ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Chapter 4
Access Control

Book Reading: Computer Security Principles and Practice (3ed), 2015, p.134-172

LEARNING OBJECTIVES

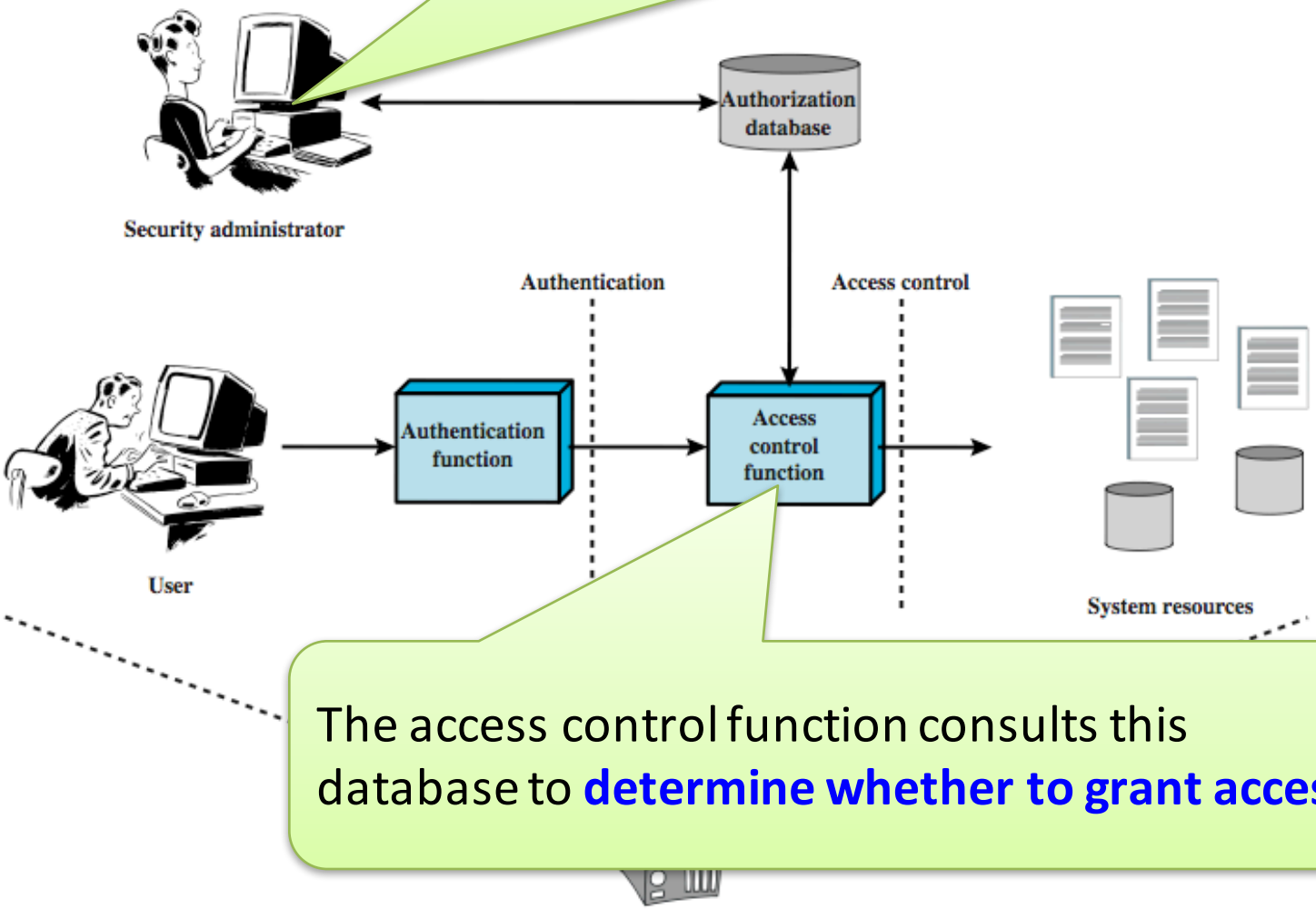
After studying this chapter, you should be able to:

- 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

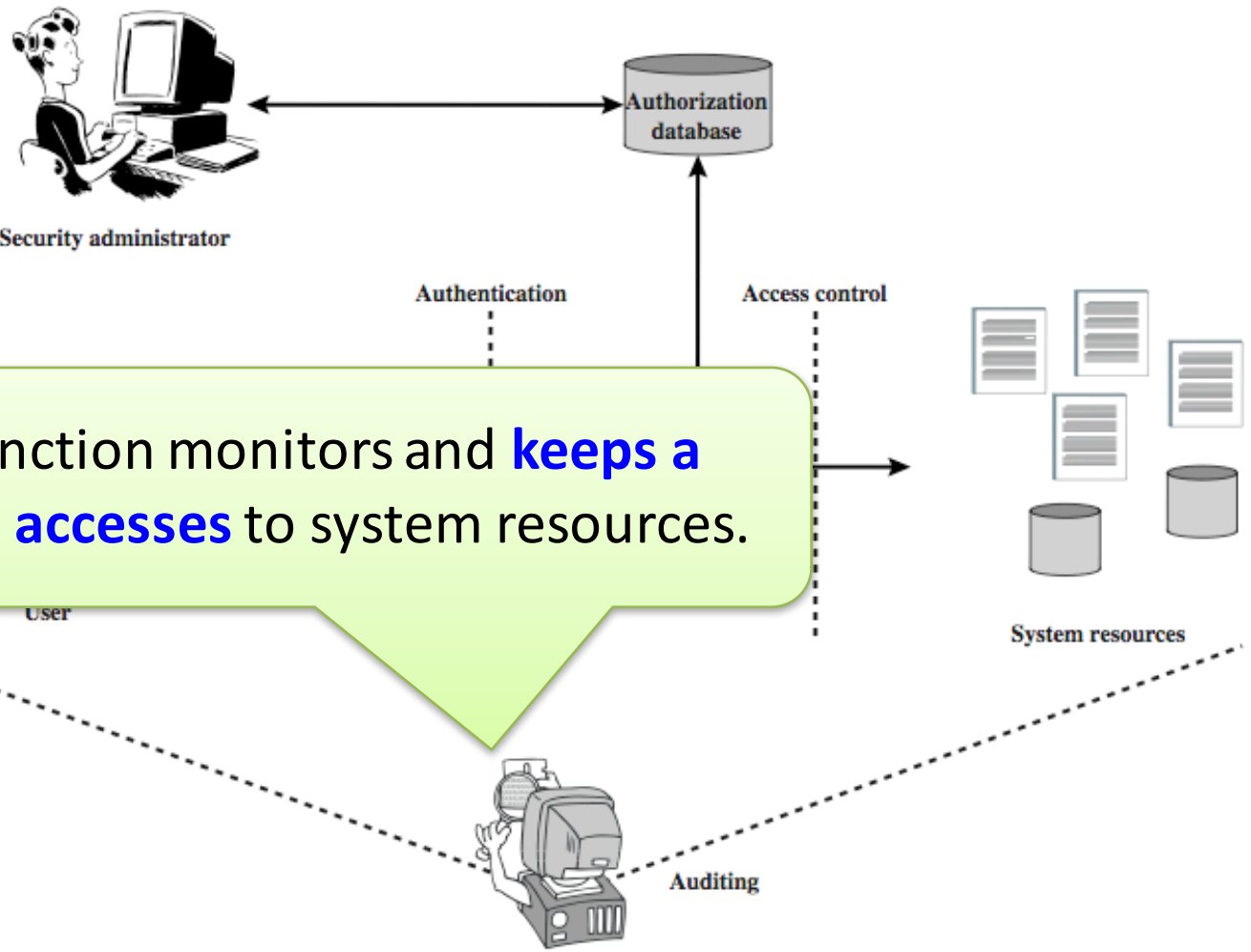
- "The **prevention of unauthorized use** of a resource, including the prevention of use of a resource in an unauthorized manner"
- **Central element** of computer security
- Assume have **users and groups**
  - authenticate to system
  - assigned access rights to certain resources on system

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**.

Access Control Principles

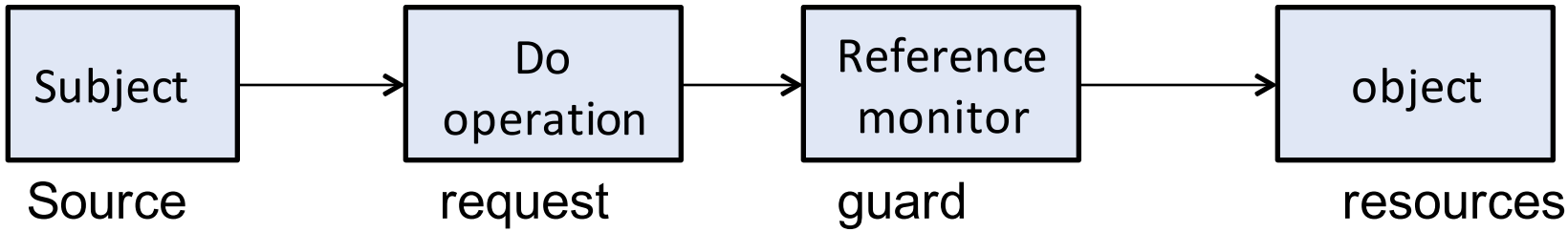


An auditing function monitors and **keeps a record of user accesses** to system resources.

Access control policies

- **Discretionary** access control (DAC): based on the identity of the requestor and access rules.
- **Mandatory** access control (MAC): based on comparing security labels with security clearances (mandatory: one with access to a resource cannot pass to others)
- **Role-based** access control (RBAC): based on user roles
- **Attribute-based** access control (ABAC): based on the attributes of the user, the resources and the current environment

Access Control Elements



- A *subject* wants to access an *object* with some *operation*. The *reference monitor* either grants or denies the access.

e.g. IVLE: a ***student*** wants to ***submit*** a ***forum post***.

IVLE: a ***student*** wants to ***read*** the ***grade of another student***.

File system: a ***user*** wants to ***delete*** a ***file***.

File system: a ***user Alice*** wants to ***change the mode*** of a ***file*** so that it can be read by Bob

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

ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

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
- **Access matrix is often sparse**
- **Can decompose by either row or column**

Access Control Structures

- Access control lists (decomposed by column)
- Capability tickets (decomposed by row)
- See page 119
- Also see alternative table representation on page 120 (tabular but not sparse)

An 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

(a) Access matrix

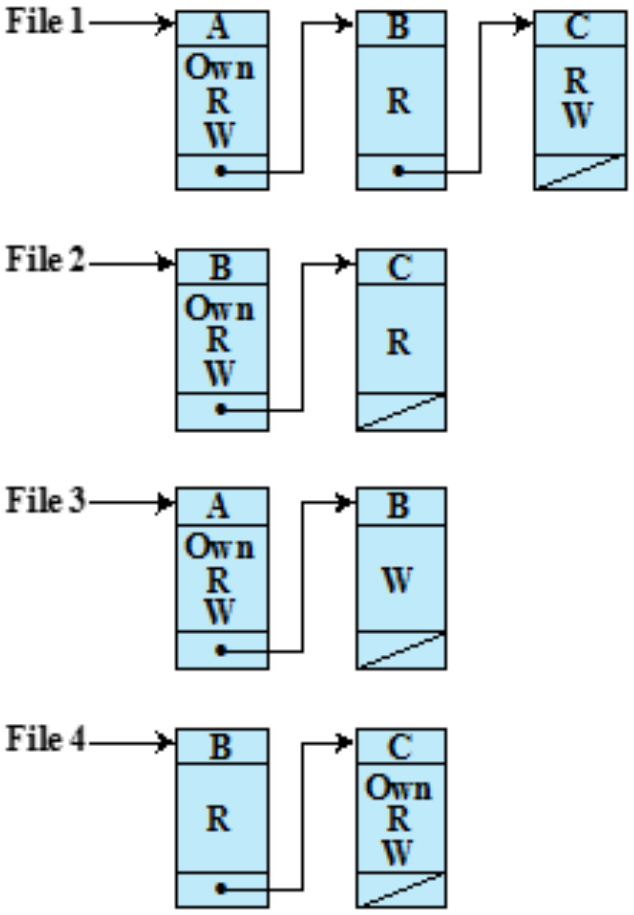
Access Control Matrix

How do we specific the access right of a particular principal to a particular object? Using a table.

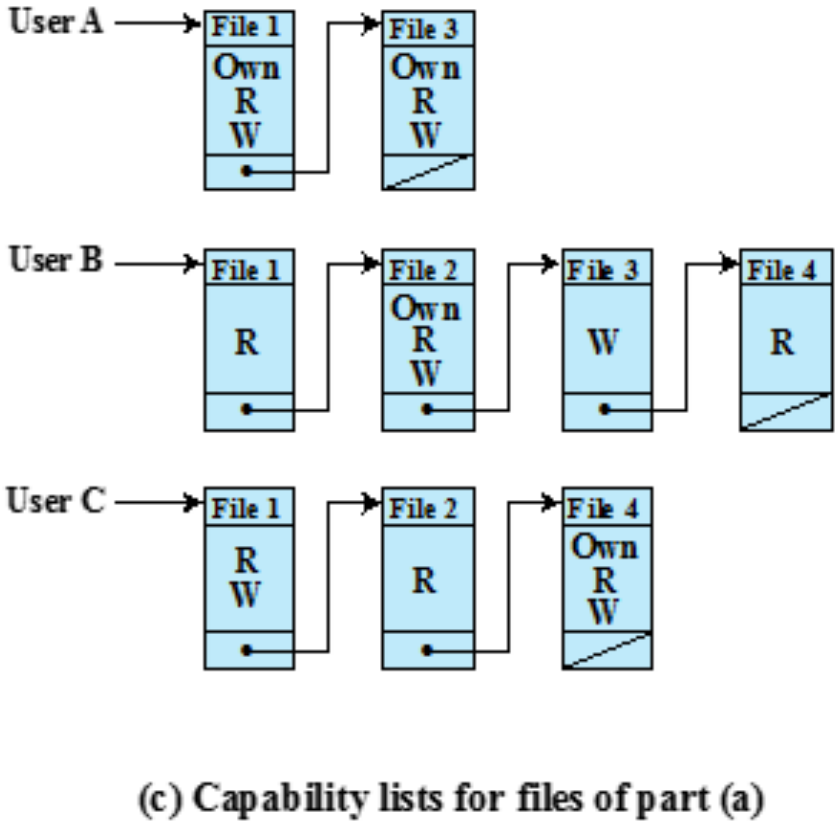
	my.c	mysh.sh	sudo	a.txt
root	{r,w}	{r,x}	{r,s,o}	{r,w}
Alice	{r,w}	{r,x,o}	{r,s}	{r,w,o}
Bob	{r,w,o}	{}	{r,s}	{}

r:read, w:write, x:execute, s: execute as owner, o: owner

Access matrix data structures



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



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

	my.c	mysh.sh	sudo	a.txt
root	{r,w}	{r,x}	{r,s,o}	{r,w}
Alice	{}	{r,x,o}	{r,s}	{r,w,o}
Bob	{r,w,o}	{}	{r,s}	{}

- **ACL**

my.c
mysh.sh
sudo
a.txt

(root, {r,w}) , (Bob, {r,w,o})

(root, {r,x}) , (Alice, {r,x,o})

(root, {r,s,o}) , (Alice, {r,s}) , (Bob, {r,s})

(root, {r,w}) , (root, {r,w,o})

- **Capability**

root
Alice
Bob

(my.c, {r,w}) , (mysh.sh, {r,x}) , (sudo, {r,s,o}) , (a.txt, {r,w})

(mysh.sh, {r,x,o}) , (sudo, {r,s}) , (a.txt, {r,w,o})

(my.c, {r,w,o}) , (sudo, {r,s})

Alternate authorization table

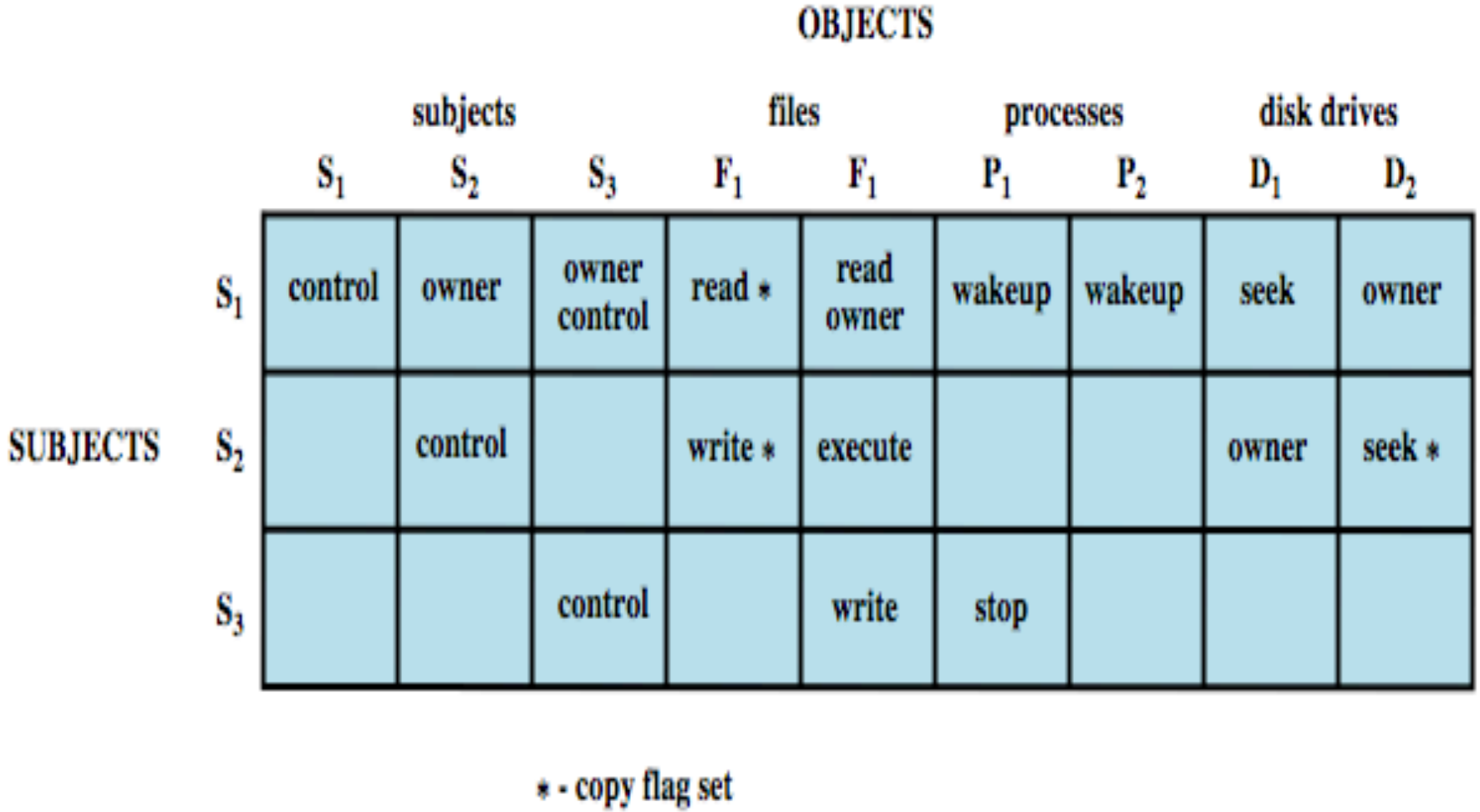
Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

An Access Control Model

Extend the universe of objects to include: **processes, devices, memory locations, subjects**

- **Processes:** Access rights include the ability to delete a process, stop (block), and wake up a process.
- **Devices:** Access rights include the ability to read/write the device, to control its operation (e.g., a disk seek), and to block/unblock the device for use.
- **Memory locations or regions:** Access rights include the ability to read/write certain regions of memory that are protected such that the default is to disallow access.
- **Subjects:** Access rights to a subject have to do with the ability to grant or delete access rights of that subject to other objects.

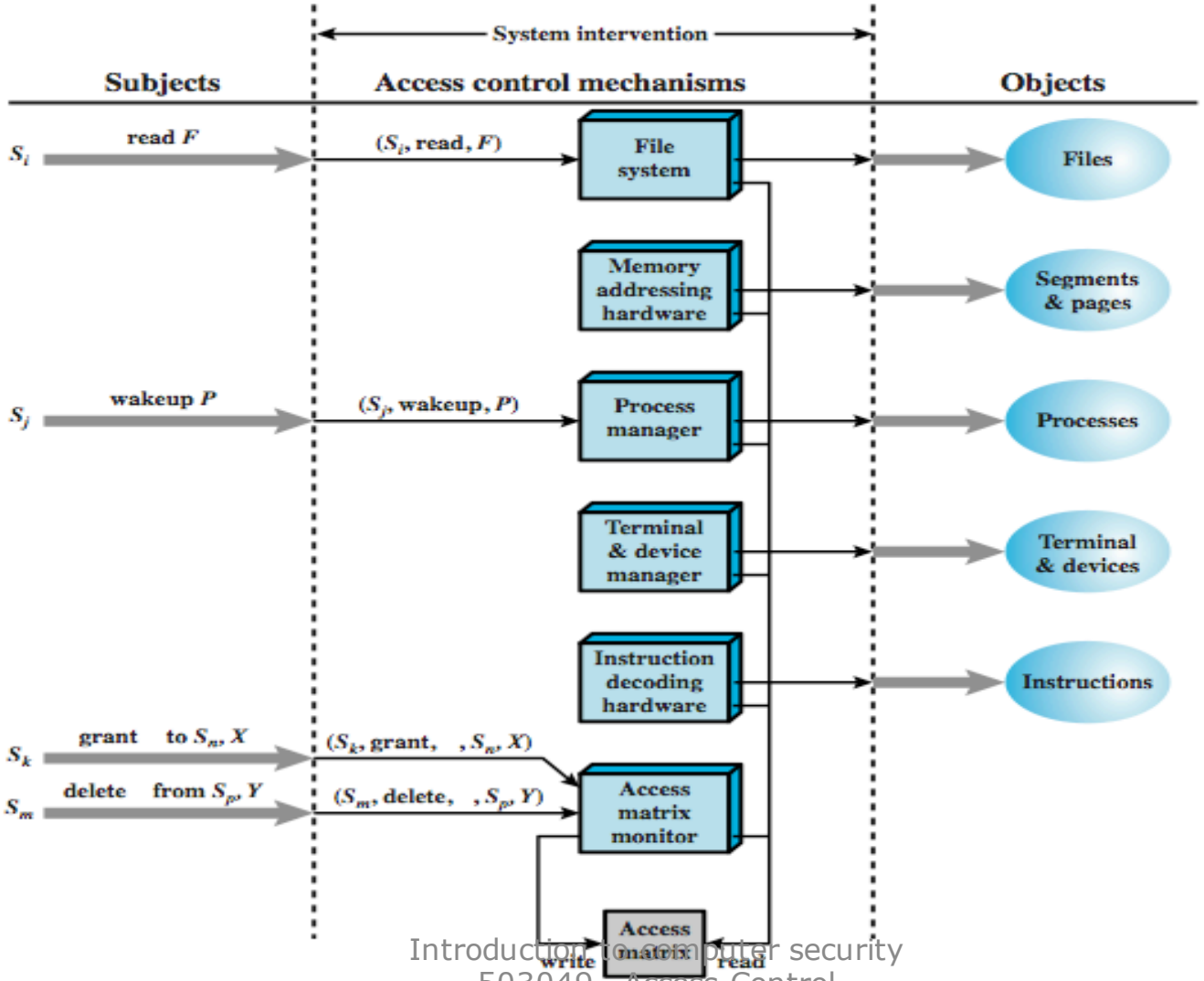
An Access Control Model



		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



Access control system commands

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

Protection Domains

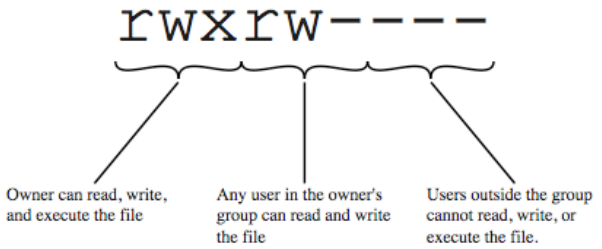
- Set of objects together with access rights to those objects
- In terms of the access matrix, a row defines a protection domain
- User can generate processes with a subset of the access rights of the user.
- Association between a process and a domain can be static or dynamic
- In user mode certain areas 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

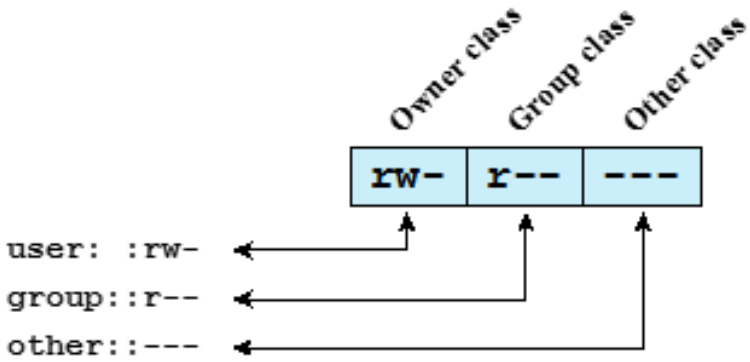
UNIX File Concepts

- All types of UNIX files are administered by using **inodes**
- **An inode (index node):**
  - Is a control structure with key info on file (attributes, permissions, …)
  - on a disk: an **inode table** or **inode list** for all files.
  - when a file is opened, its inode is brought into main memory and stored in a memory-resident inode table
- **Directories form a hierarchical tree**
  - may contain files or other directories
  - are a file of names and inode numbers

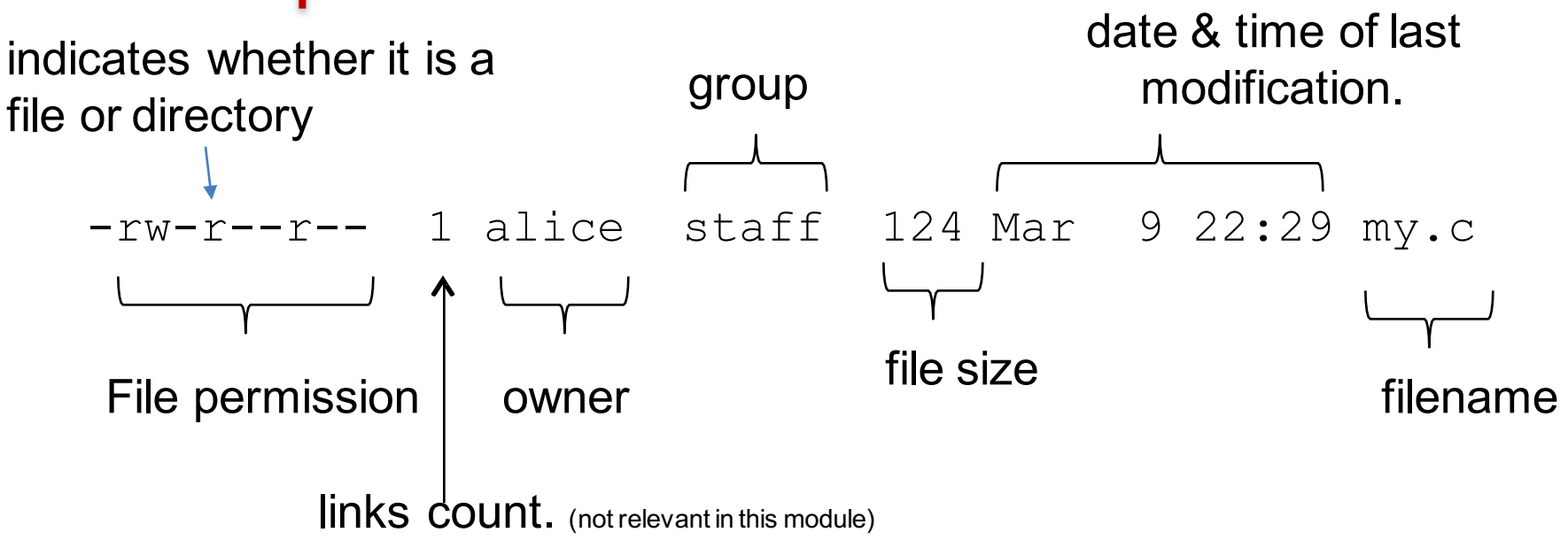
UNIX File Access Control

- Unique user identification number **(user ID)**
- Member of a primary group identified by a **group ID**
- **12 protection bits**
  - 9 specify read, write, and execute permission for the owner of the file, members of the group and all other users
  - 2 speficiy SetID, SetGID
  - 1 is the sticky bit (only owner can remove, delete, …, a directory)
- The owner ID, group ID, and protection bits are part of the file's inode





UNIX File Access Control



The file permission are grouped into 3 triples, that define the *read*, *write*, *execute* access for *owner, group, other (also called the "world").*

A '-' indicates access not granted. Otherwise

r: read

w: write (including delete)

x: execute (s: allow user to execute with the permission of the owner)

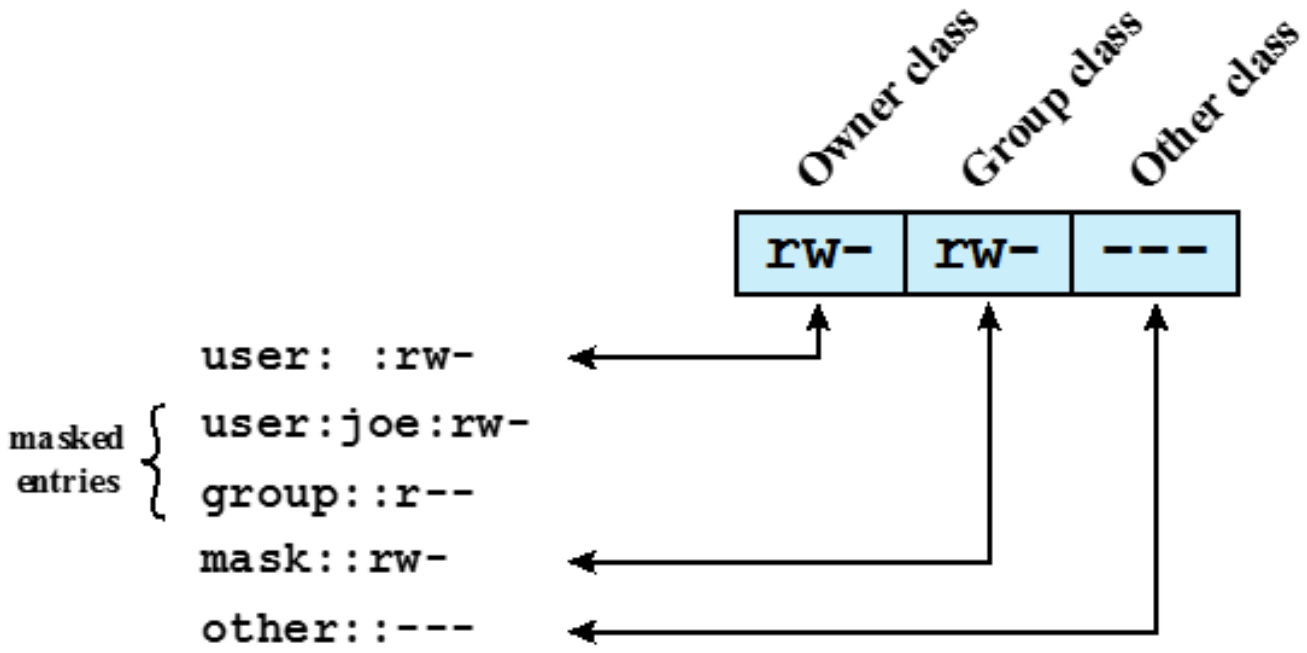
UNIX File Access Control

- "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
- When access is required
  - select most appropriate ACL
    - owner, named users, owning/named groups, others
  - check if have sufficient permissions for access

UNIX extended access control list



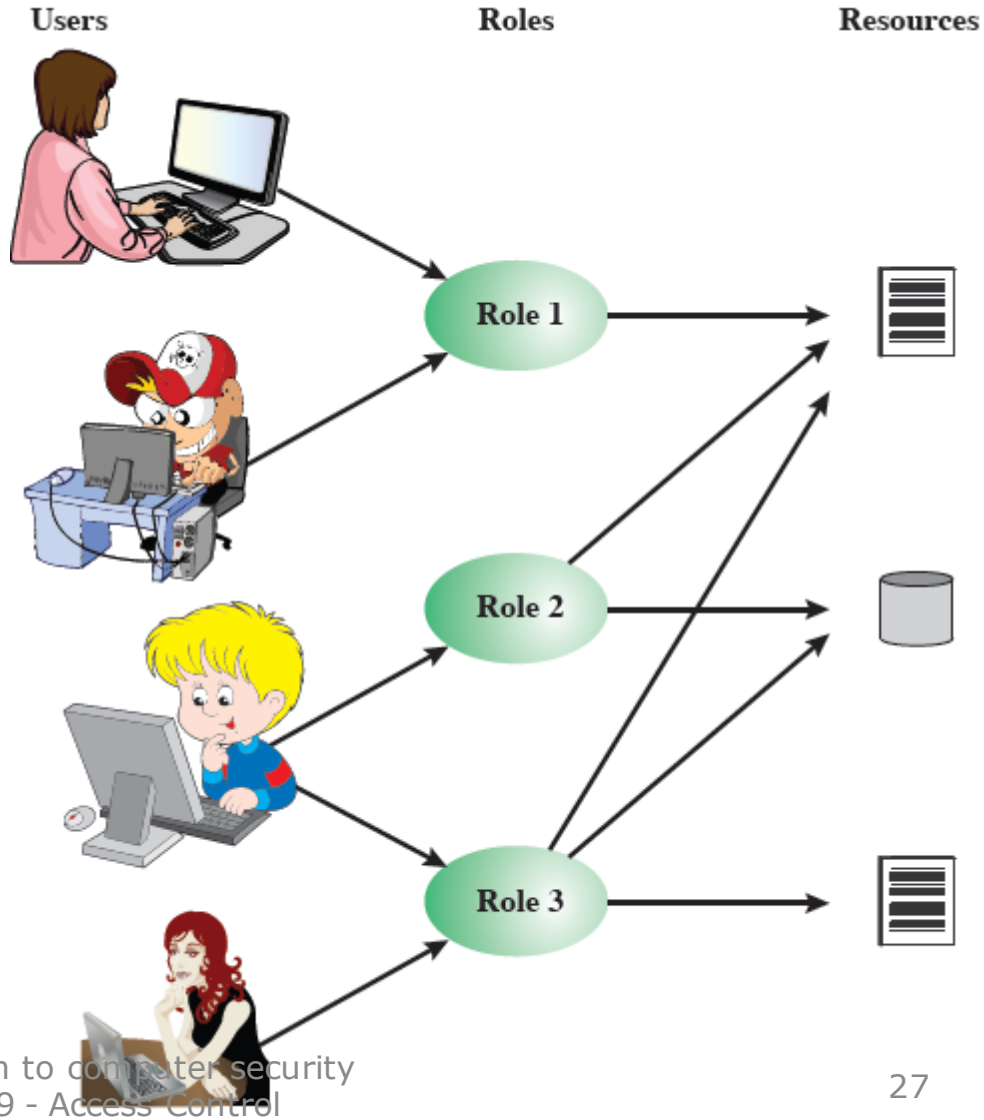
(b) Extended access control list

Role-Based Access Control

Access based on 'role', **not identity**

Many-to-many relationship between users and roles

Roles often static



Role-Based Access Control

Role-users and roles-object access matrix

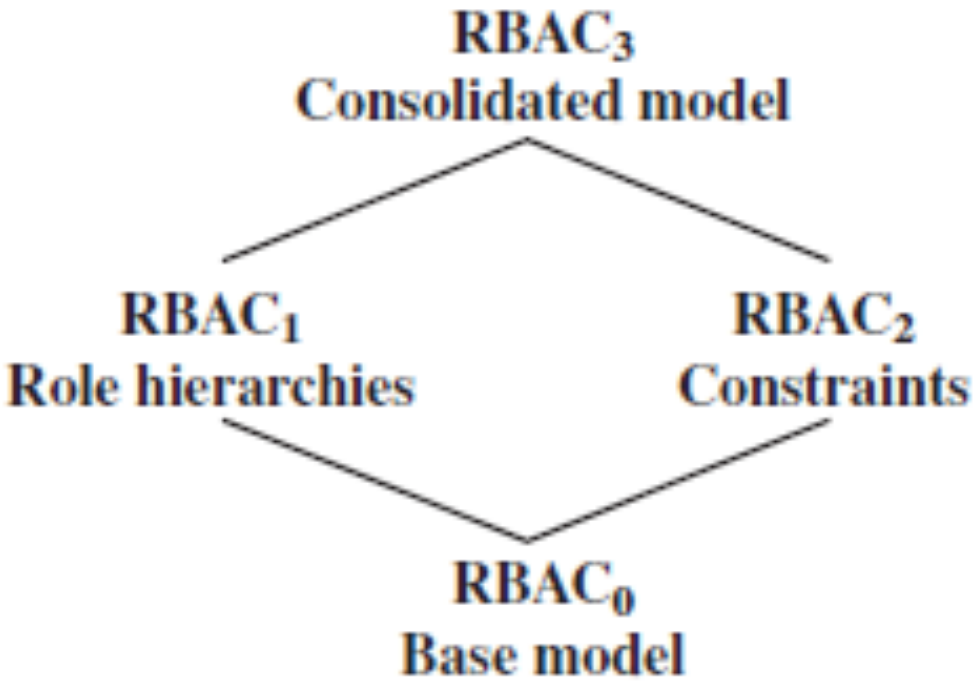
	R_1	R_2	• • •	R_n
U_1	✖			
U_2	✖			
U_3		✖		✖
U_4				✖
U_5				✖
U_6				✖
• • •				
U_m	✖			

	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			

General RBAC, Variations

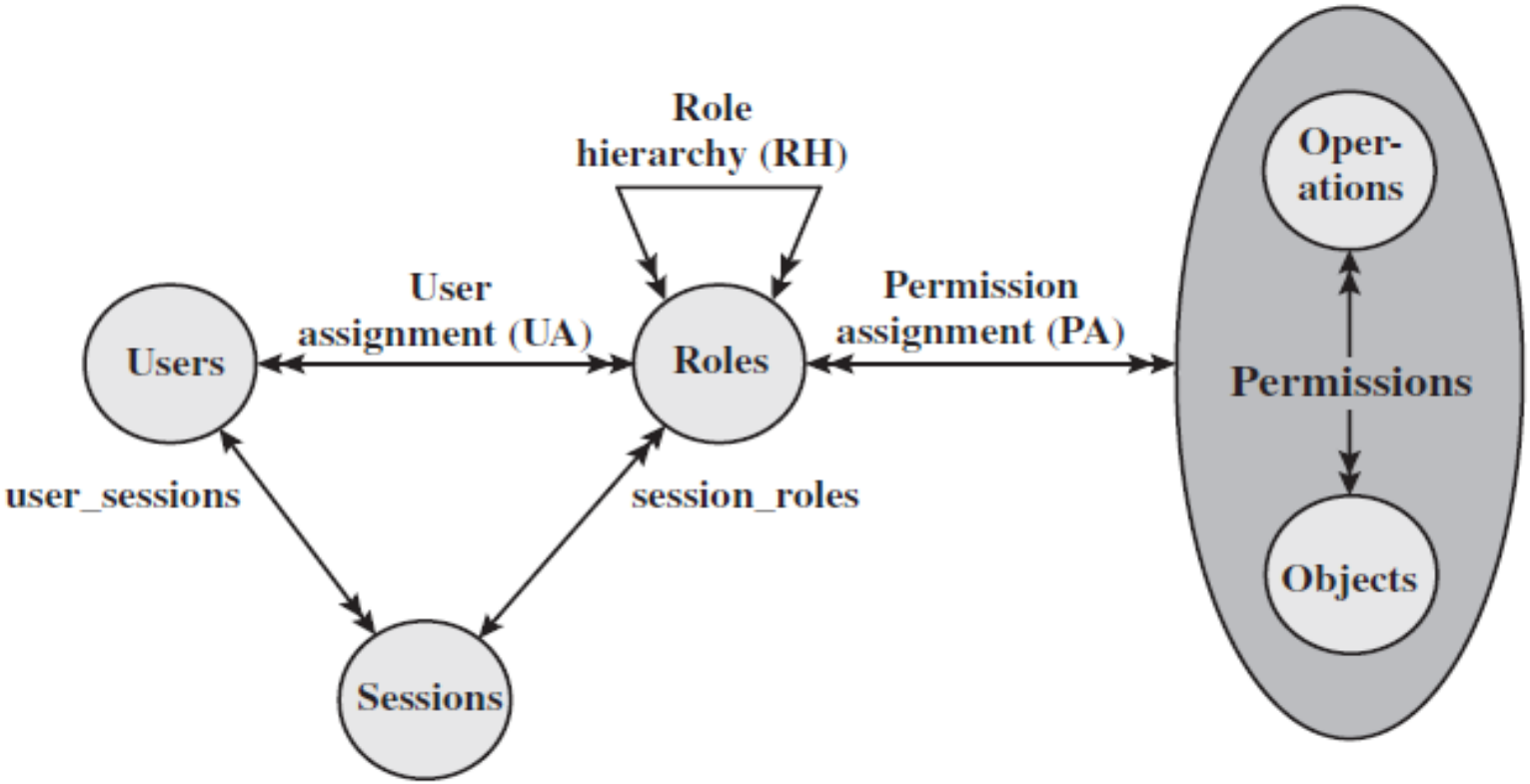
- A family of RBAC with four models:
  1. $RBAC_0$: contains the minimum functionality for an RBAC system
  2. $RBAC_1$: $RBAC_0$ plus role (permission) inheritance
  3. $RBAC_2$: $RBAC_0$ plus constraints (restrictions)
  4. $RBAC_3$: $RBAC_0$ plus all of the above
- **$RBAC_0$ entities**
  - ***User:*** an individual (with UID) with access to system
  - ***Role:*** a named job function (tells authority level)
  - ***Permission:*** equivalent to access rights
  - ***Session:*** a mapping between a user and set of roles to which a user is assigned

Role-Based Access Control



(a) Relationship among RBAC models

Role-Based Access Control



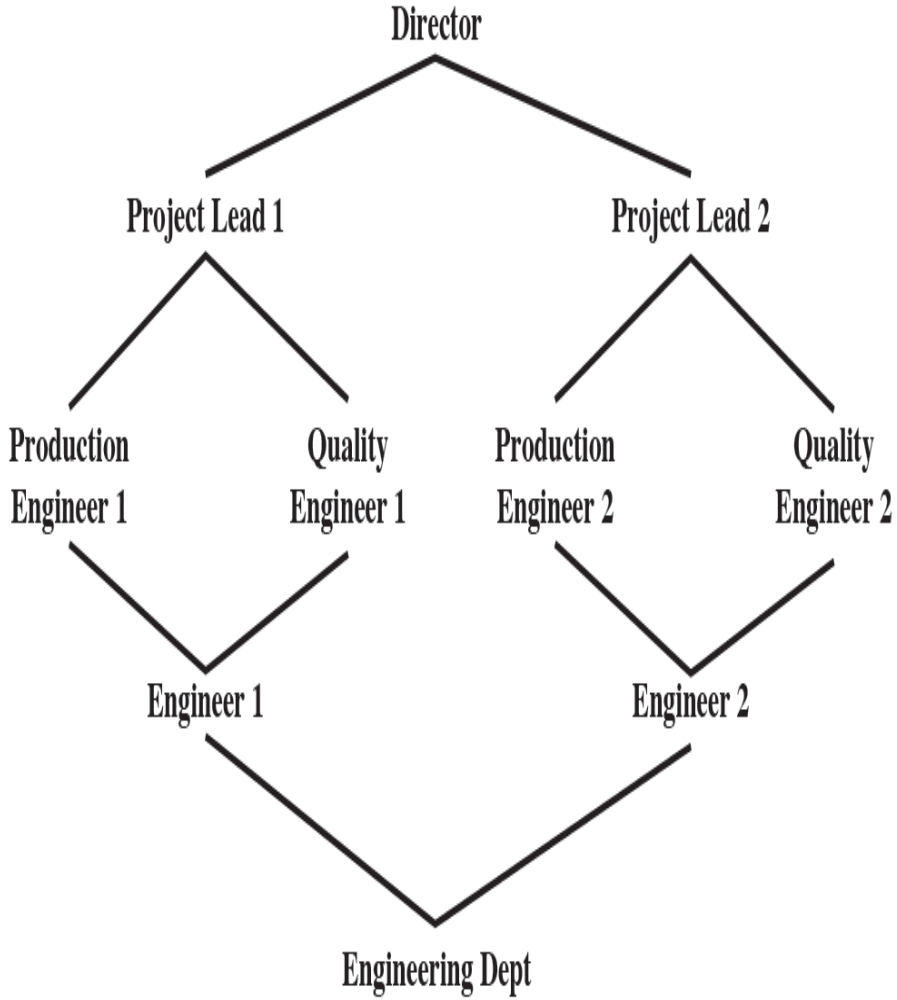
(b) RBAC models

Double arrow: 'many' relationship

Single arrow: 'one' relationship

Example of role hierarchy

- Director has most privileges
- Each role inherits all privileges from lower roles
- A role can inherit from multiple roles
- Additional privileges can be assigned to a role



Constraints

- A condition (restriction) on a role or between roles
  - **Mutually exclusive**
    - A user can only be assigned to one role in the set
    - Any permission can be granted to only one role in the set
  - **Cardinality**: set a maximum number (of users) with a role (e.g., a department chair role)
  - **Prerequisite role**: a user can be assigned a role only if that user already has been assigned to some other role

Attribute-based access control

- Fairly recent
- Define authorizations that express conditions on properties of both the resource and the subject
  - Each resource has an attribute (e.g., the subject that created it)
  - A single rule states ownership privileges for the creators
- Strength: its flexibility and expressive power
- Considerable interest in applying the model to cloud services

Types of attributes

- Subject attributes
- Object attributes
- Environment 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
- Each subject has **associated attributes** that define the identity and characteristics of the subject:
  - Name
  - Organization
  - Job title

Object attribute

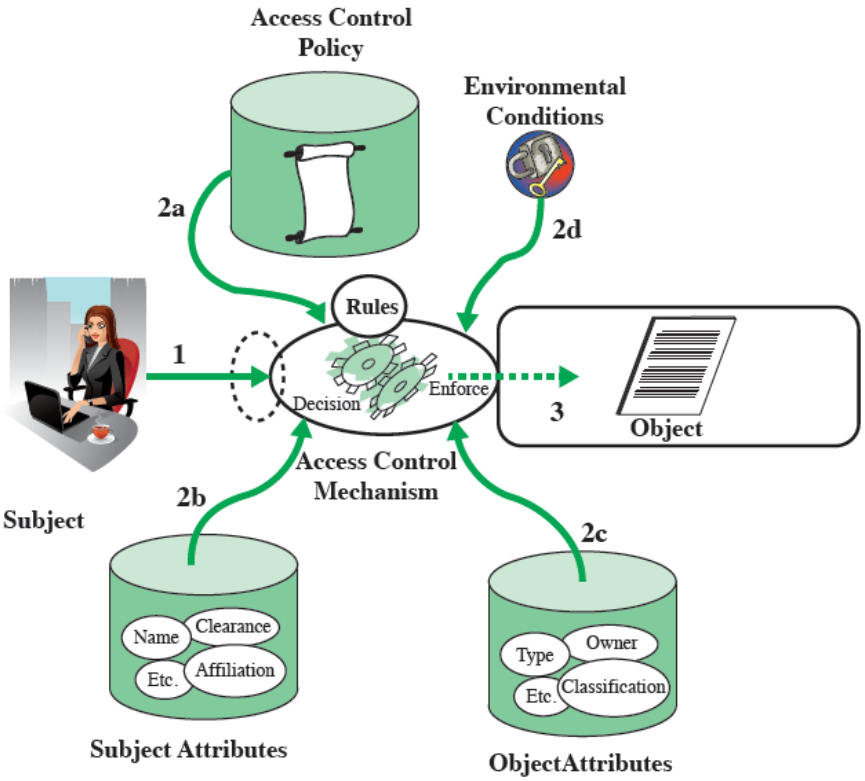
- An object (or resource) is a passive information system-related entity *(e.g., devices, files, records, tables, processes, programs, networks, domains)* containing or receiving information
- Objects have attributes that can be used to make access control decisions
  - Title
  - Author
  - Date

Environment attributes

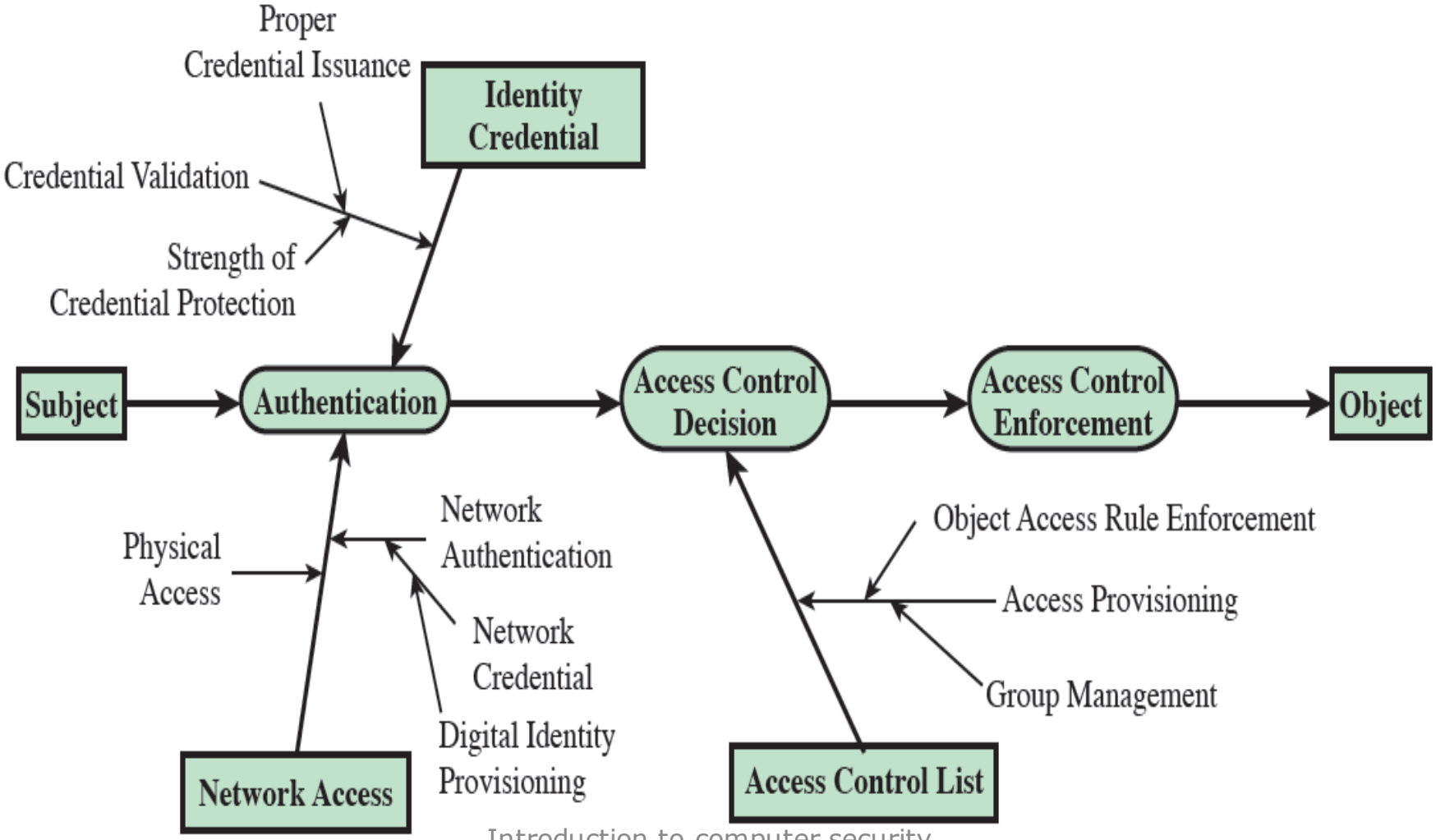
- Describe the operational, technical, and even situational environment or context in which the information access occurs
  - Current date
  - Current virus/hacker activities
  - Network security level
  - *Not associated with a resource or subject*
- These attributes have so far been largely ignored in most access control policies

Sample ABAC scenario

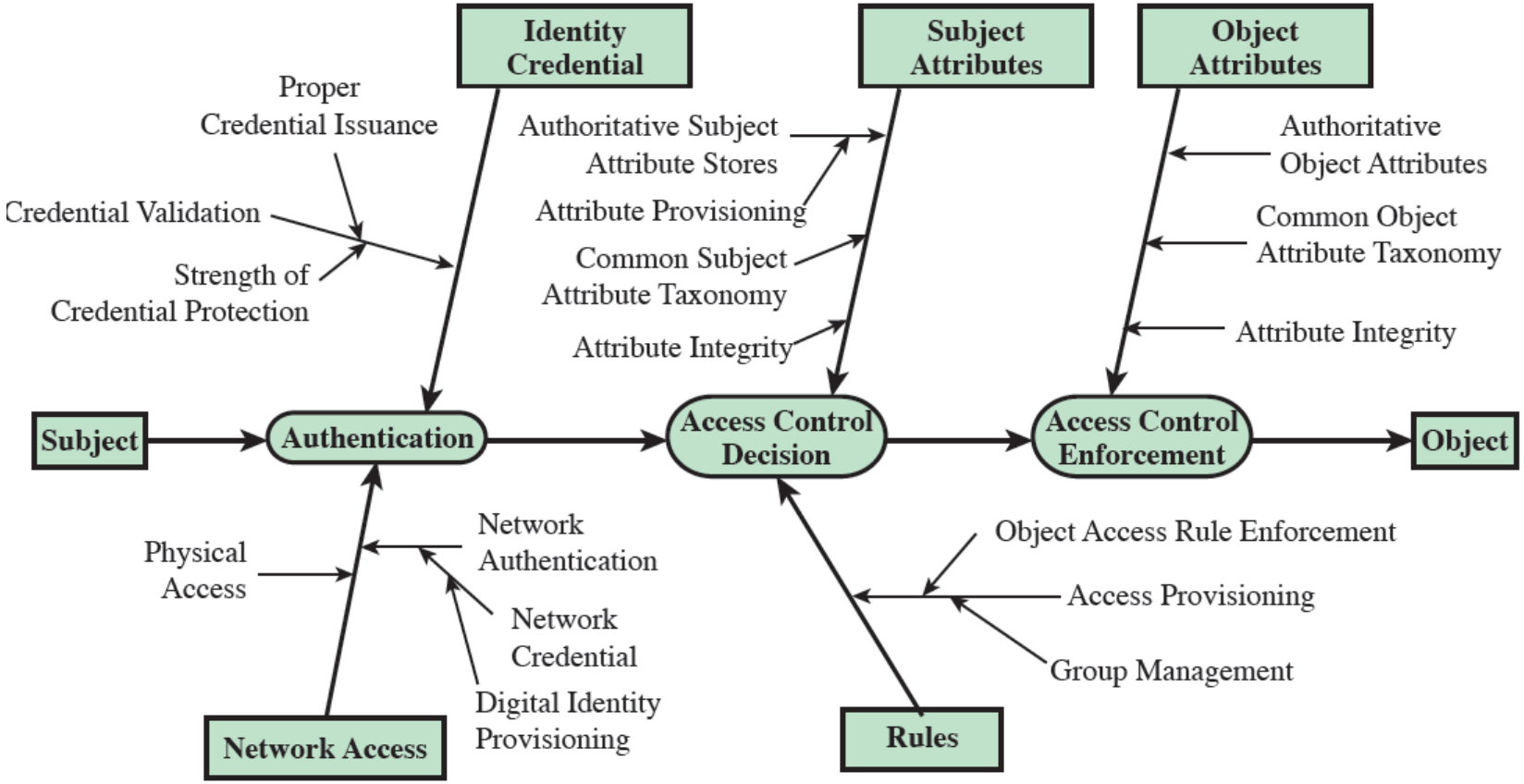
1. A subject requests access to an object
2. AC is governed by a set of rules (2a): assesses the attr of subject (2b), object (2c) and env (2d)
3. AC grants subject access to object if authorized



ACL vs ABAC trust relationships



ACL vs ABAC trust relationships



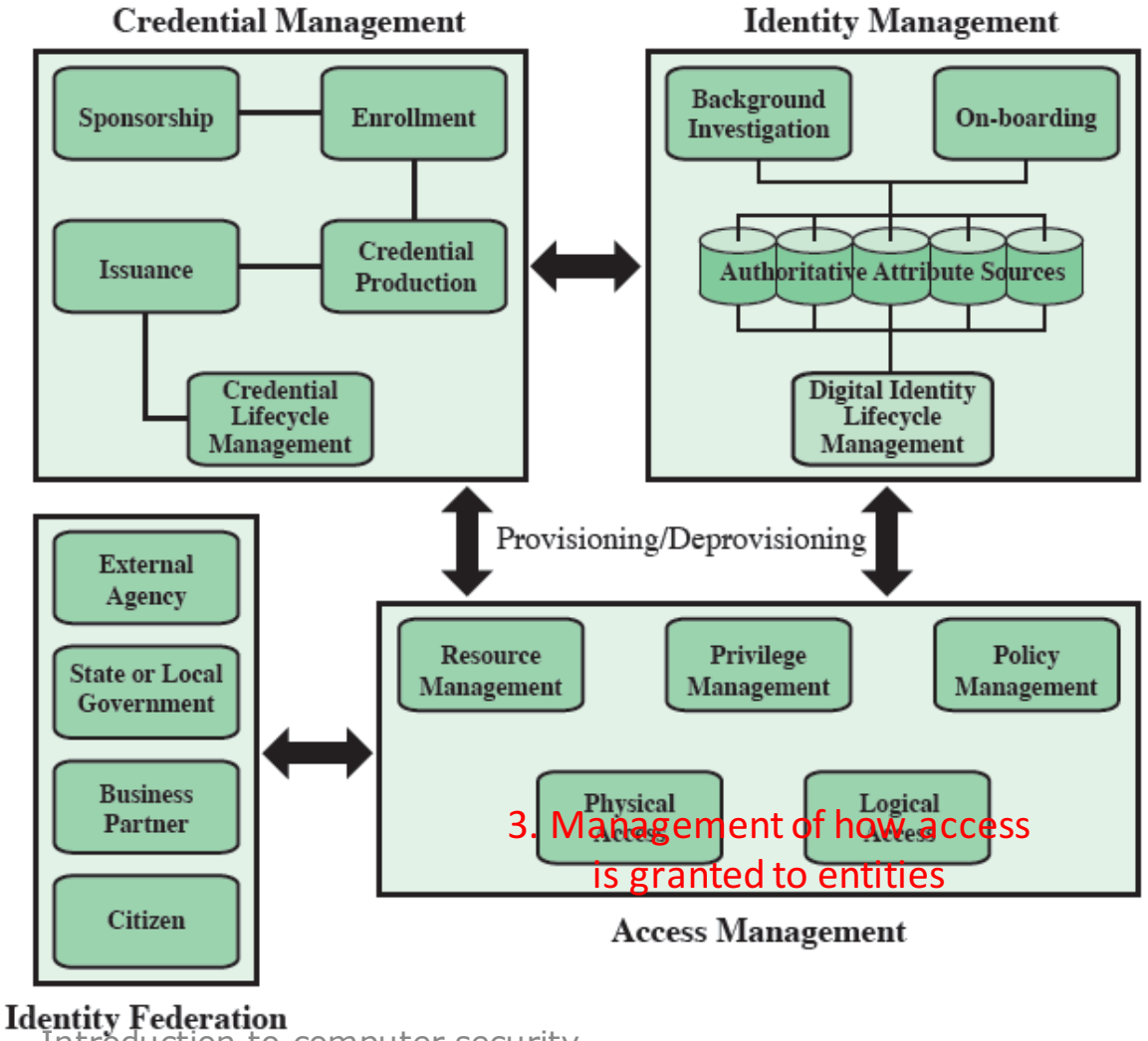
Identity, Credential, and Access Management (ICAM)

- 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)
- 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

ICAM

1. Connects digital identity to individuals

2. Data structures that binds a token possessed by a subscriber



3. Management of how access is granted to entities

4. Identity verification of individuals from external organizations

Case study: RBAC system for a bank

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

Case study: RBAC system for a bank

- b has more access than A (strict ordering)
- Inheritance makes tables simpler

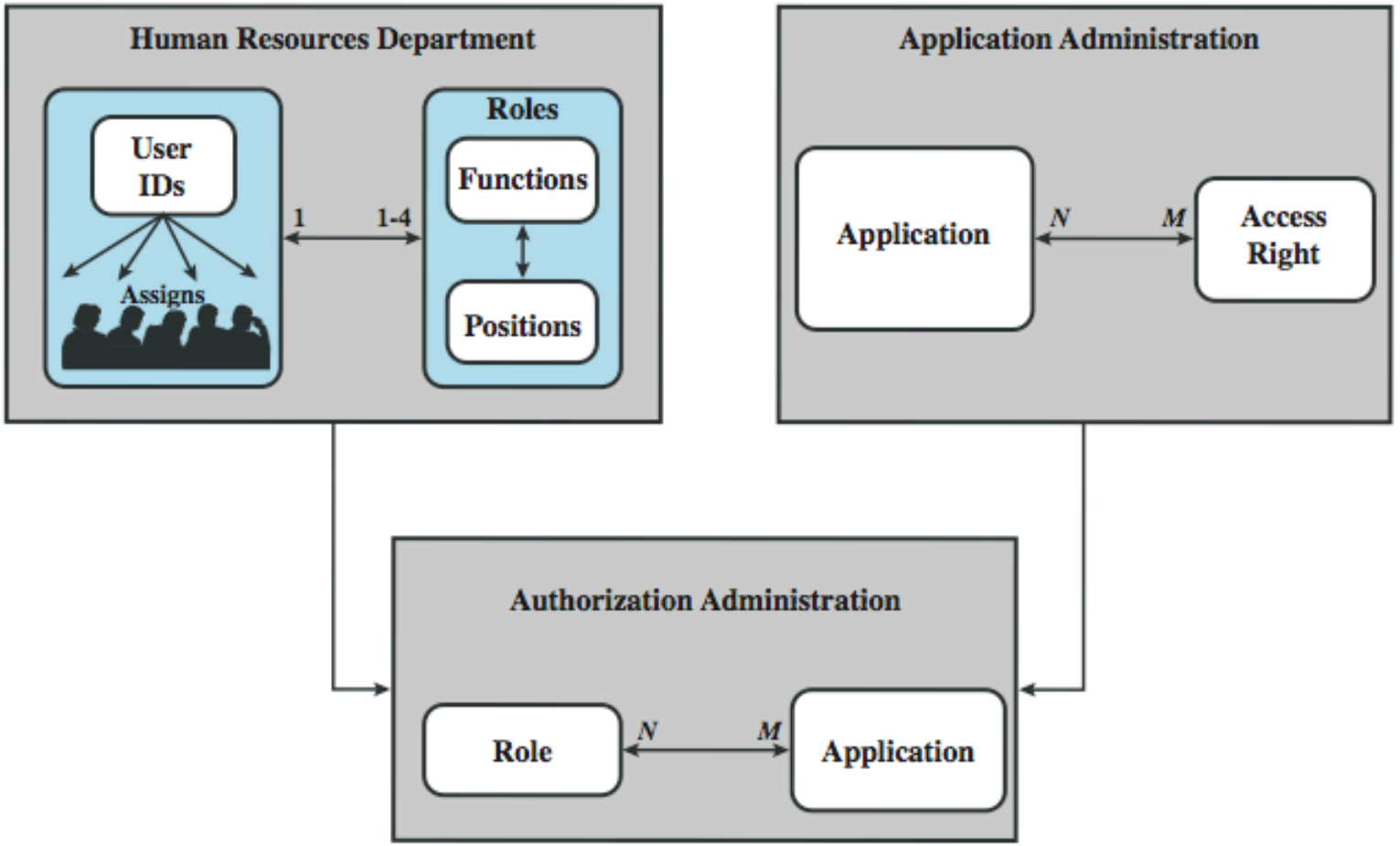
(b) Permission Assignments

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

(c) PA with Inheritance

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

Case study: RBAC system for a bank



ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Summary

- introduced access control principles
  - subjects, objects, access rights
- discretionary access controls
  - access matrix, access control lists (ACLs), capability tickets
  - UNIX traditional and ACL mechanisms
- role-based access control
- case study