of individuals and nonperson entities (NPEs)
  - Bind those identities to credentials that may serve as a proxy for the individual of NPE in access transactions
    - A credential is an object or data structure that authoritatively binds an identity to a token possessed and controlled by a subscriber
  - Use the credentials to provide authorized access to an agency's resources

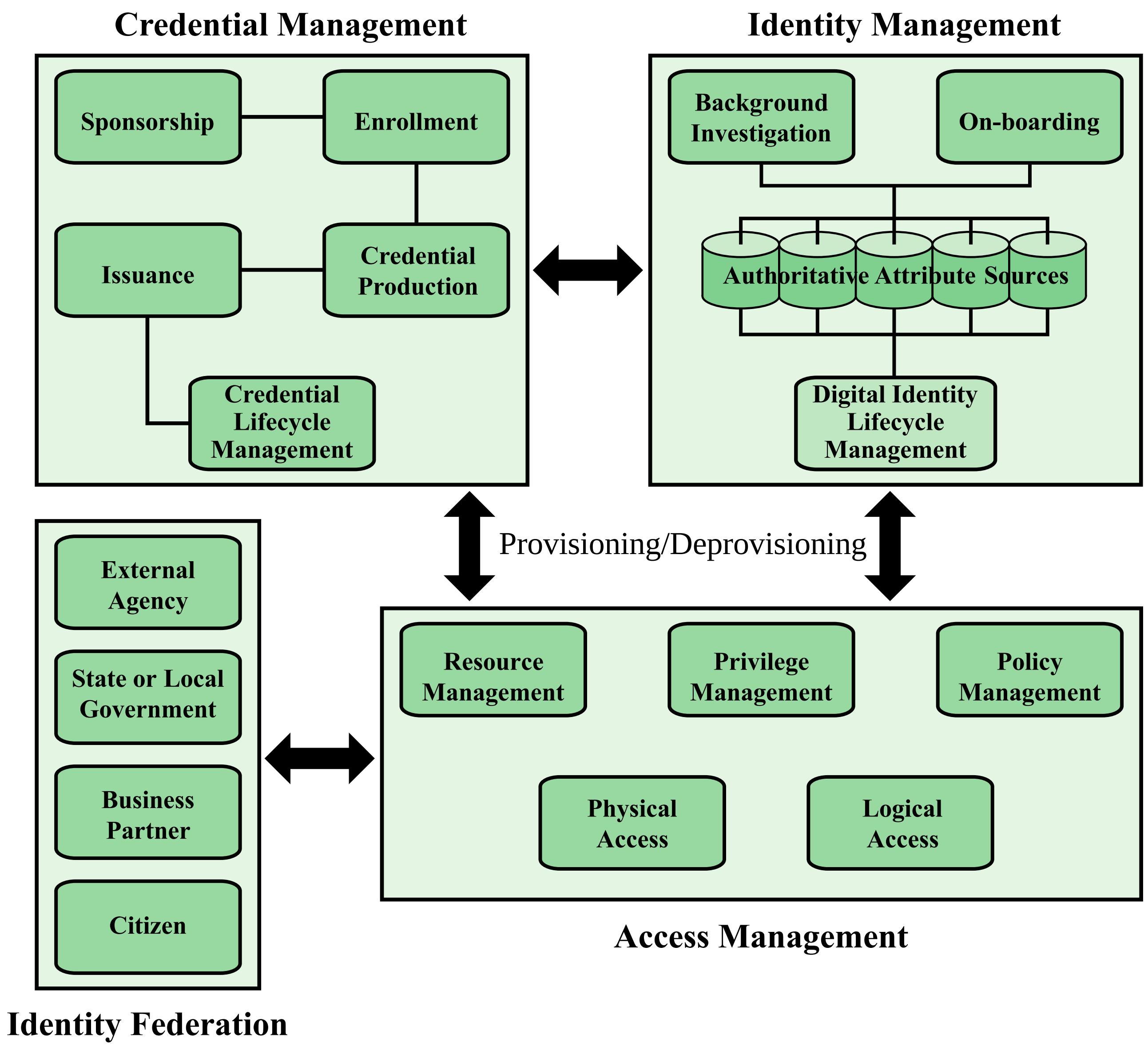


**Figure 4.12 Identity, Credential, and Access Management (ICAM)**