11464: INFORMATION SYSTEMS SECURITY

12/5/2021

Chapter 8: Access Control

2

Access Control

By

Mustafa Al-Fayoumi

Outline

- **Access Control Principles**
  - Access Control Context
  - Access Control Policies
- **Subjects, Objects, and Access Rights**
- **Discretionary Access Control**
  - An Access Control Model
  - Protection Domains
- **Example: Unix File Access Control**
  - Traditional UNIX File Access Control
  - Access Control Lists in UNIX
- **Role-Based Access Control**
  - RBAC Reference Models
- **Attribute-Based Access Control**
  - Attributes
  - ABAC Logical Architecture
  - ABAC Policies

Introduction

- Two definitions of access control are useful in understanding its scope.
  - NIST 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.
  - RFC 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.
- We can view access control as the central element of computer security. The principal objectives of computer security are to prevent unauthorized users from gaining access to resources, to prevent legitimate users from accessing resources in an unauthorized manner, and to enable legitimate users to access resources in an authorized manner.

Access Control Principles

- 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.
- Figure 4.1 shows a broader context of access control. In addition to access control, this 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 (acceptability) 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 Principles

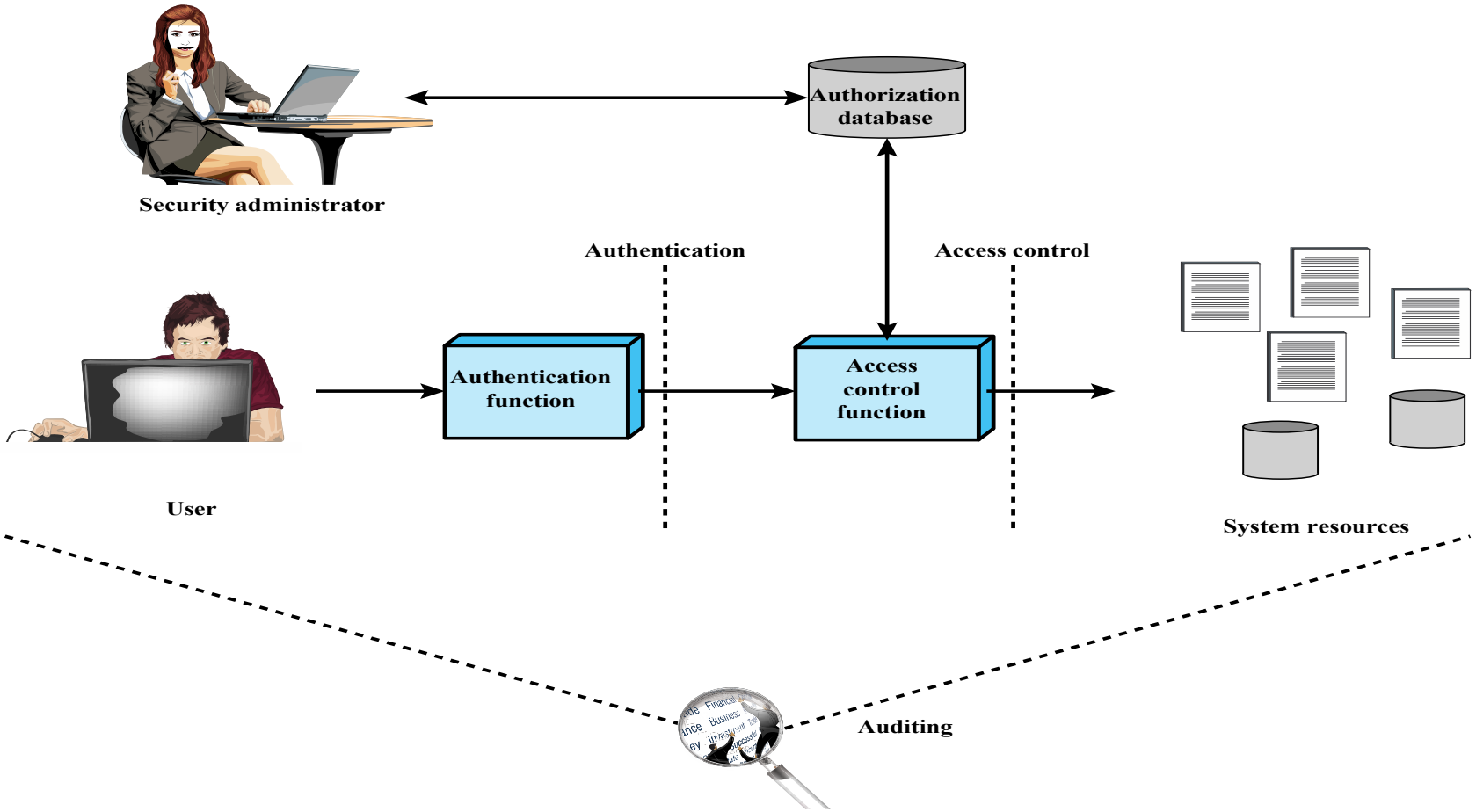


Figure 4.1 Relationship Among Access Control and Other Security Functions

Access Control Principles

- 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.
- The system must first **authenticate** an entity seeking **access**.
  - Typically, the authentication function determines whether the user is permitted to access the system at all.
    - Then the access control function determines if the specific requested access by this user is permitted.
- A security administrator maintains an authorization database that specifies what type of access to which resources is allowed for this user.
  - The access control function consults this database to determine whether to grant access.
- An **auditing** function monitors and keeps a record of user accesses to **system resources**

Access Control Requirements

- The following concepts features of an access control system:
  - reliable input
  - fine and coarse specifications
  - least privilege
  - separation of duty
  - open and closed policies
  - policy combinations, conflict resolution
  - administrative policies

Access Control Requirements

- The following concepts features of an access control system:
  - **Reliable input:** it assumes that a user is authentic; thus, an authentication mechanism is needed as a front end to an access control system. Other inputs to the access control system must also be reliable.
  - **Support for fine and coarse specifications:** fine-grained specifications allow access regulated at the level of individual fields / records in files; and each individual access by a user rather than a sequence of accesses. System administrators should also be able to choose coarse-grain specification for some classes of resource access.
  - **Least privilege:** it should be implemented so that each system entity is granted the minimum system resources and authorizations needed to do its work. This principle tends to limit damage that can be caused by an accident, error, or unauthorized act.
  - **Separation of duty:** should divide steps in a system function among different individuals, so as to keep a single individual from subverting the process.
  - **Open and closed policies:** a closed policy only allows accesses that are specifically authorized; an open policy allows all accesses except those expressly prohibited.
  - **Policy combinations and conflict resolution:** may apply multiple policies to a given class of resources, and need a procedure to resolves conflicts between policies.
  - **Administrative policies:** to specify who can add, delete, or modify authorization rules, and also need access control and other control mechanisms to enforce these administrative policies.

Access Control Policies

- An access control policy, embodied in an authorization database, dictates **what types** of access are permitted, **what circumstances**, and **by whom**.
- Access control policies are generally grouped into the following categories:
  - **Discretionary access control (DAC).**
  - **Mandatory access control (MAC).**
  - **Role-based access control (RBAC).**
  - **Attribute-based access control (ABAC).**

Access Control Policies

- Access control policies are generally grouped into the following categories:
  - 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 (which indicate how sensitive or critical system resources are) with security clearances (which indicate system entities are eligible to access certain resources).
  - 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.

Access Control Policies

- DAC is the traditional method of implementing access control.
- MAC is a concept that evolved out of requirements for military information security and is best covered in the context of trusted systems.
- both RBAC and ABAC have become increasingly popular, and will be examined later on.
- These four policies are not mutually exclusive. An access control mechanism can employ 2 or even 3 of these policies to cover different classes of system resources.

Basic Element of Access Control System: Subjects, Objects, and Access Rights

- The basic elements of access control are: subject, object, and access right.
- **A subject** is an entity capable of accessing objects.
  - Often subject is a software process.
  - Any user or application actually gains access to an object by means of a process that represents that user or application. The process takes on the attributes of the user, such as access rights.
  - Classes of subject, with different access rights for each class: Owner, Group, and World

Subjects, Objects, and Access Rights

- Classes of 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.

Subjects, Objects, and Access Rights

- An **object** is a resource to which access is controlled.
  - In general, an object is an entity used to contain and/or receive information.
  - E.g. 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.
- The number and types of objects to be protected by an access control system depends on the environment in which access control operates and the desired tradeoff between security on the one hand and complexity, processing burden (load), and ease of use on the other hand.

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.

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

Discretionary Access Control (DAC)

- **A discretionary access control** scheme: an entity may be granted access rights that permit the entity, by its own decision, if they choose so, to enable another entity to access some resource.
- A general approach to DAC, as exercised by an operating system or a database management system. (Common access control scheme in Oss and DB mang. System)
- Often provided using an **access matrix, (specifies access rights of subject on object).**
  - 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**

Discretionary Access Control (DAC)

- Figure 4.2a, is a simple example of an access matrix. Thus, user A owns files 1 and 3 and has read and write access rights to those files. User B has read access rights to file 1, and so on.
- In practice, an access matrix is usually sparse and is implemented either:
  - **Access control lists** (ACLs): For each object, list subject and their access right (Figure 4.2b)- the access matrix decomposed by **columns** yielding **access control lists** (ACLs).
  - **Capability tickets (Lists):** For each object, list object and the rights the subject have on that object. Decomposition the matrix by **rows** yields **capability tickets** (Figure 4.2c).
- Alternative implementation: authorization table listing subject, access mode and object; easily implemented in database.

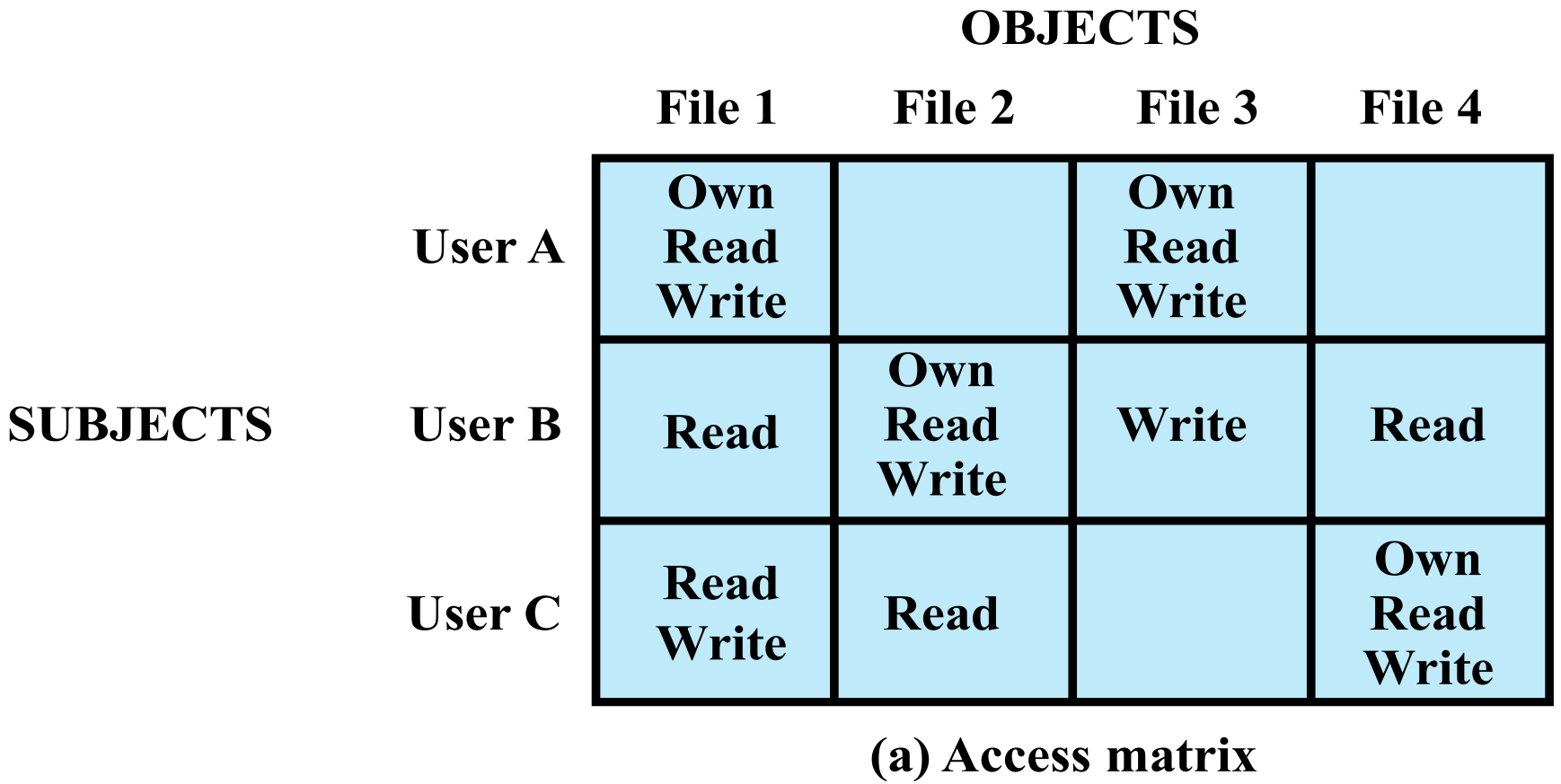
ACL

- For each object, an ACL lists users and their permitted access rights.
- The ACL may contain a default, or public, entry.
  - This allows users that are not explicitly listed as having special rights to have a default set of rights.
  - The default set of rights should always follow the rule of least privilege or read-only access, whichever is applicable.
  - Elements of the list may include individual users as well as groups of users.
- When it is **desired** to determine which subjects have which access rights to a particular resource, ACLs are convenient, because each ACL provides the information for a **given resource**.
  - However, this data structure is not convenient for determining the access rights available to a specific user.

Capability tickets (Lists)

- A capability ticket specifies authorized objects and operations for a particular user.
- Each user has a number of tickets and may be authorized to loan or give them to others.
- Because tickets may be distributed around the system, they present a **greater security problem** than access control lists.
- The **integrity** of the ticket must be protected, and guaranteed (usually by the operating system). In particular, the ticket must be unforgeable.
  - One way to accomplish this is to have the on behalf of users.
    - **Operating system hold all tickets.** These tickets would have to be held in a region of memory inaccessible to users.
  - Another alternative is to include an unforgeable token in the capability.
    - This could be a large random password, or a cryptographic message authentication code. This value is verified by the relevant resource whenever access is requested.
  - This form of capability ticket is appropriate for use in a distributed environment, when the security of its contents cannot be guaranteed

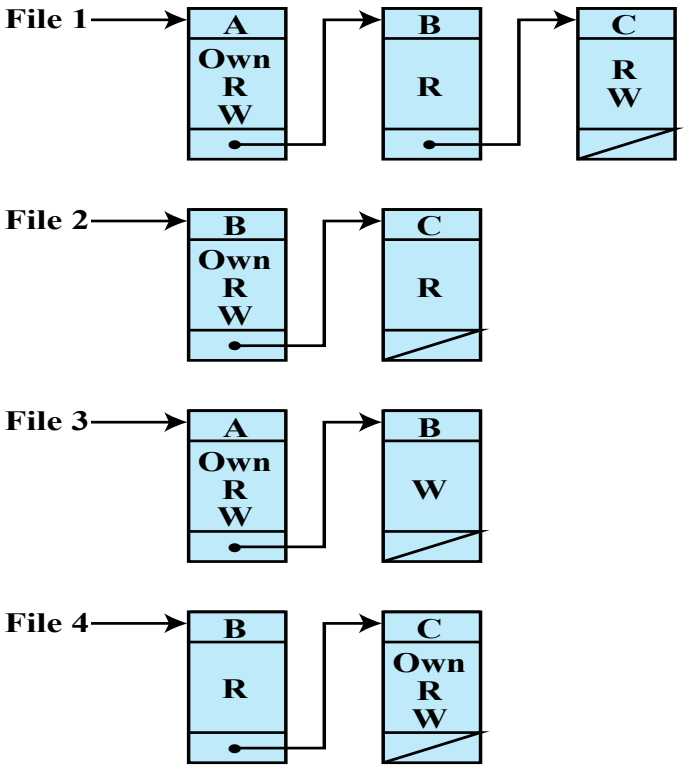
Discretionary Access Control (DAC)



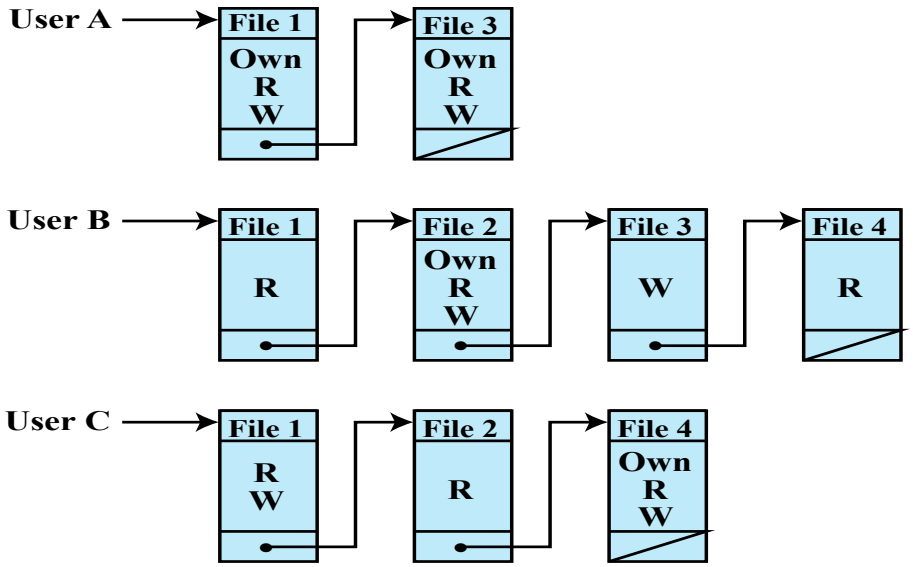
SUBJECTS	OBJECTS: File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

(a) Access matrix

Discretionary Access Control (DAC)



(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

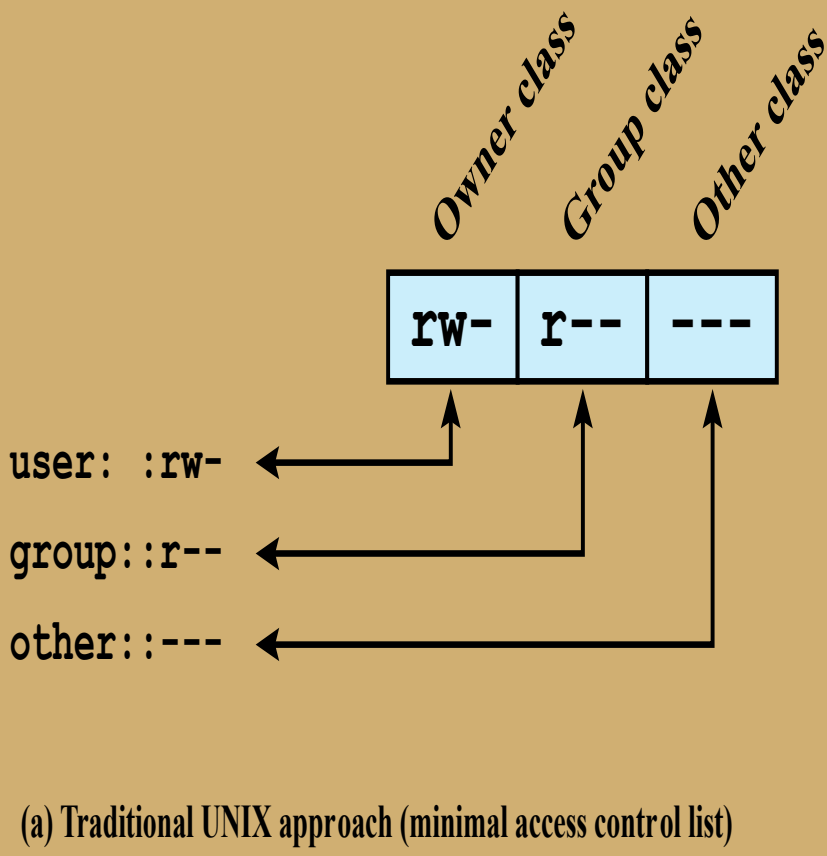
Table 4.1

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

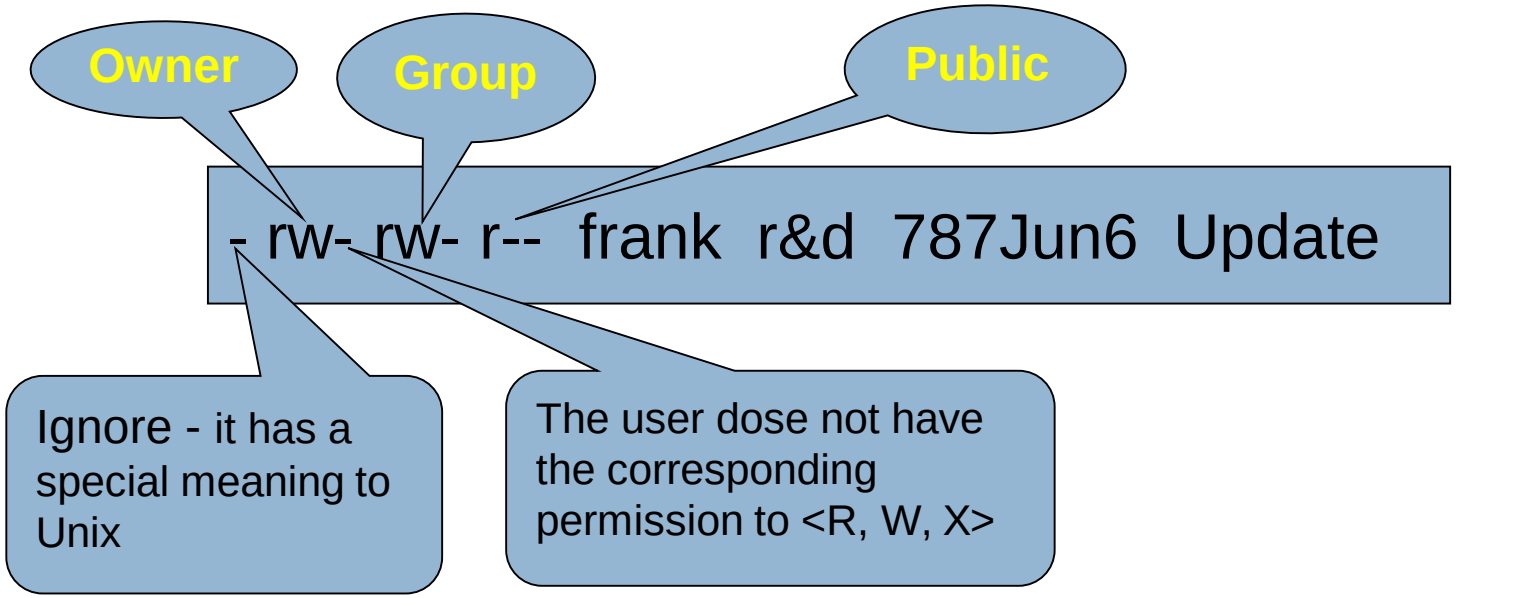
UNIX File Access Control

- Unique user identification number (**user ID**)
- Member of a primary group identified by a group ID
- Belongs to a specific group
- 12 protection bits
  - Nine of the protection bits specify read, write, and execute permission for the owner of the file, other members of the group to which this file belongs, and all other users.
  - The remaining three bits define special additional behavior for files or directories. Two of these are the "set user ID" (SetUID) and "set group ID" (SetGID) permissions.
- The owner ID, group ID, and protection bits are part of the file's inode



(a) Traditional UNIX approach (minimal access control list)

Example : in UNIX



1. Owner (frank): can read and write the update file (rw-)
2. Member of the file group (r&d) can R and W the file (rw-)
3. The rest of the world can R only (r--)

Mandatory access control (MAC)

- **Mandatory access control (MAC)**
  - Based on **multilevel security** (MLS) : top secret, secret, confidental, unclassified.
  - The system determine access rules based on security levels of the people in your system.
- **MAC** use sensitivity labels (**Security Clearance**) to determine who can access what information in the system. (Subject has security clearance of a given level)
- Objects has Security Classification of a given level. E.g. data, files.
  - Labeling and MAC implement a multi level security policy. Dividing data and users.

- **Two required properties for confidentiality**
  - No read up: Subject can only read an object of less or equal security level
  - No write down: Subject can only write into object of greater or equal security level
- Clearance and classification is determine by administrator; users can't override security policy
- **Bell-LaPadula model** formally defines multilevel security and MAC.

Sensitivity labels

- **Classification.**
  - **Done by the administrator or the security officer.**
    - **Top secret**
    - **Secret**
    - **Confidential**
    - **Un classified**
- **Set of categories**
  - **Are non hierarchical such as accounting, marketing , sales.**

Clearance example [Bell and LaPadula]

- Secret [accounting, marketing, sales].
- **Idea :**
  - Even some one who has the highest classification is not automatically cleared to see all information at that level.
- **Rules**
  - **Read.**
    - If **class (subject)** $\geq$ **class (object) and category (object)** $\subseteq$ **category (subject)**Then you can read else can not read.

Clearance example

- **Write**
  - If **class (subject) ≤ class (object) and category (subject) ⊆category (object)** Then you can write else can not write.

Clearance example

Example 1

If file has Secret [V,A] and user has Confidential [V,A] can he read or write the file ?

Answer : Can not read. Can write

Example 2

If file has Secret [V,A] and user has Top secret [V] can he read or write the file ?

Answer : Can not read. Can not write.

Clearance example

Example 3

If file has Top Secret [A,B,C] and user has Secret [A] can he read or write the file ?

Answer :

Can not read. Can write.

Example 4

If file has Top Secret [A,B,C] and user has Top Secret [A] can he read or write the file ?

Answer :

Can not read. Can write.

Role Based Access Control

- **Traditional DAC** systems define the access rights of individual users and groups of users.
- In contrast, **RBAC** is based on the **roles** that users assume in a system rather than the user's identity. (RBAC: users are assigned to roles; access rights are assigned to roles)
- Typically, RBAC models define a role as a job function and positions within an organization. (e.g. senior financial analyst in a bank, doctor in a hospital).
- **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.
- Sessions are temporary assignments of user to role(s)
- Access control matrix can map users to roles and roles to objects

Role Based Access Control

- The **relationship** of **users** to **roles** is **many to many**, as is the relationship of roles to resources, or system objects (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.

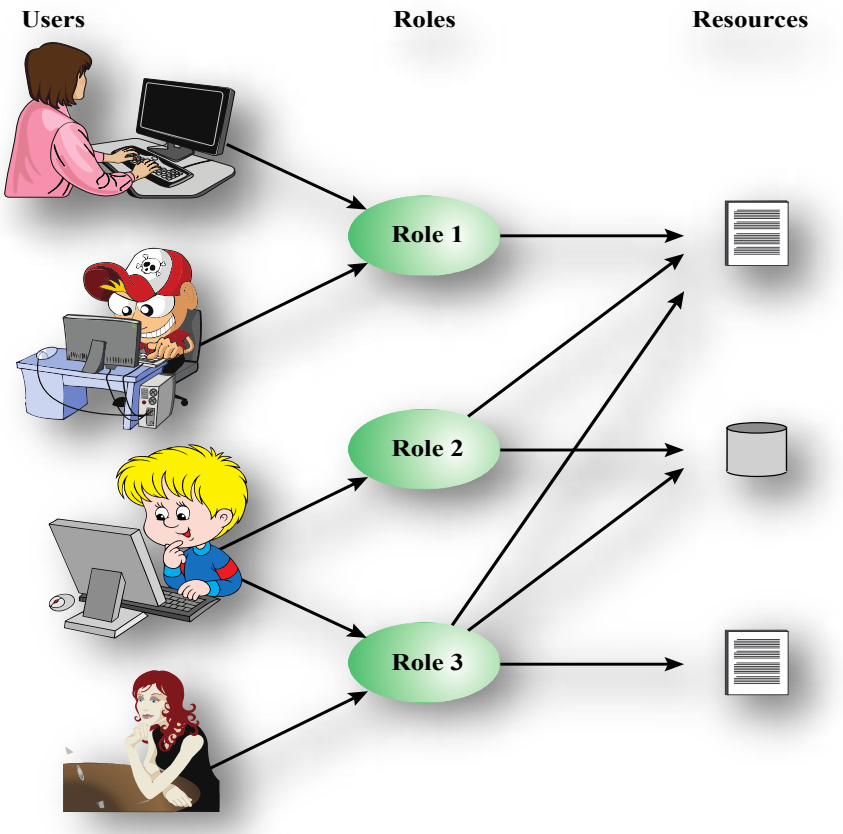
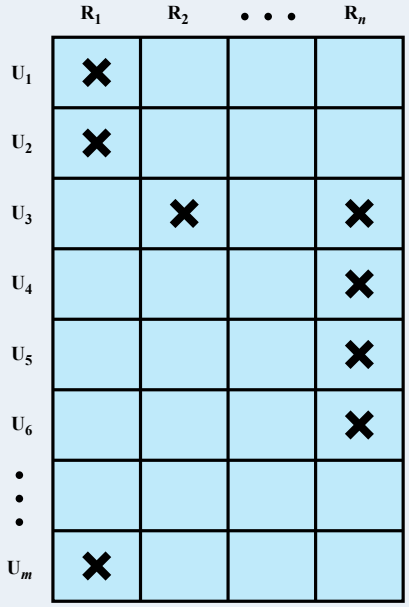


Figure 4.6 Users, Roles, and Resources

Role Based Access Control

- 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 that a single user may be assigned multiple roles (more than one mark in a row) and that multiple users may be assigned to a single role (more than one mark in a column).
- 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.
- The entries are the specific access rights enjoyed by the roles. Note that a role can be treated as an object, allowing the definition of role hierarchies.
- RBAC lends itself to an effective implementation of the principle of least privilege. Each role should contain the minimum set of access rights needed for that role.



	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 4.7 Access Control Matrix Representation of RBAC

A user is assigned to a role that enables him or her to perform only what is required for that role. Multiple users assigned to the same role, enjoy the same minimal set of access rights.

RBAC Reference Model

- RBAC consists of four models that are related to each other as shown in Figure 4.8a. and Table 4.3.
  - ***Base Model—RBAC0***
  - ***Role Hierarchies—RBAC1***
  - ***Constraints—RBAC2***
  - **RBAC3 Consolidated (unified) model**

Scope RBAC Models

- There are four RBAC models, related to each other as shown in the figure and table 4.3

$RBAC_0$ contains the minimum functionality for an RBAC system.

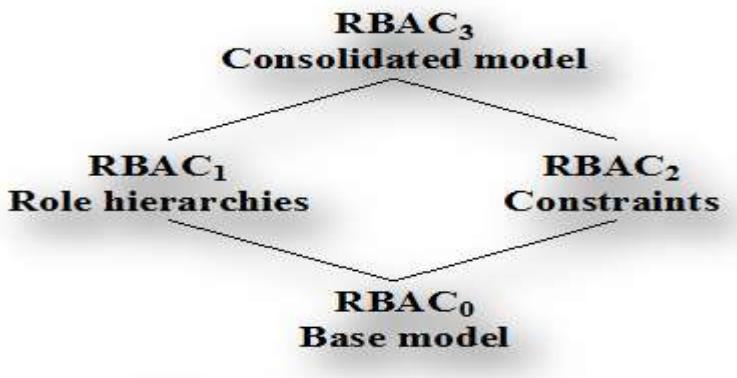
$RBAC_1$ includes the $RBAC_0$ functionality and adds role hierarchies, which enable one role to inherit permissions from another role.

$RBAC_2$ includes $RBAC_0$ and adds constraints which restrict the ways in which the components of a RBAC system may be configured.

$RBAC_3$ contains the functionality of $RBAC_0$, $RBAC_1$, and $RBAC_2$.

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

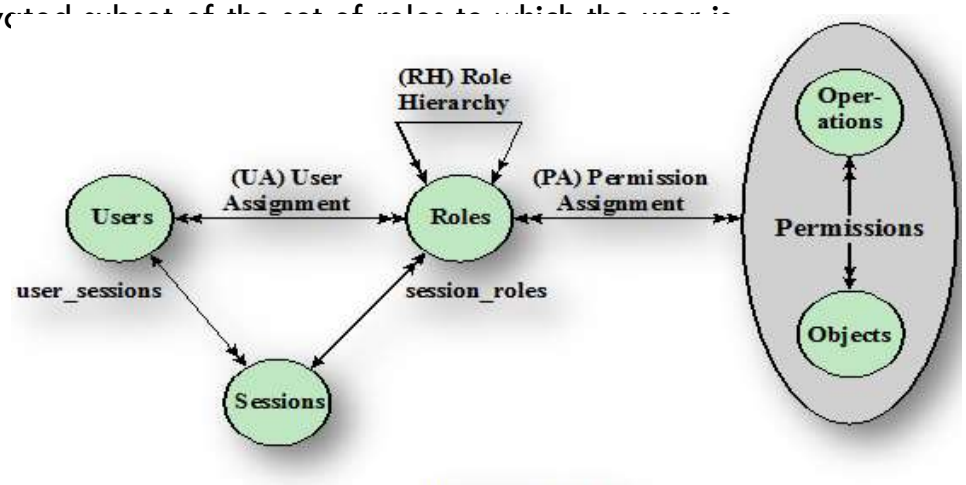


(a) Relationship among RBAC models

Scope RBAC Models

- **$RBAC_0$** contains the four types of entities in an $RBAC_0$ system:
- **User**: An individual that has access to this computer system. Each individual has an associated user ID.
- **Role**: A named job function within the organization that controls this computer system. Typically, associated with each role is a description of 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.

- **An arrow : relationships, or mappings.**
- **There is a many-to-many relationship between users and roles and also between roles and permissions.**
- **A session is used to define a temporary one-to-many relationship between a user and one or more of the roles to which the user has been assigned e**



(b) RBAC models

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

For Example: The user establishes a session with only the roles needed for a particular task; this is an example of the concept of least privilege.

Scope RBAC Models

Role **hierarchies** provide a means of reflecting the hierarchical structure of roles in an organization. Typically, job functions with greater responsibility have greater authority to access resources.
A subordinate job function may have a subset of the access rights of the superior job function. Role hierarchies make use of the concept of inheritance to enable one role to implicitly include access rights associated with a subordinate role.

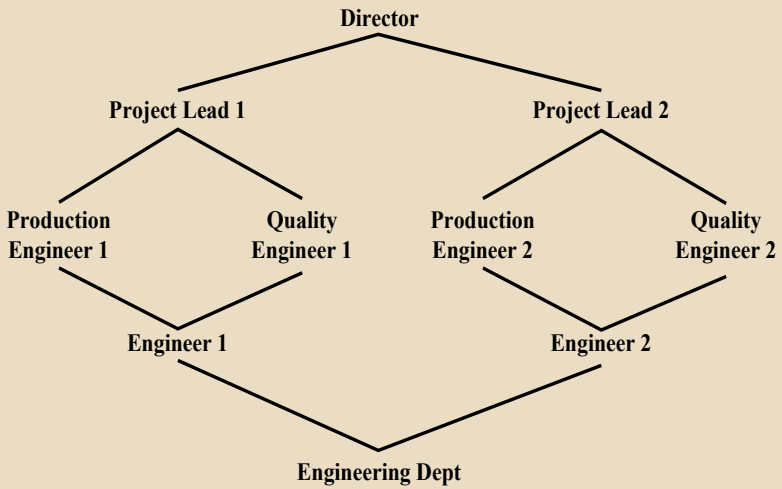


Figure 4.9 Example of Role Hierarchy

Example: of role hierarchy: By convention, subordinate roles are lower in the diagram. A line between two roles implies that the upper role includes all of the access rights of the lower role, as well as other access rights not available to the lower role. One role can inherit access rights from multiple subordinate roles.

Constraints - RBAC

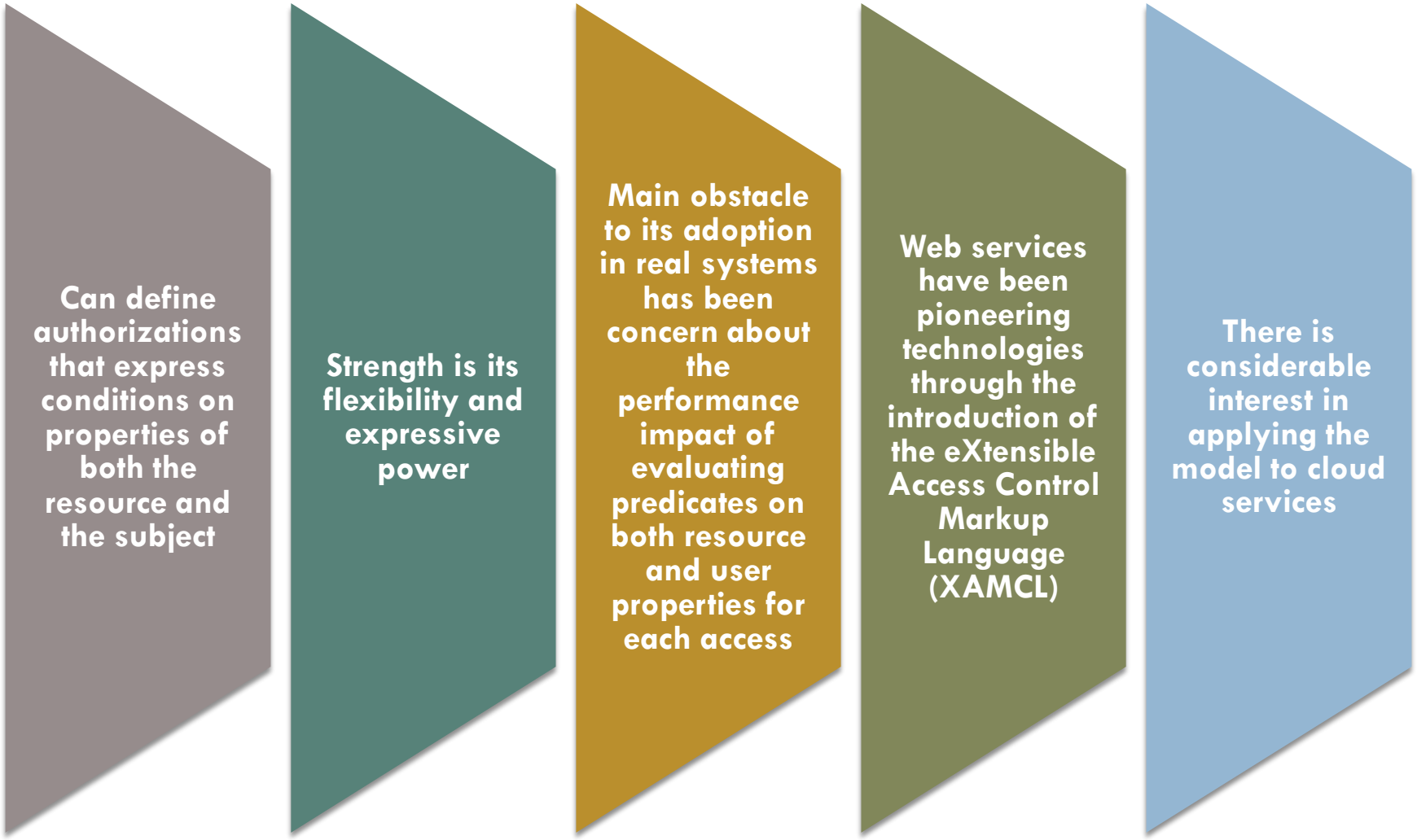
- Provide a means of adapting RBAC to the specifics of administrative and security policies of an organization
- A defined relationship among roles or a condition related to roles
- Types:

Mutually exclusive roles	Cardinality	Prerequisite roles
• A user can only be assigned to one role in the set (either during the session or staticaly) • Any permission (access right) can be granted to only one role in the set	• Setting a maximum number with respect to roles	• Dictates (instruct, order) 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)

- A relatively recent development in access control technology is the attribute-based access control (ABAC) model.
- An ABAC model can define authorizations that express conditions on properties of both the resource and the subject.
- For example, consider a configuration in which each resource has an attribute that identifies the subject that created the resource. Then, a single access rule can specify the ownership privilege for all the creators of every resource. The strength of the ABAC approach is its flexibility and expressive power.
- The 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. However, for applications such as cooperating Web services and cloud computing, this increased performance cost is less noticeable because there is already a relatively high performance cost for each access.
- Thus, Web services have been pioneering technologies for implementing ABAC models, especially through the introduction of the eXtensible Access Control Markup Language.

Attribute Based Access Control (ABAC)



Attribute Based Access Control (ABAC)

- There are three key elements to an ABAC model:
  - Attributes, which are defined for entities in a configuration;
  - A policy model, which defines the ABAC policies;
  - Architecture model, which applies to policies that enforce access control. We examine these elements in turn.

ABAC Model: Attributes

Subject attributes

- A subject is an active entity 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 information system-related entity 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

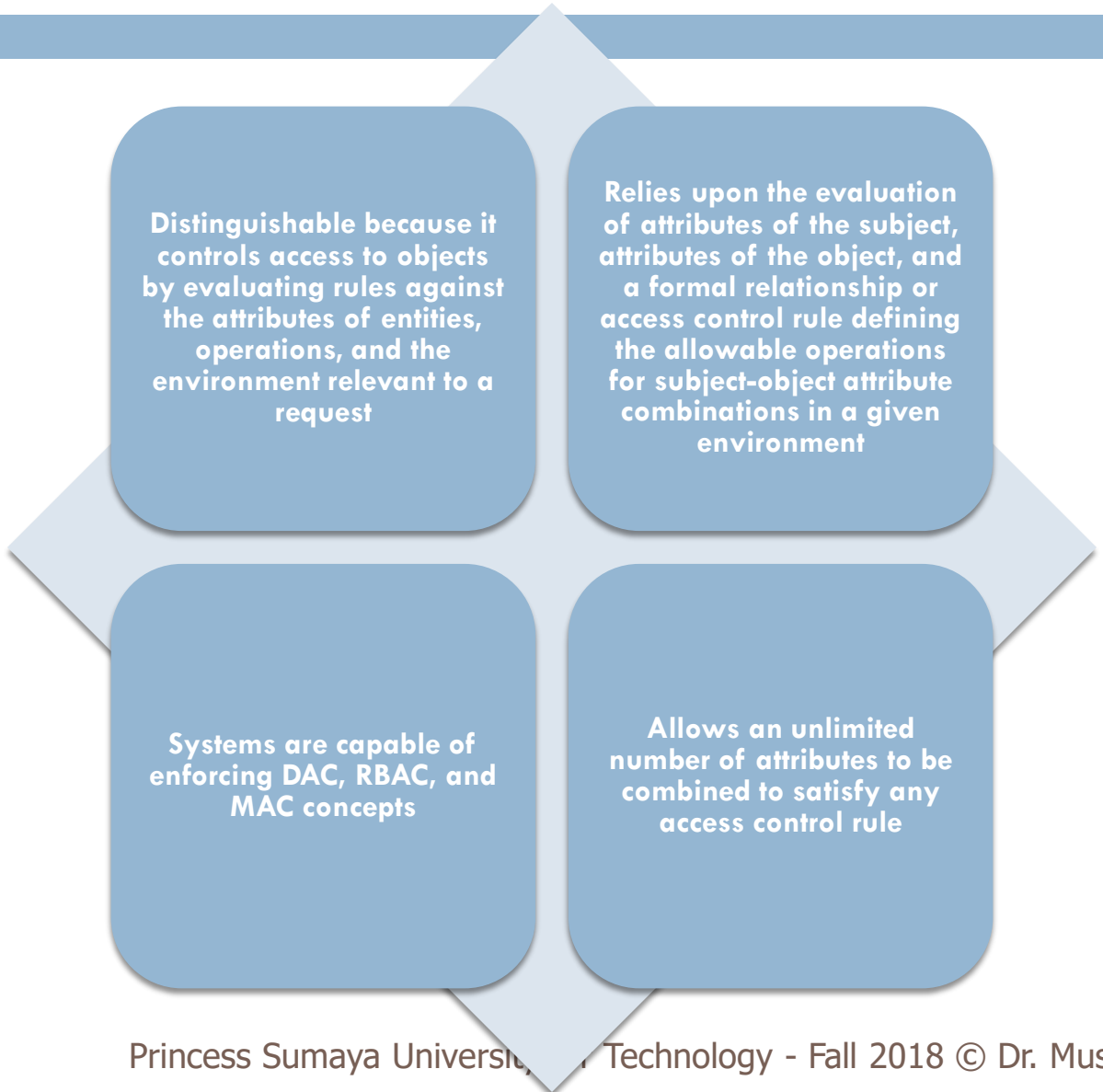
ABAC Model: Attribute

- **Attributes:** are characteristics that define specific aspects of the subject, object, environment conditions, and/or requested operations that are predefined and preassigned by an authority.
- Attributes contain information that indicates the class of information given by the attribute, a name, and a value (e.g., Class=HospitalRecordsAccess, Name=PatientInformationAccess, Value=MFBusinessHoursOnly).
- The following are the three types of attributes in the ABAC model:
  - **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. Each subject has associated attributes that define the identity and characteristics of the subject. Such attributes may include the subject's identifier, name, organization, job title, and so on. A subject's role can also be viewed as an attribute.
  - **Object attributes:** An object, also referred to as a **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. As with subjects, objects have attributes that can be leveraged to make access control decisions.

ABAC Model: Attribute

- The following are the three types of attributes in the ABAC model:
  - **Object attributes:** A Microsoft Word document, for example, may have attributes such as title, subject, date, and author. Object attributes can often be extracted from the metadata of the object. In particular, a variety of Web service metadata attributes may be relevant for access control purposes, such as ownership, service taxonomy, or even Quality of Service (QoS) attributes.
  - **Environment attributes:** These attributes have so far been largely ignored in most access control policies. They describe the operational, technical, and even situational environment or context in which the information access occurs. For example, attributes, such as current date and time, the current virus/hacker activities, and the network's security level (e.g., Internet vs. intranet), are not associated with a particular subject nor a resource, but may nonetheless be relevant in applying an access control policy.

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

ABAC Logical Architecture

- **Essential components of an ABAC system.**
- An access by a subject to an object proceeds as follows:

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 pre-configured 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.

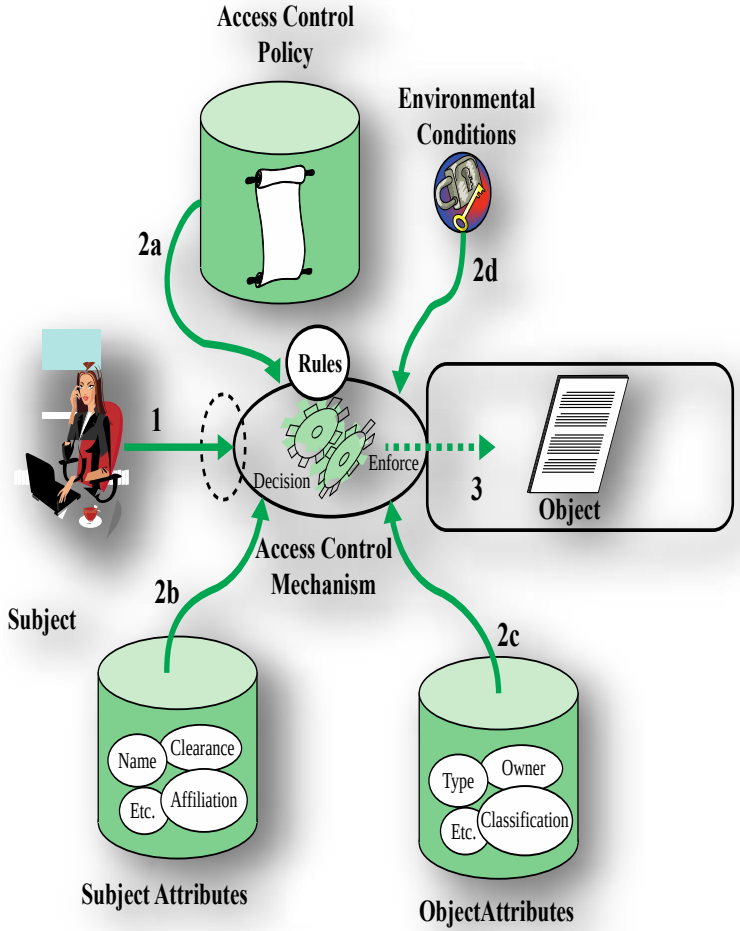


Figure 4.10 Simple ABAC Scenario

ABAC Logical Architecture

- Fig 4.11, is taken from NIST SP 800-162
- Compares the scope of an ABAC model with DAC (using ACLs).
- It illustrates the relative complexity and clarifies the trust requirements of the two models.
- A comparison of representative **trust relationships** (indicated by arrowed lines) for ACL use and ABAC use shows that there are many more **complex** trust relationships required for ABAC to work properly.
- with ACLs the **root** of trust is with the **object owner**, who ultimately enforces the object access rules by provisioning access to the object through addition of a user to an ACL. In ABAC, the root of trust is derived from **many sources** of which the object owner has no control, such as Subject Attribute Authorities, Policy Developers, and Credential Issuers

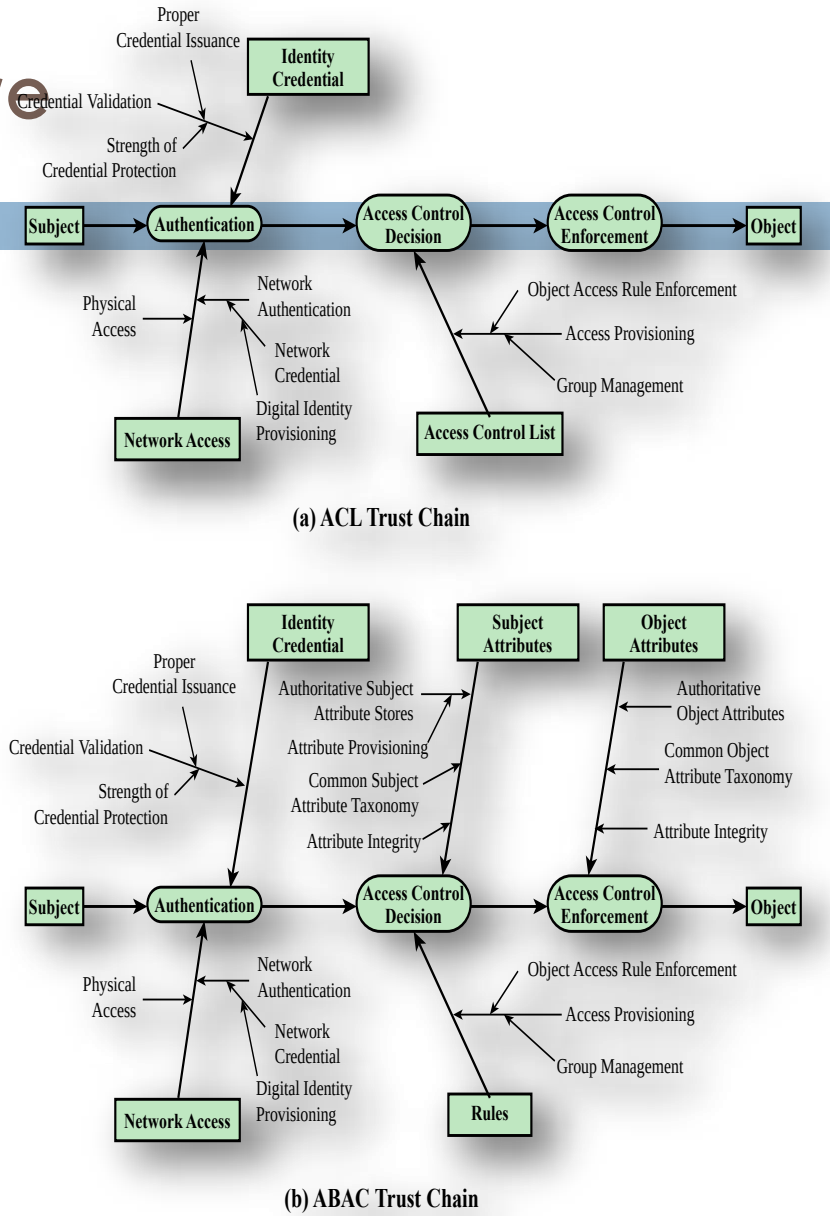


Figure 4.11 ACL and ABAC Trust Relationships

ABAC Logical Architecture

- 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

Summary

- Access control principles
  - Access control context
  - Access control policies
- Subjects, objects, and access rights
- Discretionary access control
  - Access control model
  - Protection domains
- UNIX file access control
  - Traditional UNIX file access control
  - Access control lists in UNIX
- Role-based access control
  - RBAC reference models
- Attribute-based access control
  - Attributes
  - ABAC logical architecture
  - ABAC policies