Έλεγχος πρόσβασης

Νικόλαος Ε. Κολοκοτρώνης
Επίκουρος Καθηγητής

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Πελοποννήσου
Email: nkolok@uop.gr
Web: http://www.uop.gr/~nkolok/

ΑΣΦΑΛΕΙΑ ΣΥΣΤΗΜΑΤΩΝ

Περιεχόμενα

Περιεχόμενα

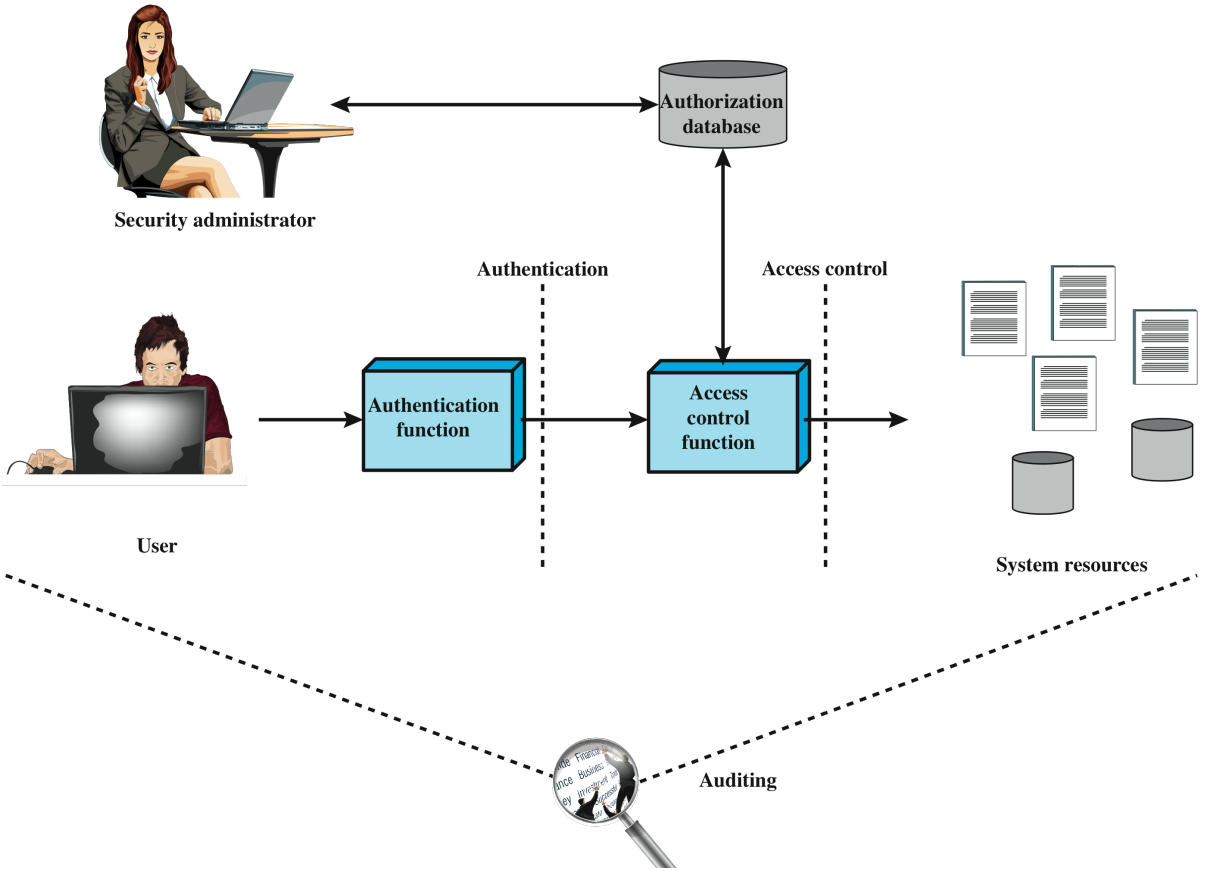
- Access control
  - Discretionary access control
  - Mandatory access control
  - Role-based access control
- Trusted systems

Access control principles

- ITU-T recommendation X.800 defines access control as
  - The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner
- RFC 2828 defines computer security as follows
  - Measures that implement and assure security services in a computer system, particularly those that assure access control service

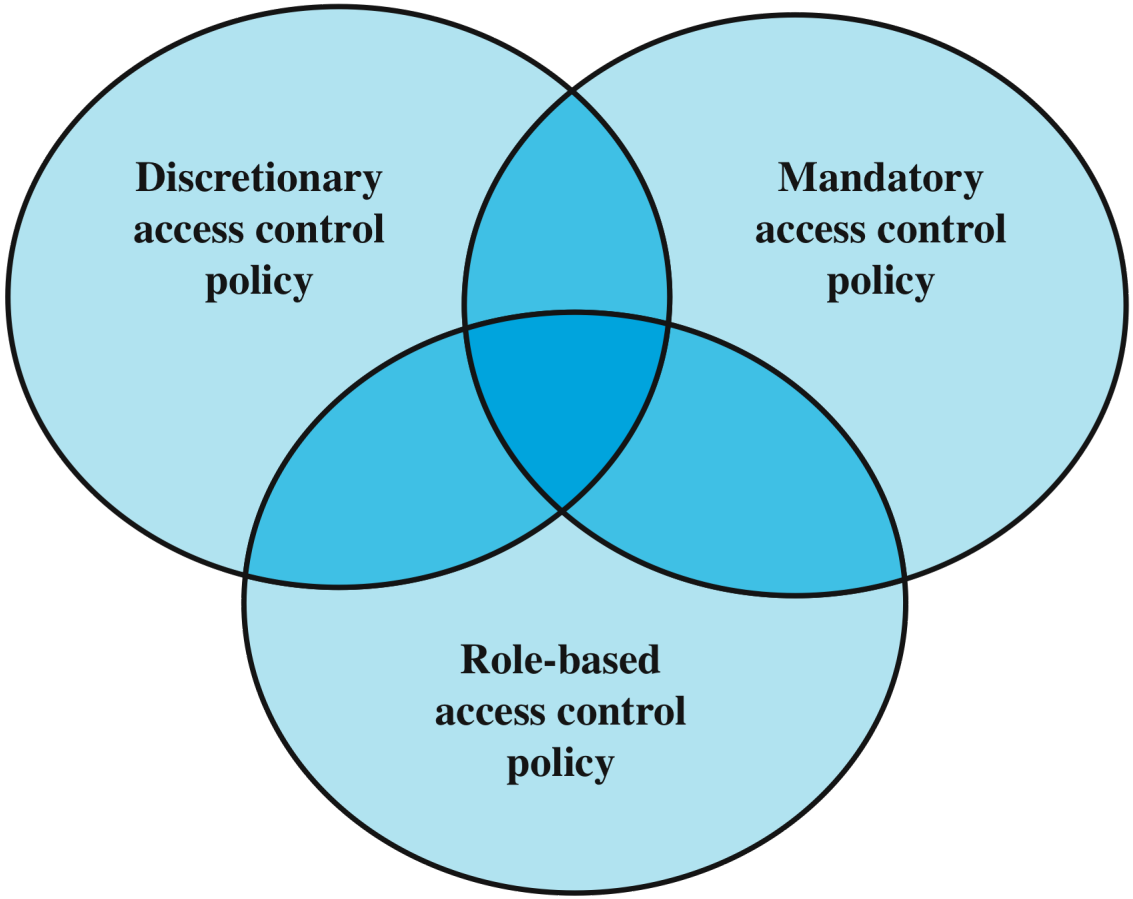
Access control scope

Relationship among access control and other security functions



Access control policies

DAC, MAC, and RBAC are not mutually exclusive. A system may implement two or even three of these policies for some or all types of access.



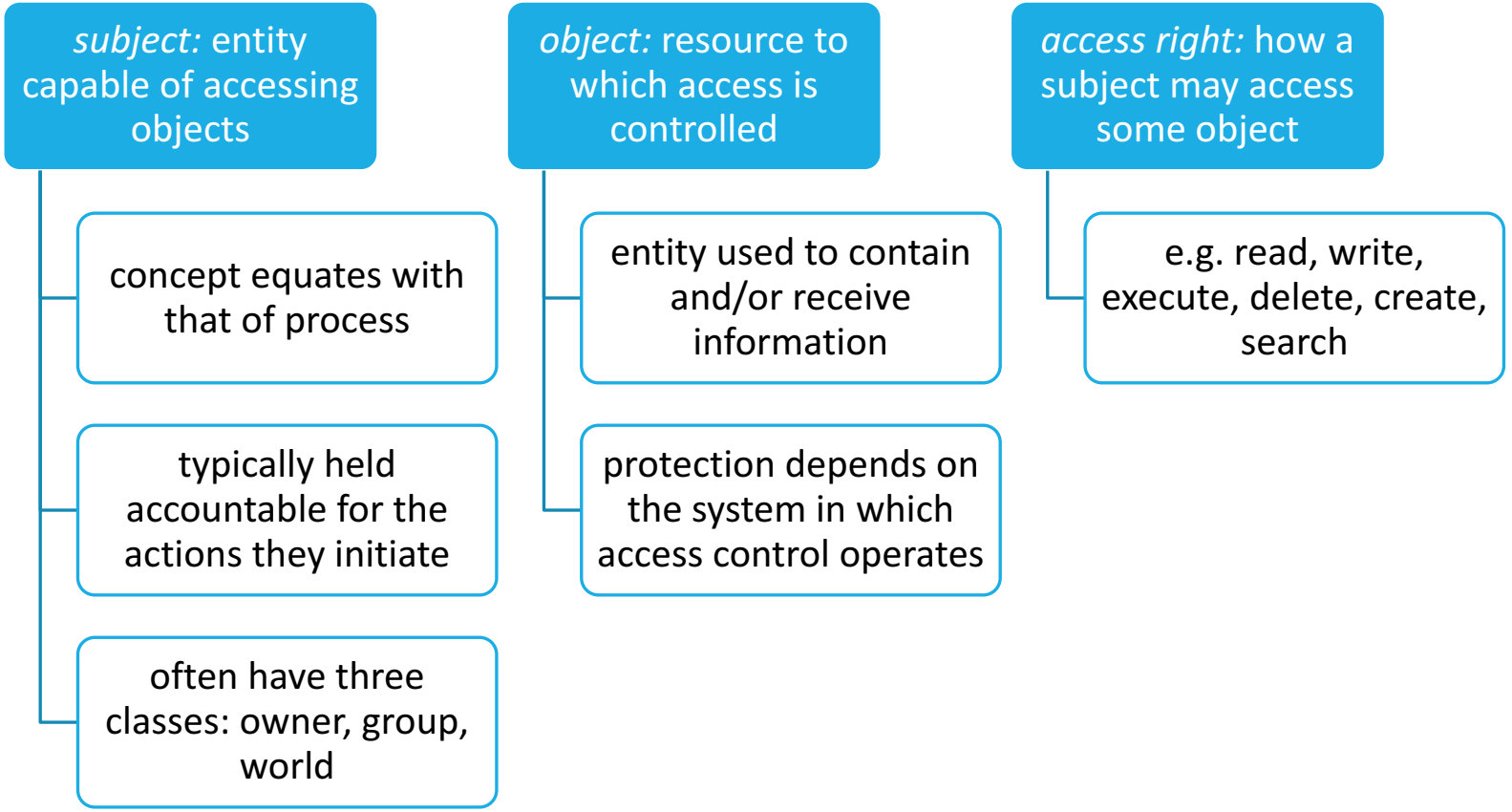
Access control policies

- Discretionary access control (DAC)
  - Controls access based on the identity of a requestor and on access rules (authorizations) stating what requestors are allowed to do
- Mandatory access control (MAC)
  - Controls access comparing security labels and security clearances
    - security labels indicate how sensitive or critical system resources are
    - security clearances indicate system entities are eligible to access certain resources
- Role-based access control (RBAC)
  - Controls access based on the roles that users have in the system and on rules stating what accesses are allowed to any given role

Access control requirements

- Reliable input
- Support for fine and coarse specifications
- Least privilege
- Separation of duty
- Open and closed policies
- Policy combinations and conflict resolution
- Administrative policies
- Dual control

Access control basic elements



Discretionary access control

DAC: discretionary access control

- Scheme in which an entity may enable another entity to access some resource
  - often provided using an access control 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

Access control matrix

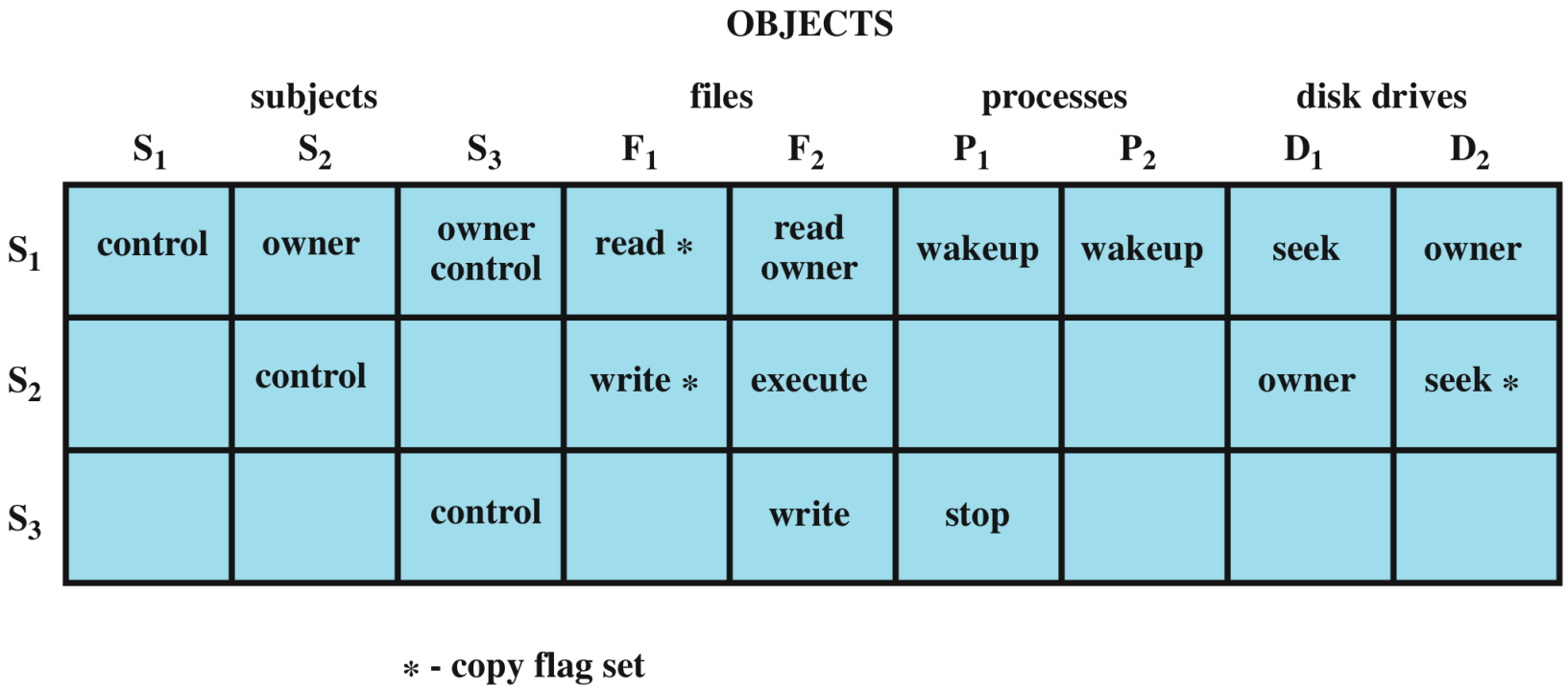
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

Authorization table

Example for the files in previous access control matrix

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

Extended access control matrix

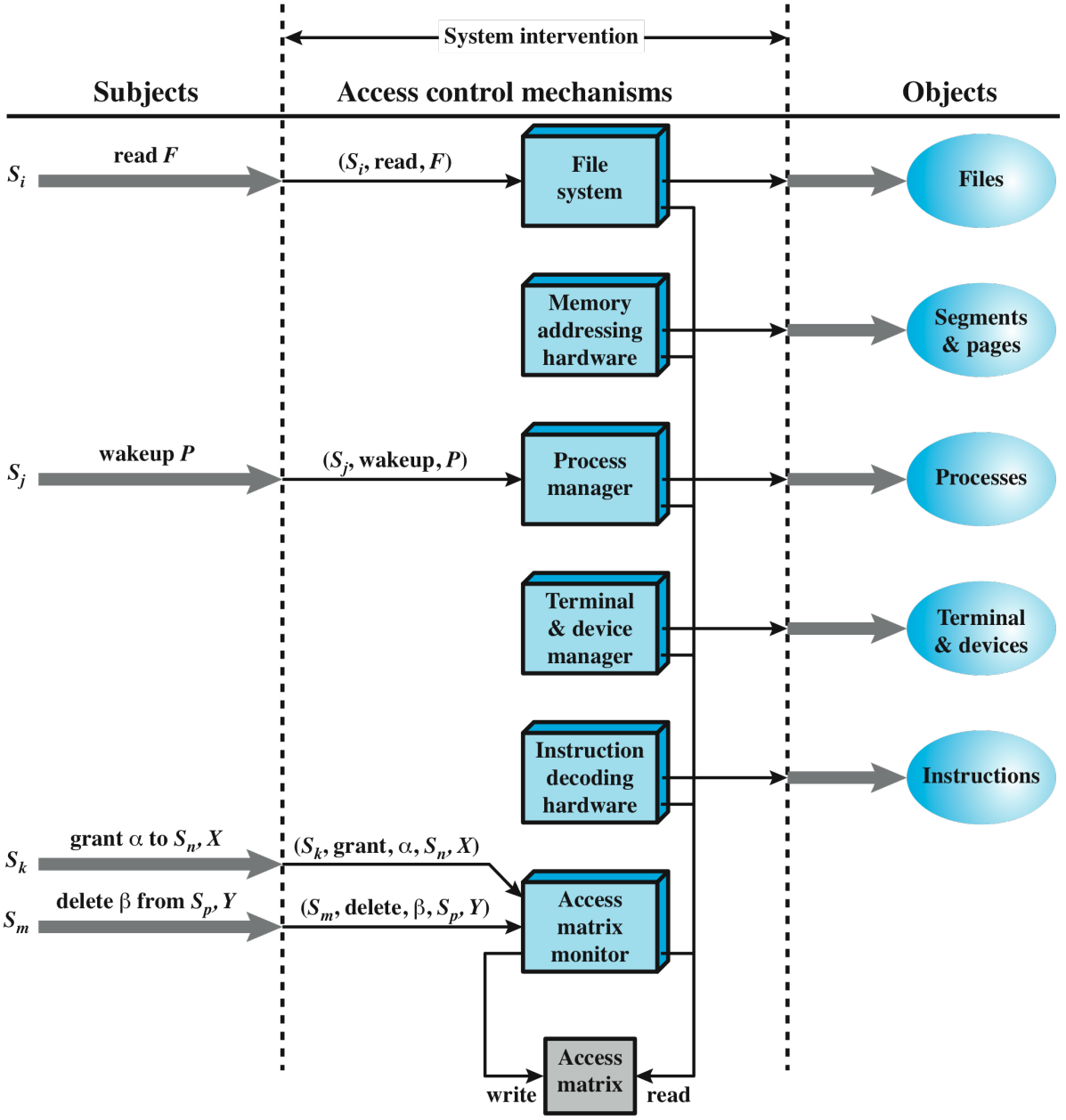


	OBJECTS								
	subjects			files		processes		disk drives	
	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
S_2		control		write *	execute			owner	seek *
S_3			control		write	stop			

* - copy flag set

Access control function

Organizational diagram



Access control commands

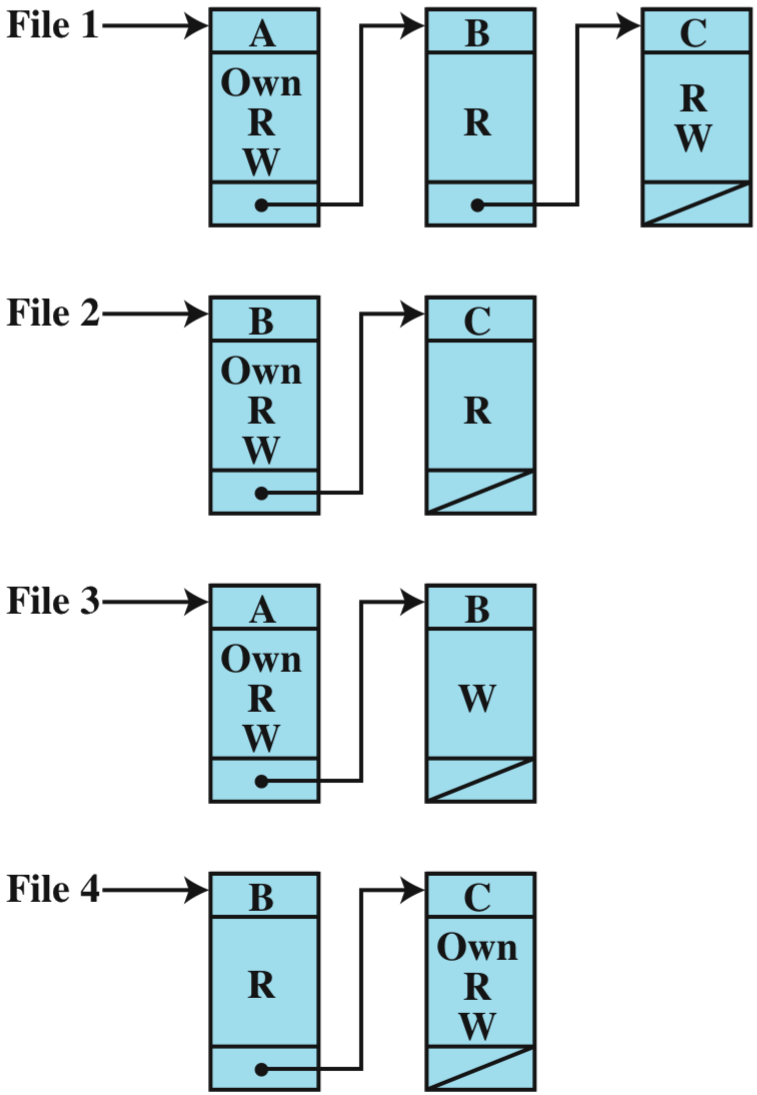
System commands used for managing the access matrix

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

Protection domains

- A protection domain is a set of objects together with access rights to those objects
  - More flexibility when associating capabilities with protection domains
  - 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

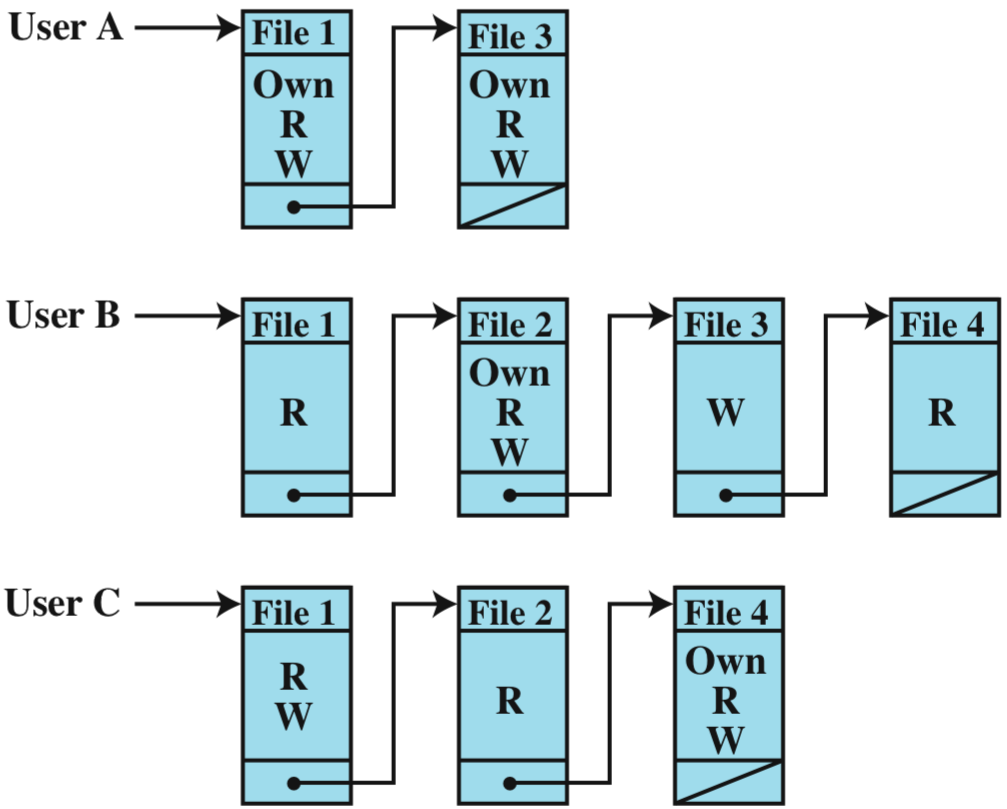
Access control lists



Access control lists

- Normal: if not named, no rights over file
  - Principle of fail-safe defaults
- Ownership assigned based on creating process
- Who can modify ACLs?
  - Creator is given own right that allows this
- Do ACLs apply to privileged users (root)?
  - The root is exempt from usual access control restrictions
  - Has system-wide access

Capability lists



Capability lists

- Like a bus ticket
  - Mere possession indicates rights that subject has over object
  - Object identified by capability (as part of the token)
- Must prevent process from altering capabilities
  - Otherwise subject could change rights encoded in capability or object to which they refer
- Revocation of right is far too expensive!
  - Scan all capability lists to remove relevant capabilities
  - Indirection can be used as a solution

Comparison of approaches

- Both approaches are theoretically equivalent
- They consider the following questions
  - Given some subject, what objects can it access, and how?
  - Given some object, what subjects can access it, and how?
- ACLs (c-lists) answer the second (first) easily

UNIX file access control

UNIX files are administered via 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 info. 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

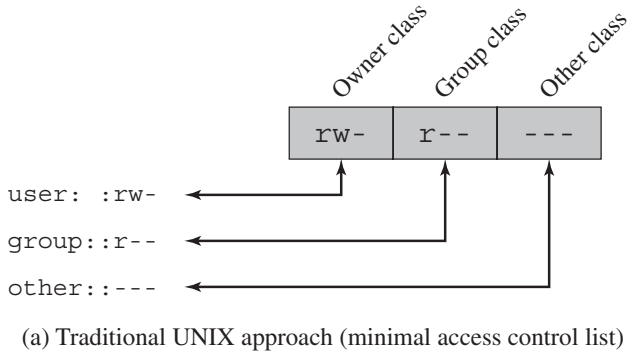
UNIX file access control

- Unique user identification number (user ID)
- Member of a main group identified by a group ID
- Belongs to a specific group
- 12 protection bits

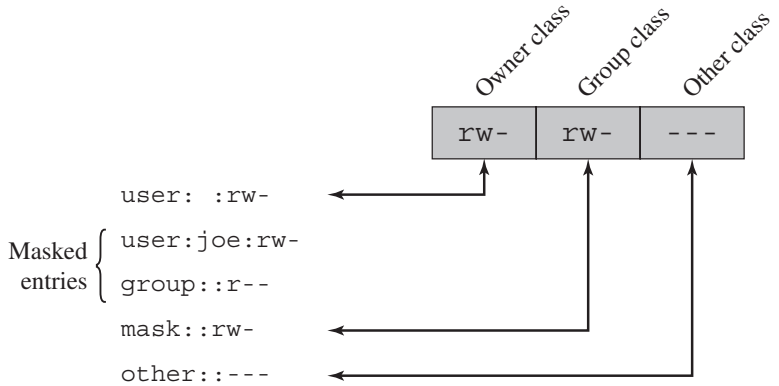
- For the owner of the file, members of the group and all other users specify the permission
  - read, write, and execute
- Part of the file's inode are also the bits
  - owner ID, group ID, and protection

UNIX file access control

- Unique user identification number (user ID)
- Member of a main group identified by a group ID
- Belongs to a specific group
- 12 protection bits



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



(b) Extended access control list

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

UNIX access control lists

- 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

UNIX access control lists

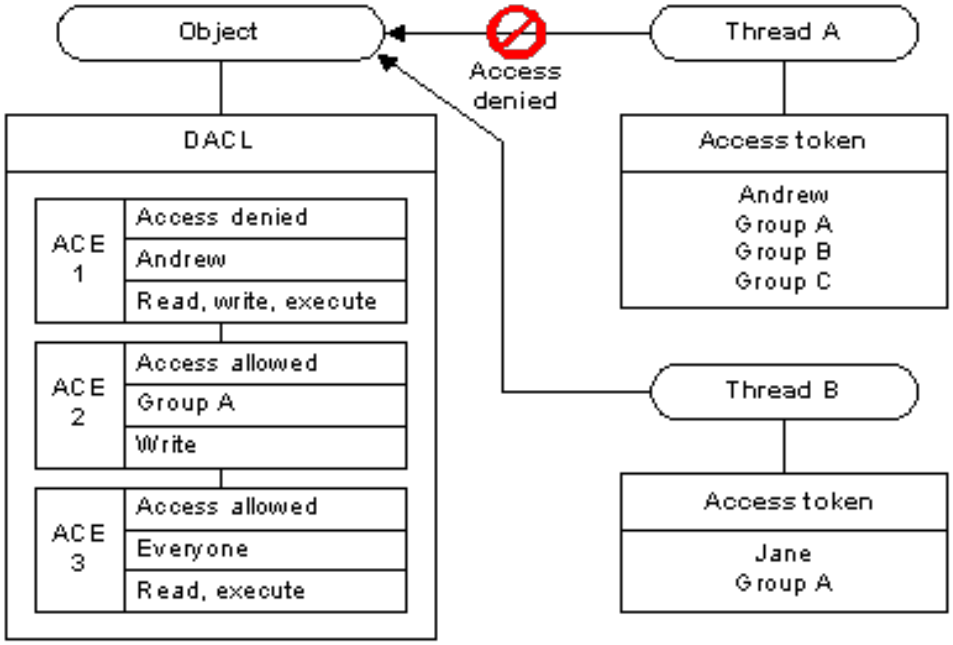
- When a process requests access to a file system object two steps are performed
  - Step 1. selects the most appropriate ACL
    - owner
    - named users
    - owning / named groups
    - others
  - Step 2. checks if the matching entry contains sufficient permissions

Windows access control lists

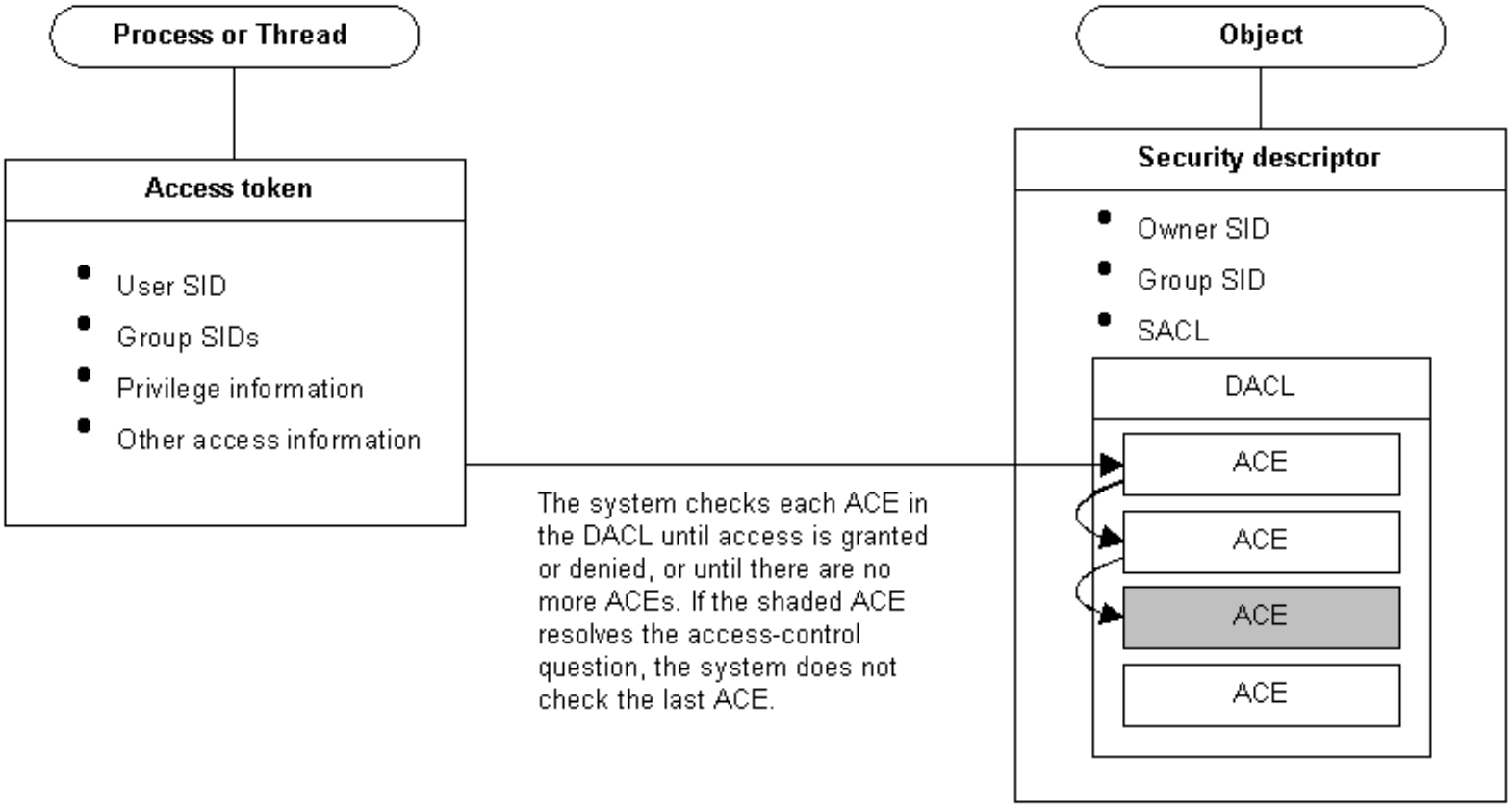
- Different sets of rights
  - Basic: read, write, execute, delete, change permission, take ownership
  - Generic: no access, read (read/execute), change (read/write/ execute/delete), full control (all), special access (assign any of the basics)
  - Directory: no access, read (read/execute files in directory), list, add, add and read, change (create, add, read, execute, write files; delete subdirectories), full control, special access

Windows access control lists

- Examine ACEs until
  - An ACE explicitly denies *any* of access rights requested
  - One or more ACEs explicitly grant *all* the requested rights
  - All ACEs are checked
    - access is implicitly denied for requested right that are not explicitly allowed



Windows access control lists



Mandatory access control

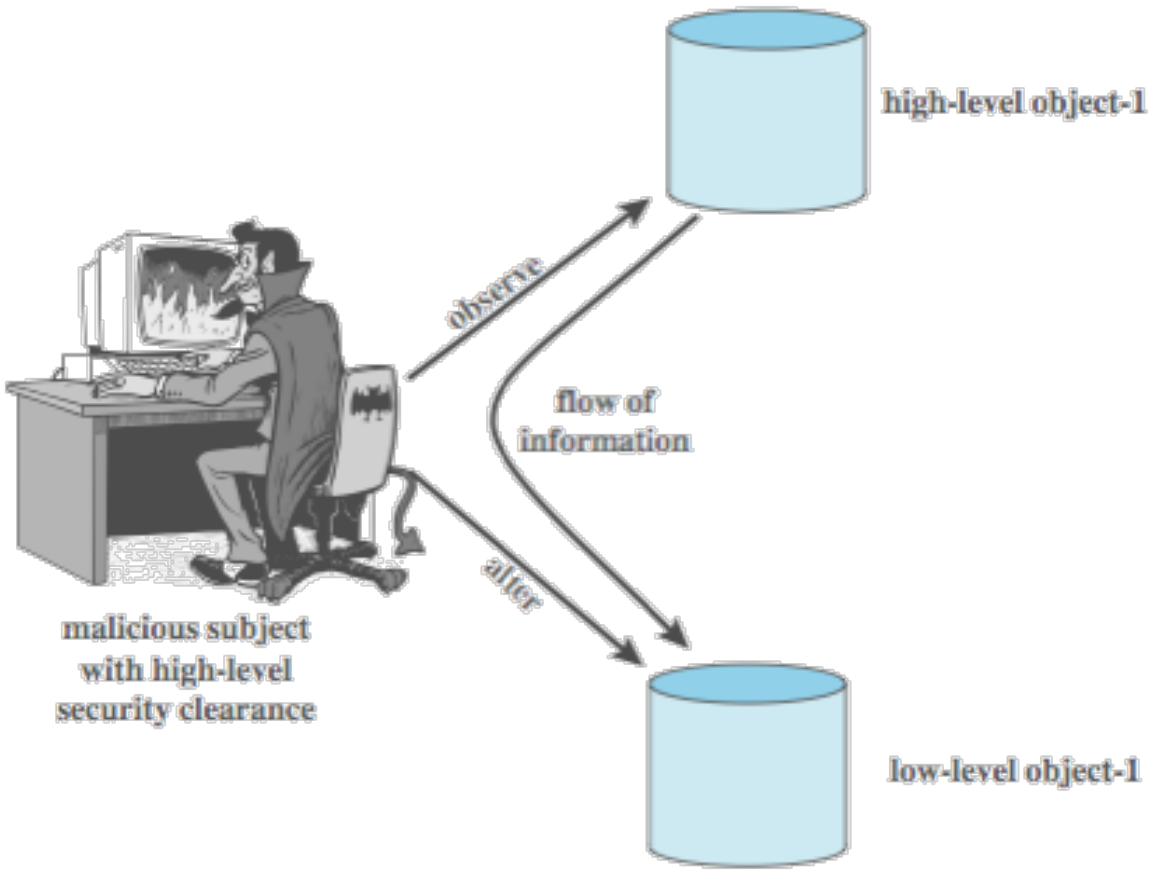
Formal computer security models

- Two fundamental computer security facts
  - All complex software systems have flaw/bugs
  - Is extraordinarily difficult to build computer hardware/software not vulnerable to attack
- Hence desire to prove design and implementation satisfy security requirements
- Led to development of formal security models
  - Initially funded by US DoD
- Bell-LaPadula (BLP) model very influential

The BLP model

- Developed in 1970s as a formal access control model
- Subjects and objects have a security class
  - top secret > secret > confidential > unclassified
  - subject has a security clearance level
  - object has a security classification level
  - class control how subject may access an object
- Applicable if have info and user categories

Multi-level security



Formal BLP description

- Based on current state of system (M, b, f, H)
  - M = access matrix
  - b = current access set
  - f = level function
  - H = hierarchy
- BLP give formal theorems
  - Theoretically possible to prove system is secure
  - In practice usually not possible

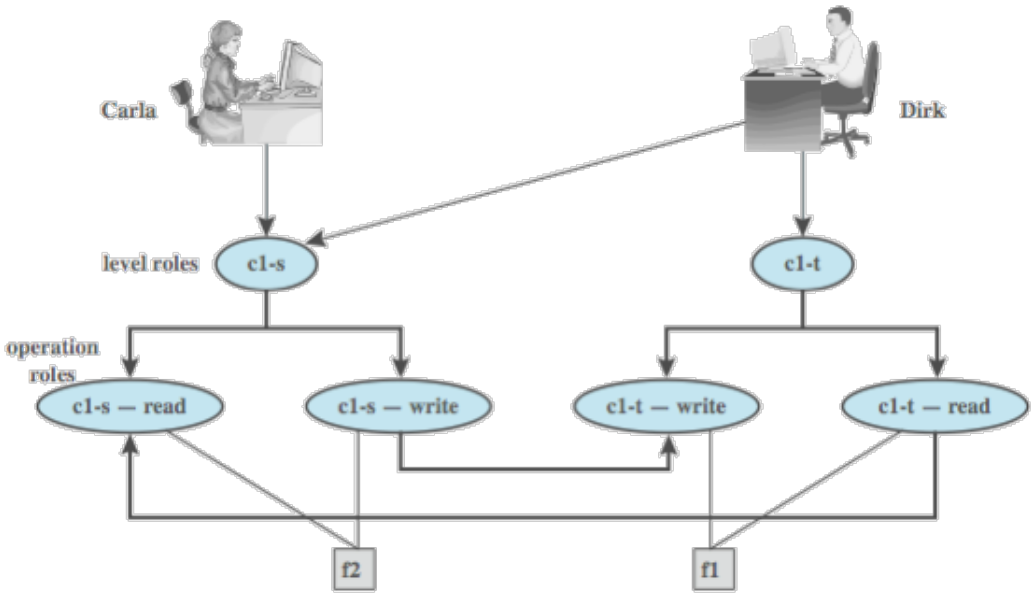
Formal BLP description

- Three BLP properties
  - ss-property: Every triple of the form (S_i, O_j, read) in the current access set b has the property $f_c(S_i) \geq f_o(O_j)$
  - *-property: Every triple of the form (S_i, O_j, append) in the current access set b has the property $f_c(S_i) \leq f_o(O_j)$; every triple of the form (S_i, O_j, write) in the current access set b has the property $f_c(S_i) = f_o(O_j)$
  - ds-property: If (S_i, O_j, A_x) is a current access (is in b), then access mode A_x is recorded in the (S_i, O_j) element of M. That is, (S_i, O_j, A_x) implies that $A_x \in M[S_i, O_j]$

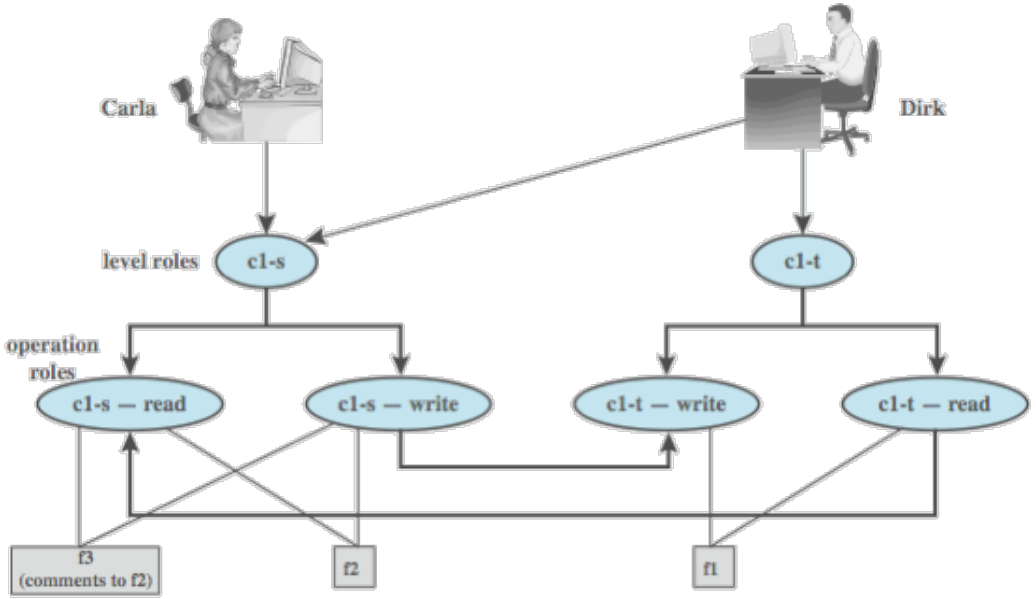
The BLP rules

- Get/release access
- Change object/current level
- Give/rescind access permission
- Create an object
- Delete a group of objects

A BLP example

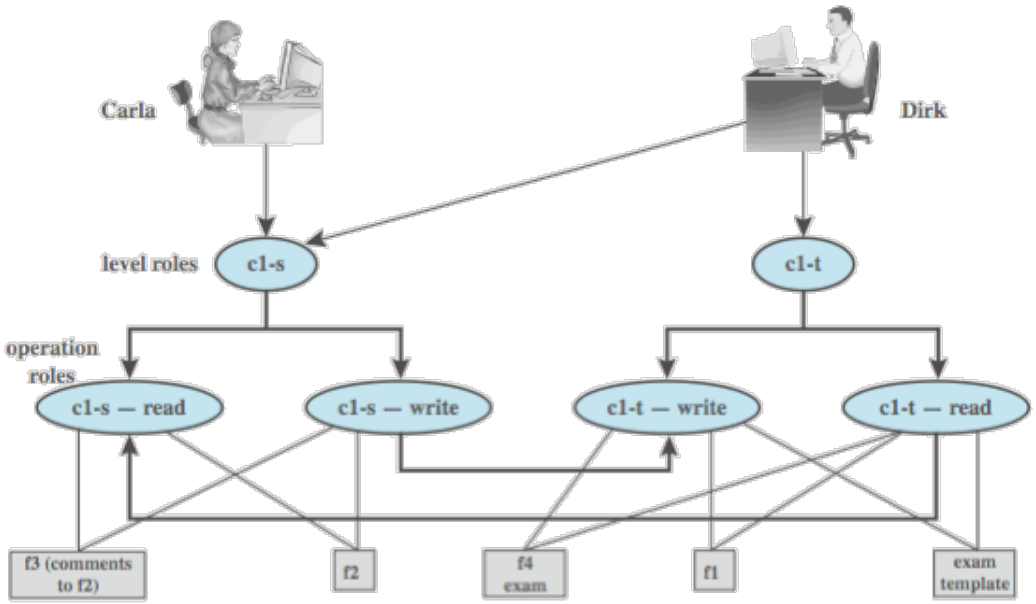


(a) Two new files are created: f1: c1-t; f2: c1-s

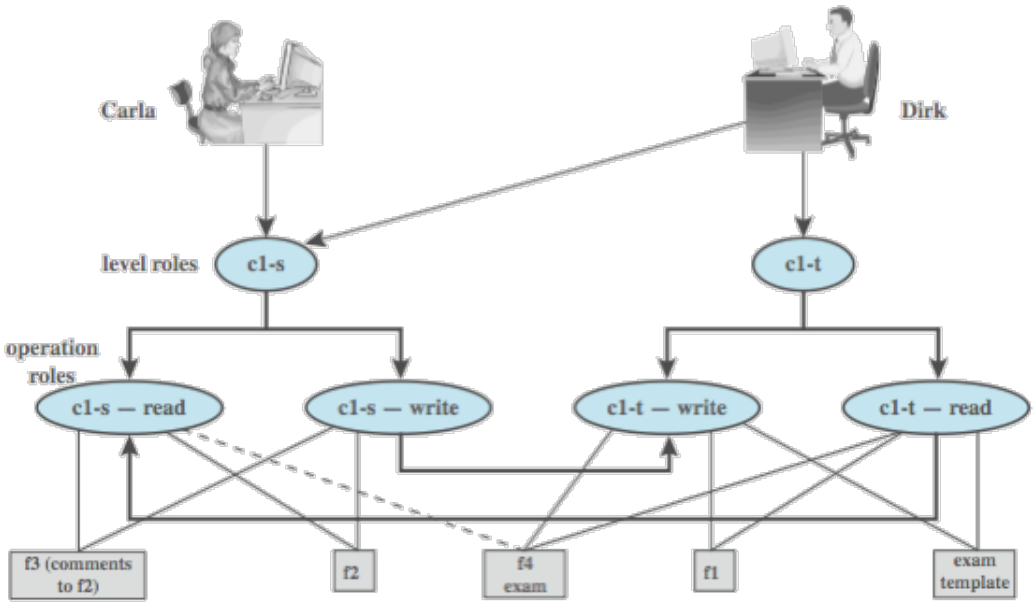


(b) A third file is added: f3: c1-s

A BLP example

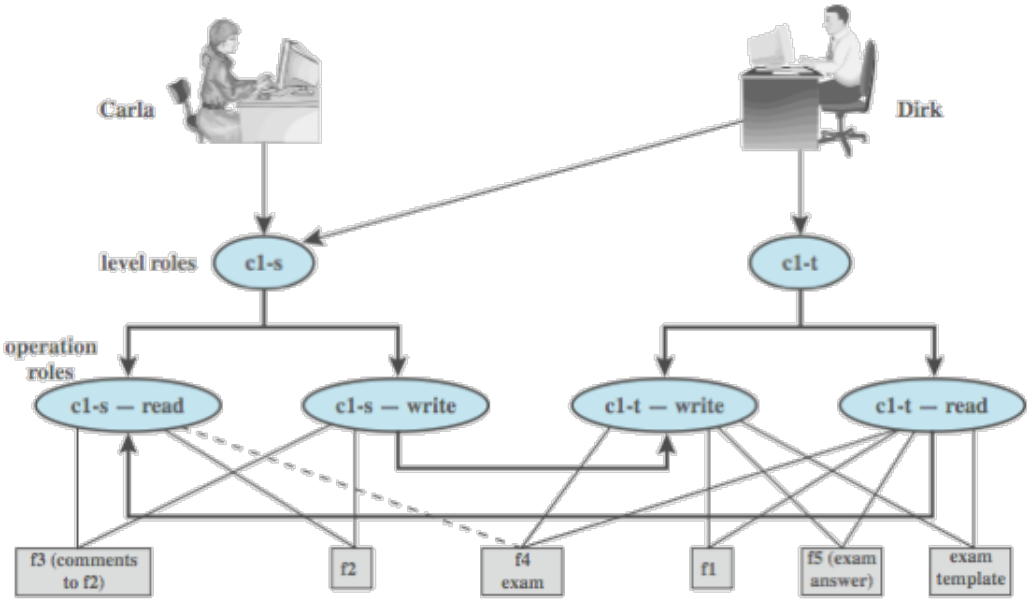


(c) An exam is created based on an existing template: f4: c1-t



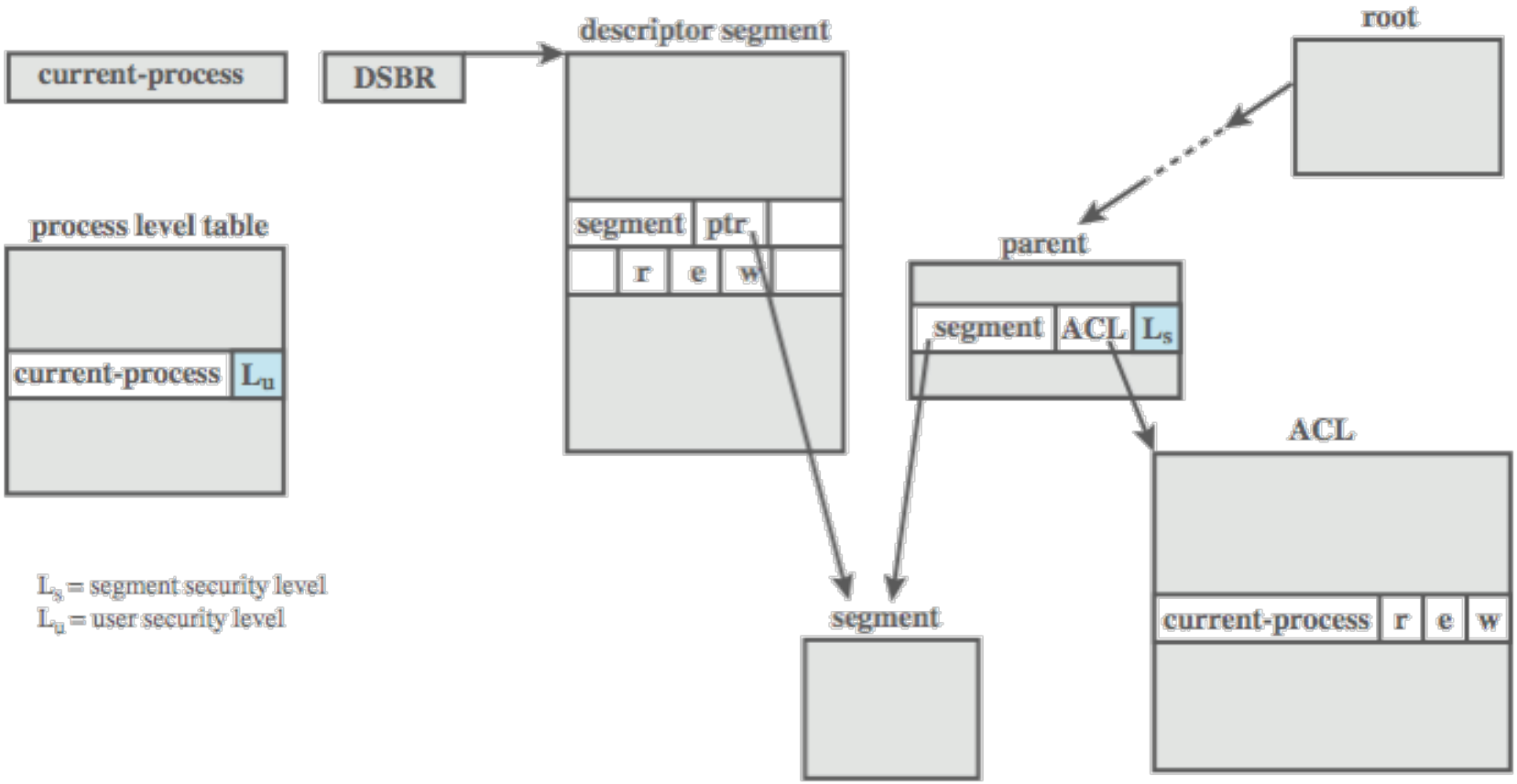
(d) Carla, as student, is permitted acess to the exam: f4: c1-s

A BLP example



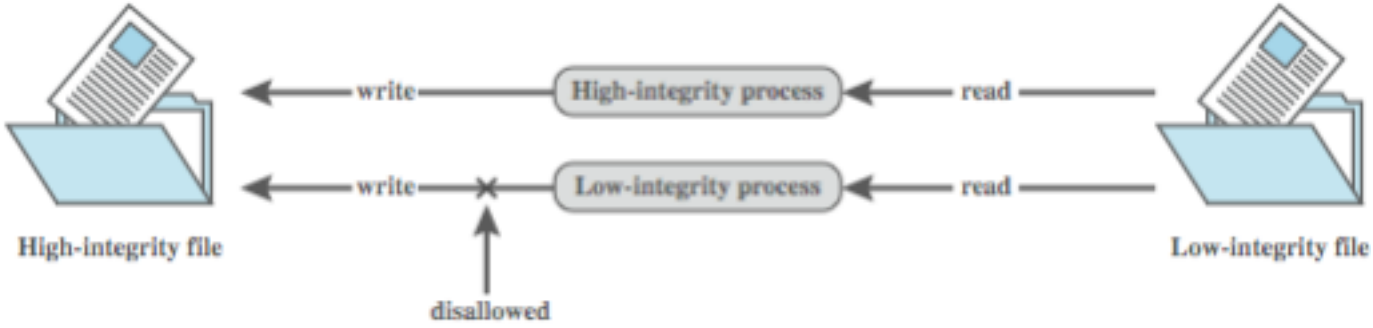
(e) The answers given by Carla are only accessible for the teacher: f5: c1-t

MULTICS example



Biba integrity model

- Περιλαμβάνει διάφορα μοντέλα που ορίζουν κανόνες για την διασφάλιση της ακεραιότητας



- Γενικά, όσο πιο υψηλό το επίπεδο της ακεραιότητας, τόσο μεγαλύτερη η εμπιστοσύνη
  - Υπάρχει διαφορά μεταξύ επιπέδων ακεραιότητας και ασφάλειας

Biba: low water mark model

- Θεώρημα
  - Αν υπάρχει μονοπάτι μεταφοράς πληροφορίας από το $o \in O$ στο $o' \in O$, η επιβολή του low water mark απαιτεί $i(o') \leq i(o)$
- Κανόνες
  - Το $s \in S$ μπορεί να διαβάσει ένα $o \in O$ και $i(s) := \min\{ i(s), i(o) \}$
  - Το $s \in S$ μπορεί να γράψει σε ένα $o \in O$ αν και μόνο αν $i(o) \leq i(s)$
  - Το $s \in S$ μπορεί να εκτελέσει το $s' \in S$ αν και μόνο αν $i(s') \leq i(s)$

Biba: ring model

- Γενικές αρχές
  - Το επίπεδο ακεραιότητας ενός υποκειμένου είναι στατικό
  - Τα άλλα χαρακτηριστικά παραμένουν τα ίδια
- Κανόνες
  - Το $s \in S$ μπορεί να διαβάσει ένα $o \in O$ (δηλαδή, κάθε $o \in O$)
  - Το $s \in S$ μπορεί να γράψει σε ένα $o \in O$ αν και μόνο αν $i(o) \leq i(s)$
  - Το $s \in S$ μπορεί να εκτελέσει το $s' \in S$ αν και μόνο αν $i(s') \leq i(s)$

Biba: strict integrity model

- Όταν αναφερόμαστε στο μοντέλο Biba εννοούμε αυτό
  - Είναι το δυικό του μοντέλου Bell-LaPadula
  - Εφαρμογή στο λειτουργικό σύστημα LOCUS
- Κανόνες
  - Το $s \in S$ μπορεί να διαβάσει ένα $o \in O$ αν και μόνο αν $i(s) \leq i(o)$
  - Το $s \in S$ μπορεί να γράψει σε ένα $o \in O$ αν και μόνο αν $i(o) \leq i(s)$
  - Το $s \in S$ μπορεί να εκτελέσει το $s' \in S$ αν και μόνο αν $i(s') \leq i(s)$

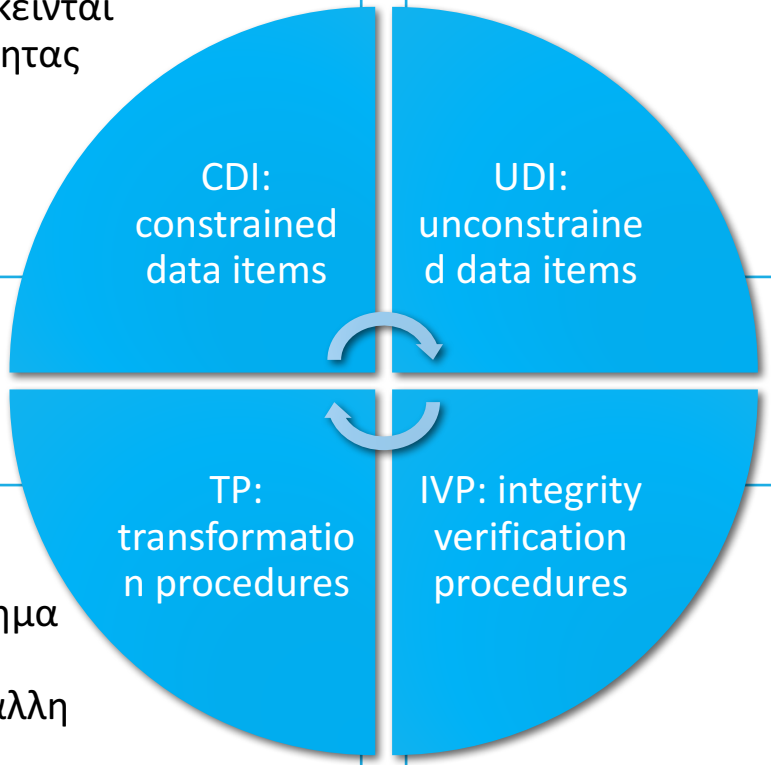
Clark-Wilson integrity model

- Ορίστηκε το 1987 από τους David Clark και David Wilson
  - Χρησιμοποιεί την έννοια του transaction ως βασική λειτουργία
  - Η ακεραιότητα ορίζεται με ένα σύνολο από περιορισμούς
- Π.χ. η ιδιότητα ακεραιότητας TB = YB + D – W σε τράπεζα
  - D = ημερήσιο ποσό κατάθεσης
  - W = ημερήσιο ποσό ανάληψης
  - YB = αρχικό ποσό στην τράπεζα
  - TB = τελικό ποσό στην τράπεζα

Clark-Wilson integrity model

- Δεδομένα που υπόκεινται σε έλεγχο ακεραιότητας

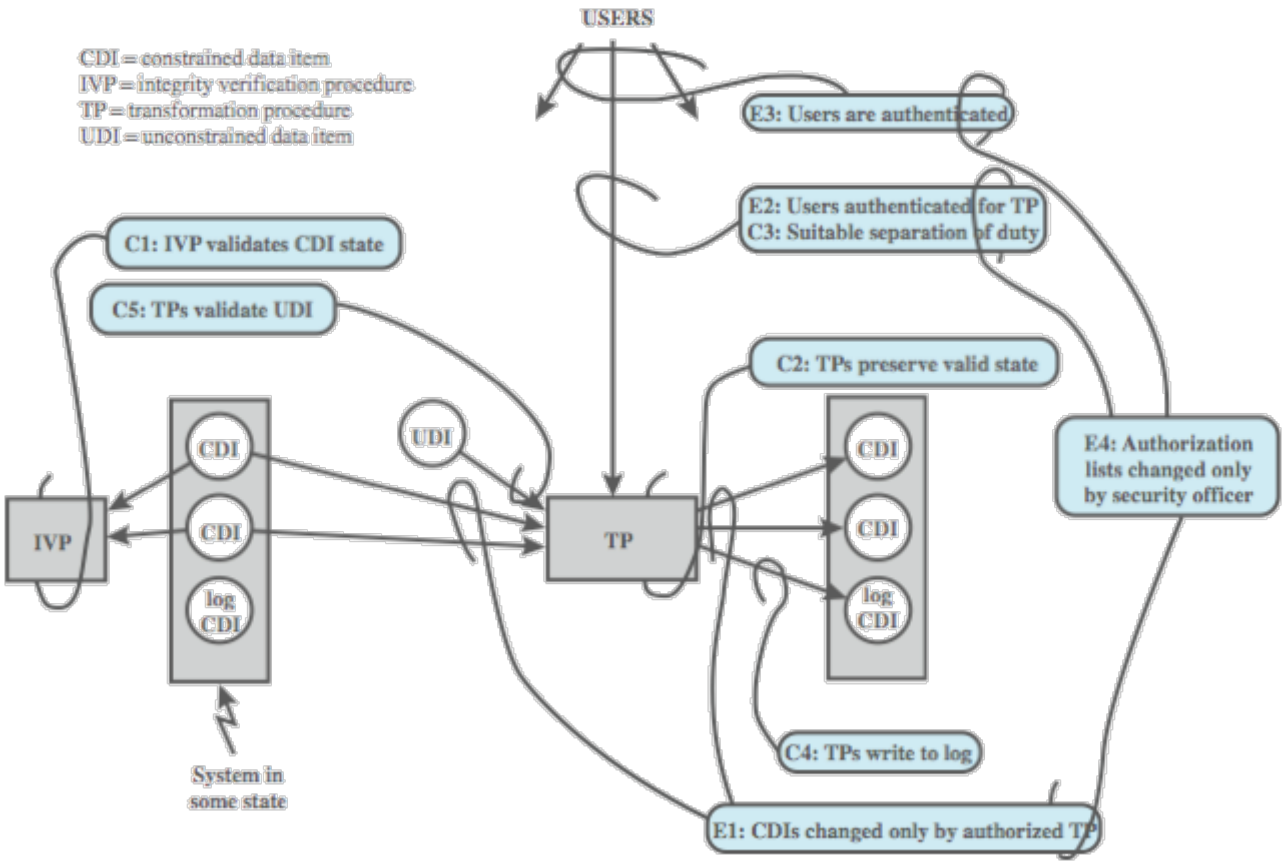
- Δεδομένα που δεν υπόκεινται σε έλεγχο ακεραιότητας



- Διαδικασίες που πηγαίνουν το σύστημα από τη μια έγκυρη κατάσταση σε μια άλλη (σωστή υλοποίηση)

- Διαδικασίες που ελέγχουν ότι τα CDIs ικανοποιούν τους περιορισμούς ακεραιότητας

Clark-Wilson integrity model



Clark-Wilson certification rules

- Όταν μια διαδικασία IVP εκτελείται, πρέπει να εξασφαλίζει ότι όλα τα δεδομένα CDIs είναι έγκυρα
- Μια διαδικασία TP πρέπει να μετατρέπει ένα σύνολο από σχετιζόμενα δεδομένα CDIs από μια έγκυρη κατάσταση σε μια άλλη έγκυρη
  - Συσχετισμός CDIs με TPs (ένα TP πρέπει να έχει πιστοποιηθεί ότι λειτουργεί με CDI)

Clark-Wilson certification rules

- Οι επιτρεπόμενες σχέσεις πρέπει να πληρούν τις απαιτήσεις της αρχής διαχωρισμού καθήκοντος
- Όλες οι διαδικασίες TPs πρέπει να προσαρτούν αρκετή πληροφορία, σε δεδομένα μόνο-προσάρτησης CDI, για να ανακτηθεί η πράξη
- Κάθε διαδικασία TP που δέχεται είσοδο UDI μπορεί να εκτελέσει μόνο έγκυρους μετασχηματισμούς (ή να μην εκτελέσει), για όλες τις δυνατές τιμές του UDI
  - Ο μετασχηματισμός είτε απορρίπτει το UDI ή το μετατρέπει σε CDI

Clark-Wilson enforcement rules

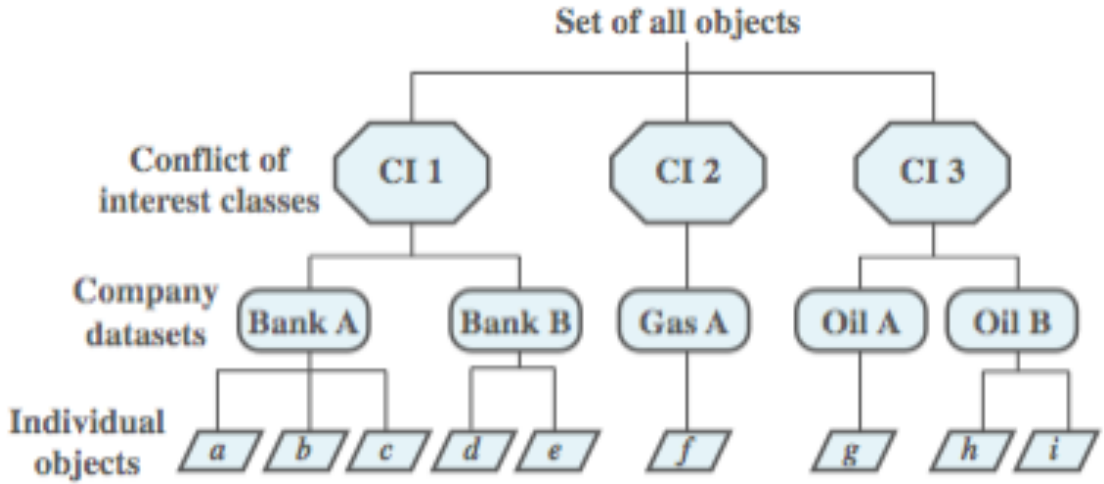
- Το σύστημα πρέπει να διατηρεί πιστοποιημένες σχέσεις και πρέπει να εξασφαλίζει ότι μόνο διαδικασίες TPs, που είναι πιστοποιημένες να χρησιμοποιούν δεδομένα CDI, μπορούν να χειρίζονται αυτά τα CDI
  - Δηλ. αν f είναι TP, o είναι CDI, και C είναι το σύνολο πιστοποιημένων σχέσεων, τότε πρέπει (f, o) $\in$ C
- Το σύστημα πρέπει να σχετίζει έναν χρήστη με κάθε διαδικασία TP και ένα σύνολο από CDIs
  - Η διαδικασία TP μπορεί να έχει πρόσβαση σε αυτά τα CDIs για λογαριασμό τα του χρήστη, δηλ. { user, TP, { CDI set } } = allowed_A

Clark-Wilson enforcement rules

- Το σύστημα πρέπει να αυθεντικοποιήσει κάθε χρήστη που επιχειρεί να εκτελέσει μια διαδικασία TP
  - Δεν απαιτεί αυθεντικοποίηση όταν ο χρήστης εισέρχεται στο σύστημα (login) καθώς μπορεί να έχει πρόσβαση σε UDIs
- Μόνο ένας πιστοποιών μιας διαδικασίας TP μπορεί να αλλάξει τη λίστα των οντοτήτων που σχετίζονται με την TP
  - Κανένας πιστοποιών μιας TP, ή μιας οντότητας σχετιζόμενη με την TP, μπορεί να έχει άδεια εκτέλεσης σχετιζόμενη με αυτή την TP

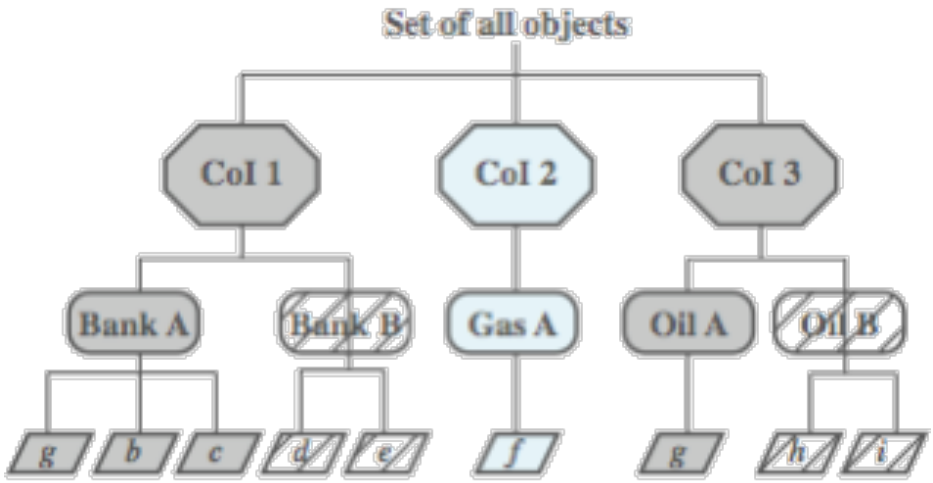
Chinese wall model

- Οργανώνει τις οντότητες σε κλάσεις αντικρουόμενων συμφερόντων: conflict of interest (CI) classes

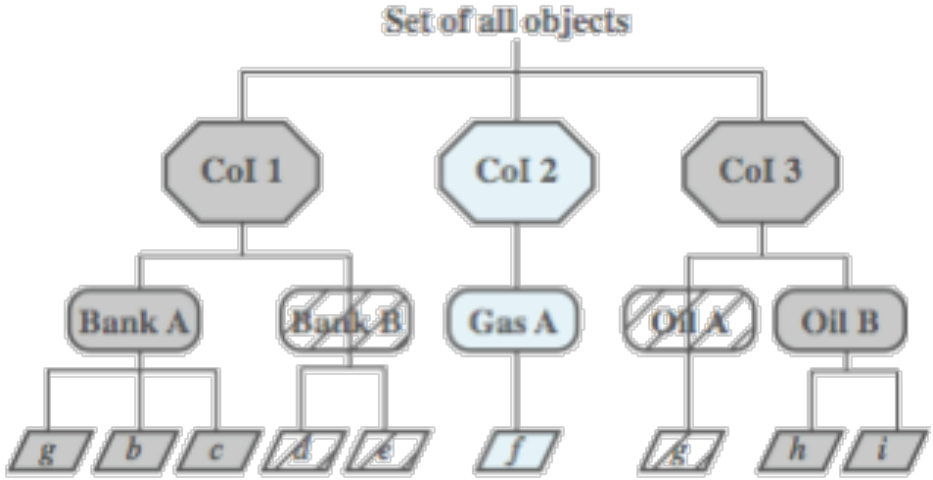


(a) Example set

Chinese wall model



(b) John has access to Bank A and Oil A

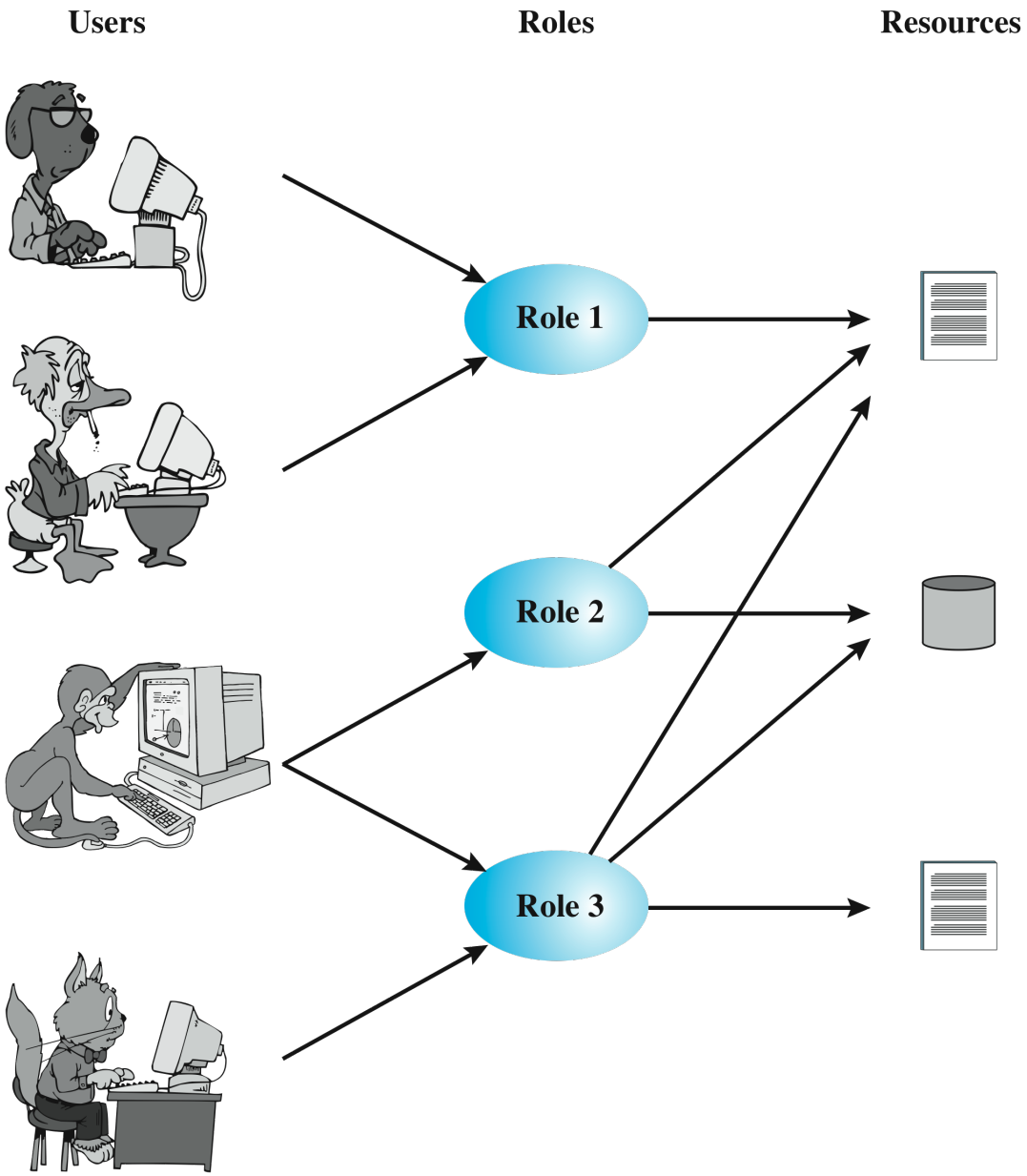


(c) Jane has access to Bank A and Oil B

Role-based access control

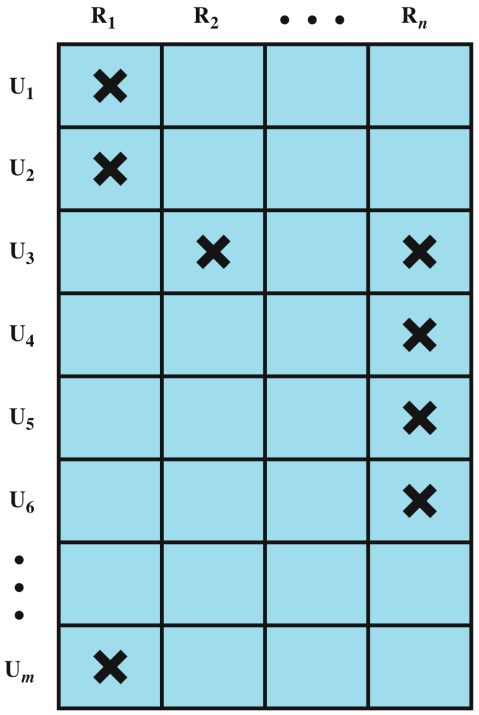
Role-based access control (RBAC)

Users, roles, and resources



Access control matrix

The access control matrix representation of RBAC

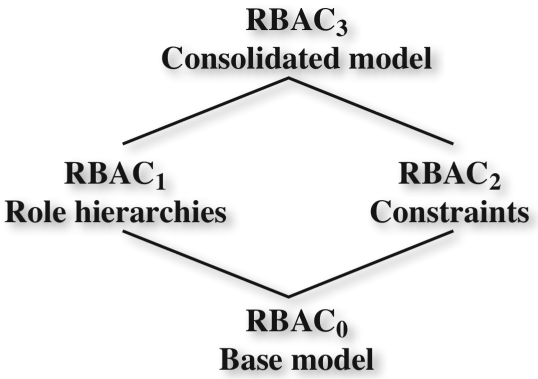


OBJECTS

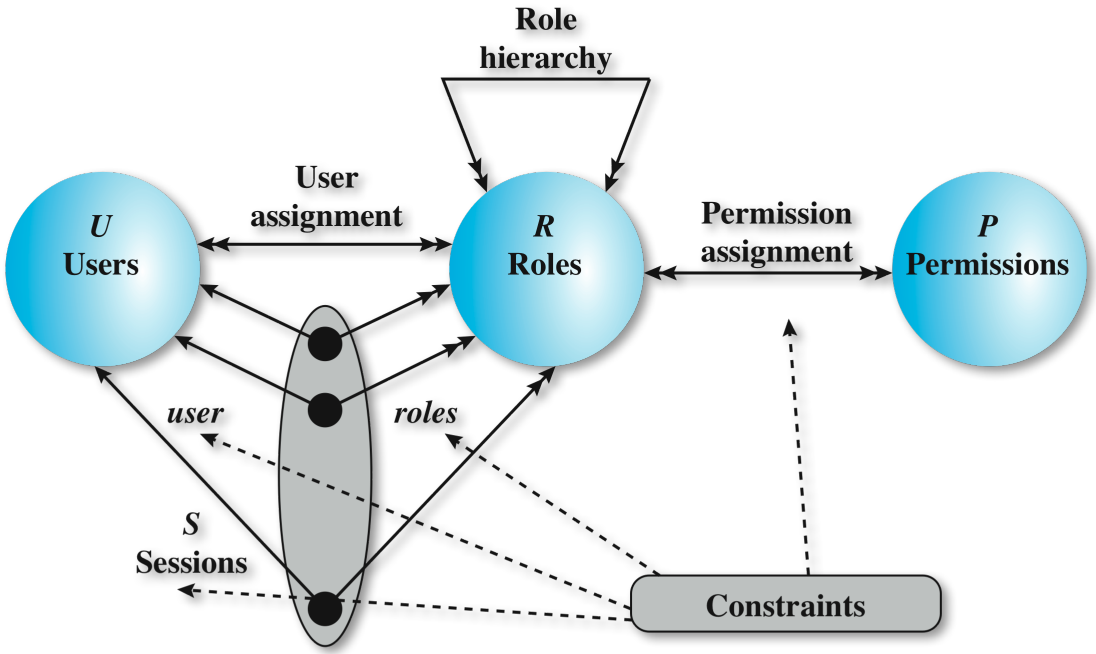
ROLES	R_1	R_2	R_n	F_1	F_1	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
R_n			control		write	stop			

Family of RBAC models

- $RBAC_0$ is the minimum requirement for an RBAC system.
- $RBAC_1$ adds role hierarchies
- $RBAC_2$ adds constraints
- $RBAC_3$ includes $RBAC_1$ and $RBAC_2$



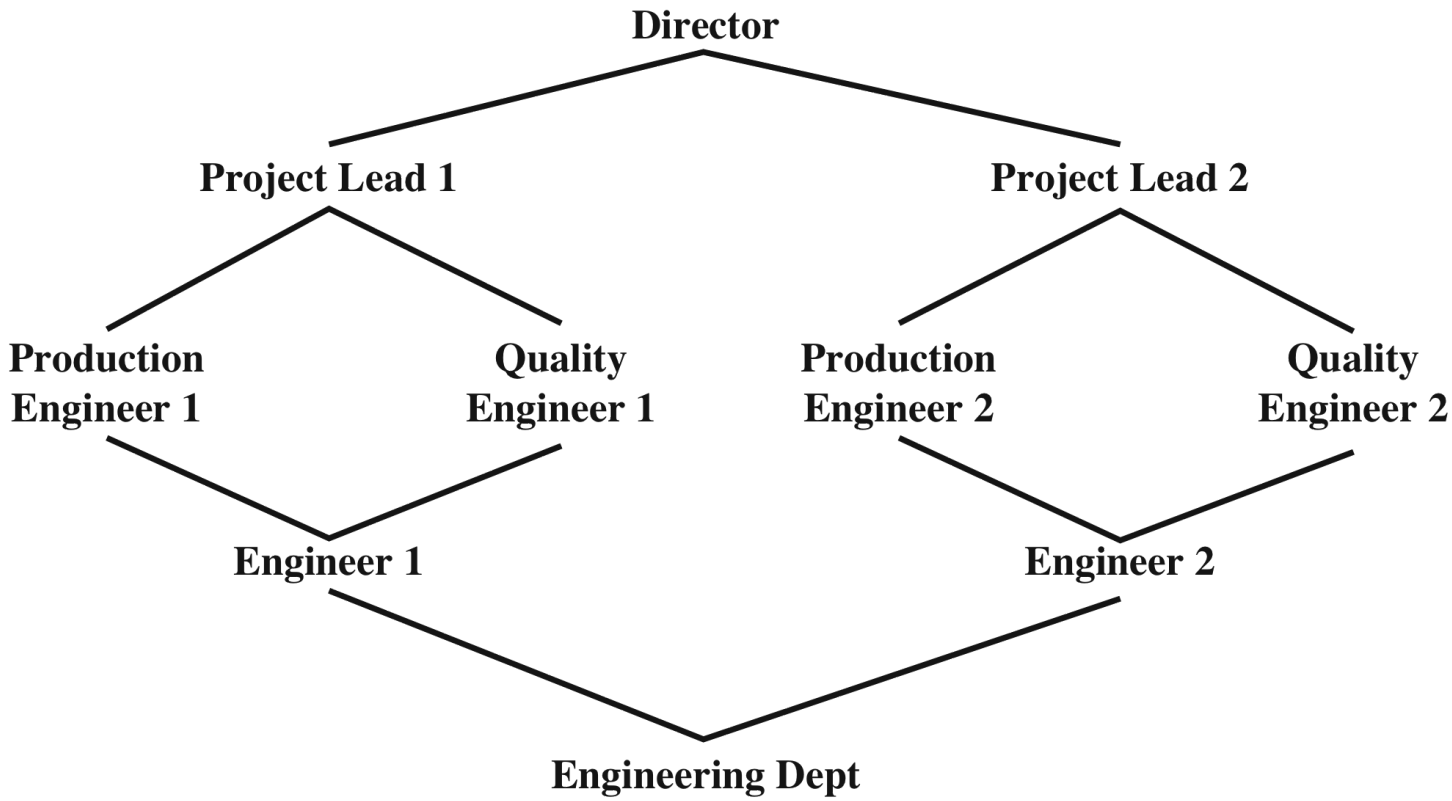
(a) Relationship among RBAC models



Scope of RBAC models

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

Example of role hierarchy



RBAC constraints

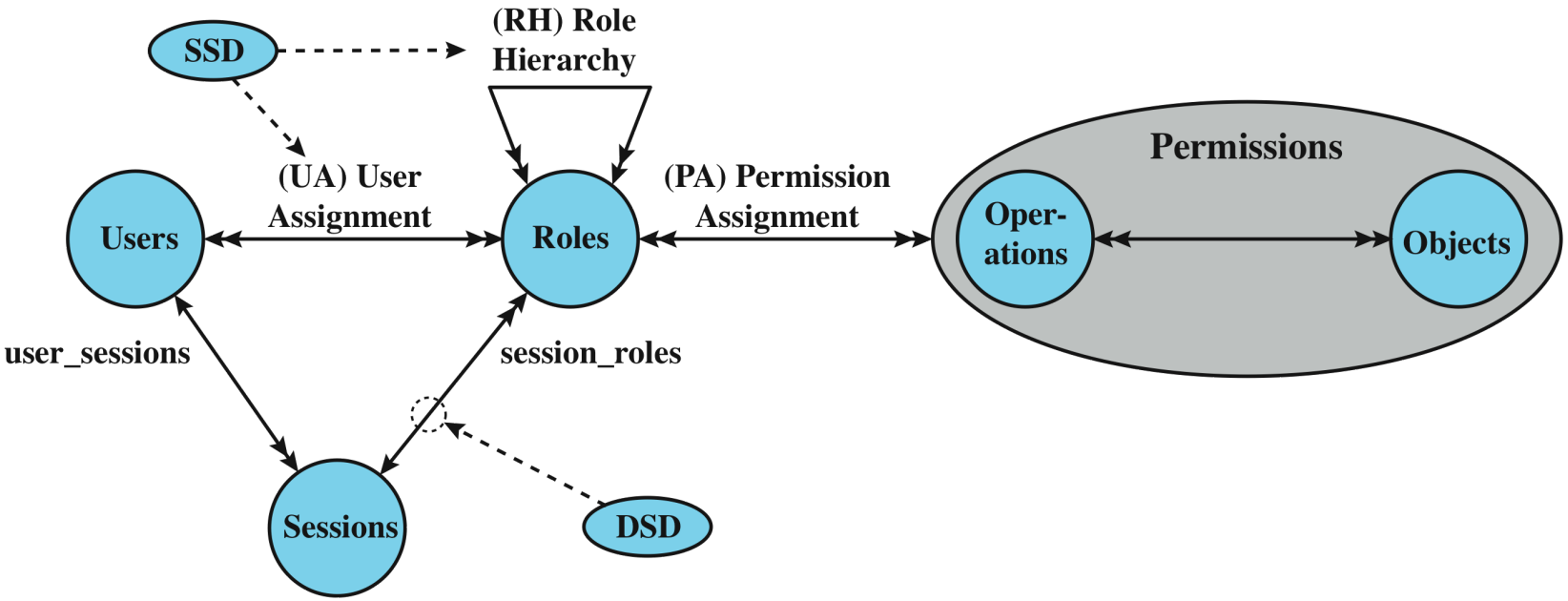
- Provide a means of adapting RBAC to the specifics of administrative and security policies of an organization
- A relationship among roles or a condition related to roles

mutually exclusive roles	cardinality	prerequisite roles
• A user can only be assigned to one role (in a session or statically) • Any permission can be granted to only one role in the set	• Setting a maximum number with respect to roles	• A user can only be assigned to a particular role if it is already assigned to some other specified role

RBAC functional specification

- provide the capability to create, delete, and maintain RBAC elements and relations
- provide functions for session management and for making access control decisions
- provide the capability to perform query operations on RBAC elements and relations

RBAC model of NIST



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

Basic definitions

- Object
  - Any system resource subject to access control, like a file, printer, terminal, database record
- Operation
  - An executable image of a program, which upon invocation executes some function for the user
- Permission
  - An approval to perform an operation on one or more RBAC protected objects

Core RBAC

administrative functions

- add and delete users from the set of users
- add and delete roles from the set of roles
- create and delete instances of user-to-role assignment
- create and delete instances of permission-to-role assignment

supporting system functions

- create a user session with a default set of active roles
- add an active role to a session
- delete a role from a session
- check if the session subject has permission to perform a request operation on an object

review functions

- enable an administrator to view but not modify all the elements of the model and their relations

Hierarchical RBAC

General role hierarchies

- Allow an arbitrary partial ordering of the role hierarchy
- Supports multiple inheritance: a role can inherit permissions from many subordinate roles; many roles can inherit from the same subordinate role

Limited role hierarchies

- Impose restrictions resulting in a simpler tree structure
- Role may have one or more immediate ascendants but is restricted to a single immediate descendant

SSD: static separation of duty

- Enables the definition of a set of mutually exclusive roles
  - If a user is assigned to one role in the set, the user may not be assigned to any other role in the set
- Can place a cardinality constraint on a set of roles
- Defined as a pair (role set *n*) where no user is assigned to *n* or more roles from the role set
- Includes administrative functions for creating and deleting role sets and adding and deleting role members
- Has review functions for viewing SSD properties

DSD: dynamic separation of duty

- Limit the permissions available to a user
- Places constraints on the roles that can be activated within or across a user's sessions
- Define constraints as a pair (role set *n*)
  - $n \leq 2$ is a natural number with the property that no user session may activate *n* or more roles from the role set
- Enables the administrator to specify certain capabilities for a user at different, non-overlapping spans of time
- Has review and administrative functions for DSD relations

Functions and roles for banking example

(a) functions and official positions

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

Functions and roles for banking example

(b) permission assignments

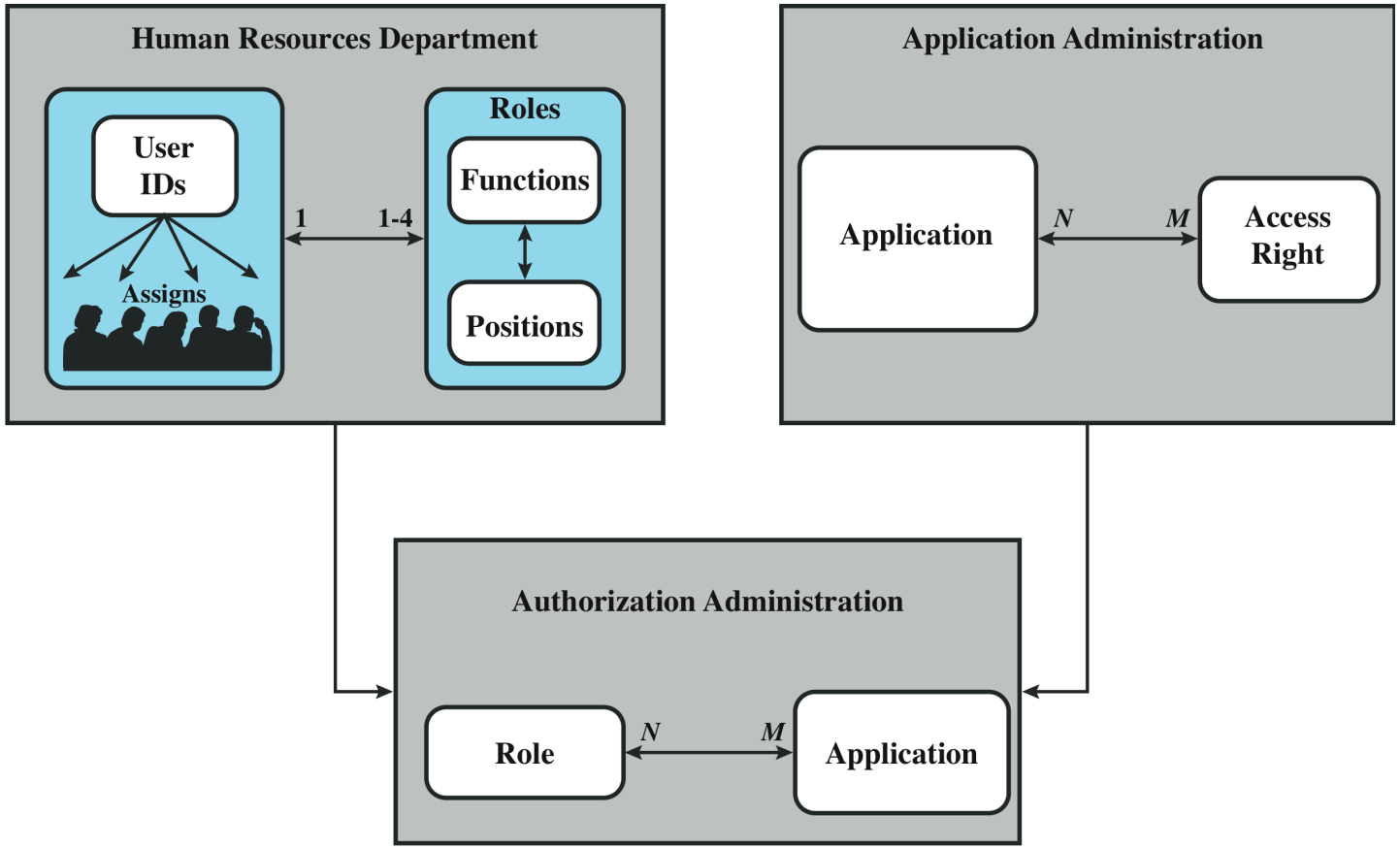
Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	1, 2, 3, 4, 7
	derivatives trading	1, 2, 3, 7, 10, 12, 14
	interest instruments	1, 4, 8, 12, 14, 16
	private consumer instruments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

Functions and roles for banking example

(c) PA with inheritance

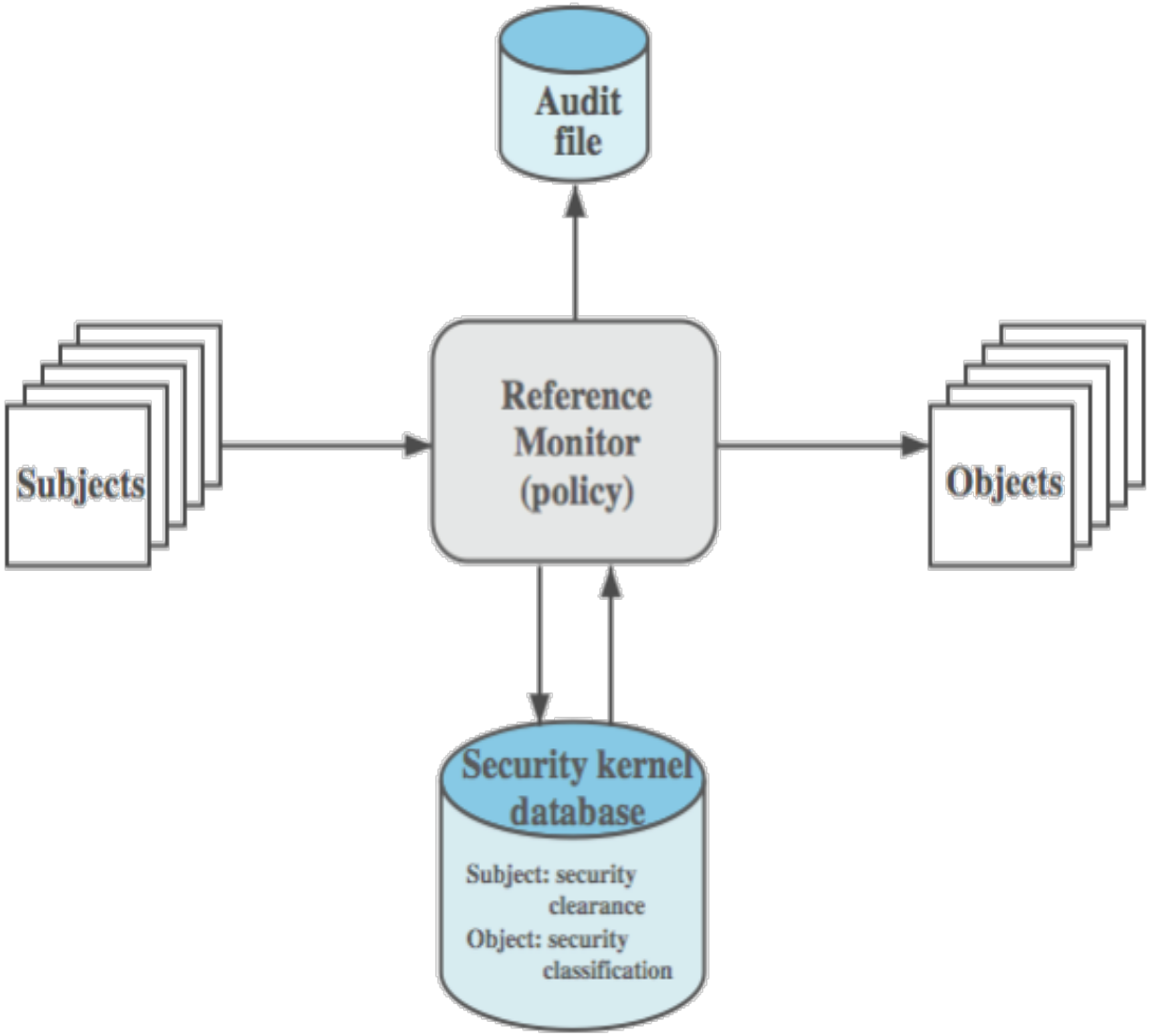
Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	7
	derivatives trading	14
	private consumer instruments	1, 2, 4, 7
• • •	• • •	• • •

Access control admin. example

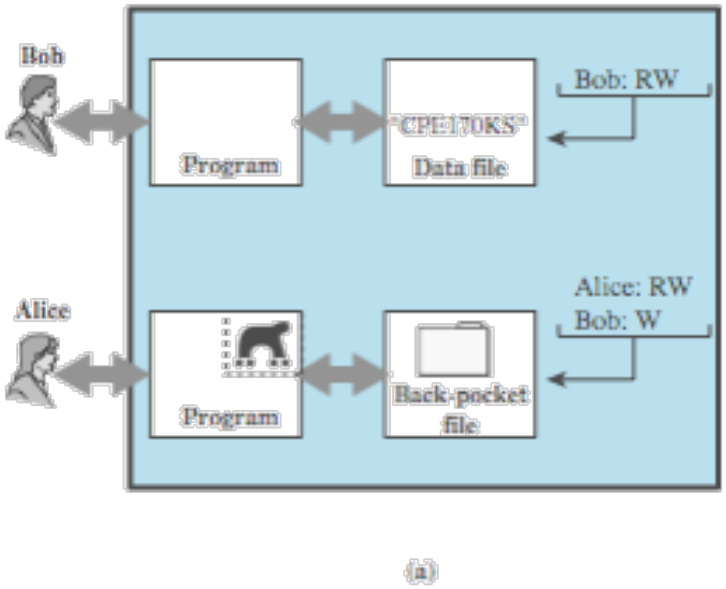


Trusted systems

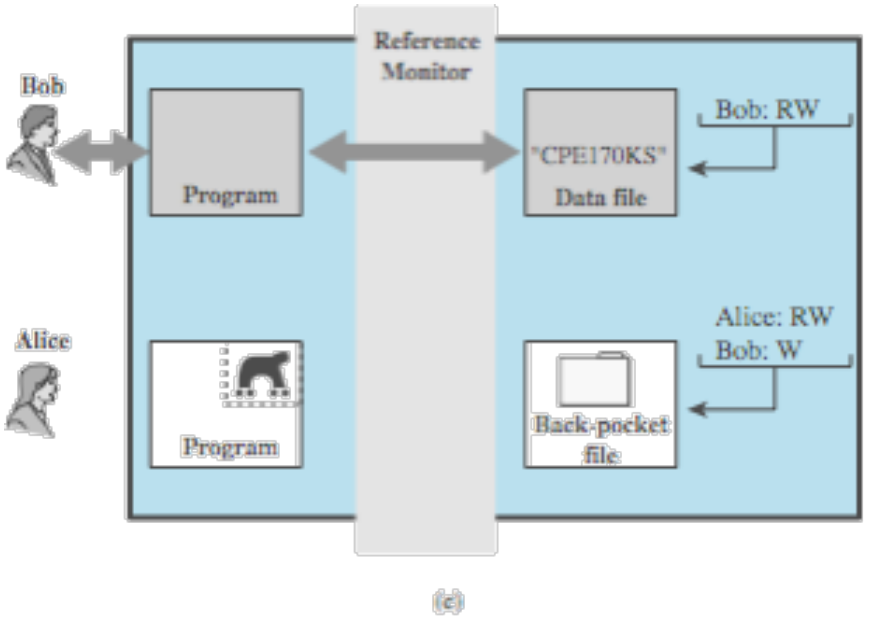
Reference monitor



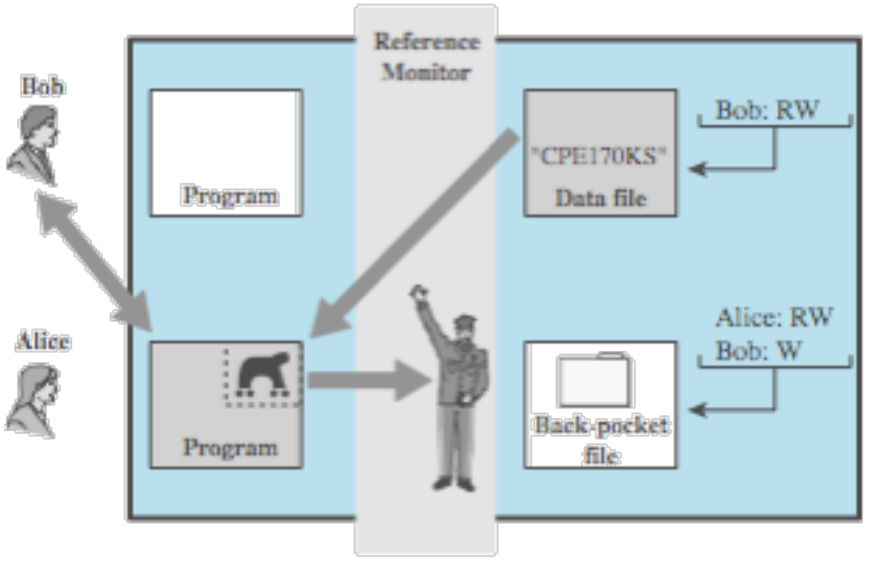
Trojan horse defense



Trojan horse defense



(c)



(d)

Προτεινόμενη βιβλιογραφία

- W. Stallings
  Cryptography and Network Security: Principles & Practice
  7th Ed., Prentice Hall, 2017
- W. Stallings and L. Brown
  Computer Security: Principles & Practice
  3rd Ed., Prentice Hall, 2015
- M. Bishop
  Computer Security: Art and Science
  Addison Wesley, 2003