Topic 11: Access Control

Study mechanisms for controlling access to resources

Sources:

Computer Security by Matt Bishop.

Related information is also available on the Internet, e.g. http://csrc.nist.gov/staff/Kuhn/towards-std.pdf.

The RBAC slides are from Prof. Ravi Sandhu's slides; Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, "Role-Based Access Control Models." IEEE Computer, Volume 29, Number 2, February 1996, pages 38-47.

Overview

- Part 1
  - Basic concepts
- Part 2
  - Discretionary Access Control (DAC)
- Part 3
  - Mandatory Access Control (MAC)
- Part 4
  - RBAC (Role Based Access Control)
  - Conclusion

Access Control Basic Concepts

- An access control system is to
  - Keep the bad guys out.
  - Let the good guys in (who can read, who can modify, …).
  - To enforce a specified access control policy preventing unauthorised access to data, services or other resources.

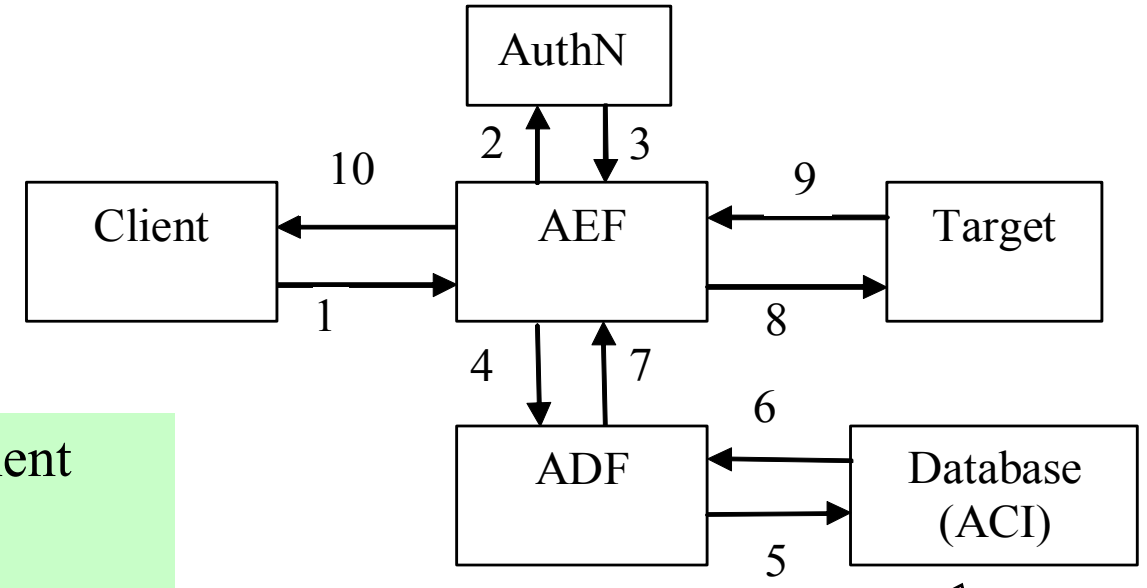
- Resources may be in different forms and the control should be provided at multiple levels:
  - At the system level, e.g. security kernel in an operating system, database management systems (DBMS).
  - At the network level, e.g. firewalls.
  - At physical level, e.g. lockers, biometric scanning.

Access Control Basic Concepts

- Access control = AuthN + AuthZ.
  - Authentication (AuthN) establishes the identity of a subject (*who*).
  - Authorization (AuthZ) specifies and enforces that each object is accessed correctly and only by those that are allowed to do so
    - An **access control policy** specifies *who* (subject) can perform what access operations (access types/modes) on what (object).
    - These rules are enforced at run-time by an AuthZ Decision Engine.

Access Control Basic Concepts

Identification: Who I am.
AuthN: Prove it!
AuthZ: Decide if you have the right permission.



- AEF = Access control Enforcement Function (enforce AC decision);
- ADF = Access control Decision Function (grant/deny based on AC policy);
- ACI = Access Control Information (AC policy).
- Target = resources (data, services, etc)

A simple AC rule (non-RBAC):
<Alice, Write, FileA>
Another simple AC rule (RBAC):
<Alice, Staff>, <Staff, Write, FileA>

Access Control Basic Concepts

- A subject: an active entity requesting for resource access
  - The identifier (ID) of a user.
  - The ID of a process/software executing programs on behalf of a user.
  - A role - a group of users, a collection of privileges or a functional entity in an organisation, etc.

- An object: a passive entity and target of protection, e.g.
  - a file or directory of files, a data structure (e.g. a stack, inter-process messages, network packets).
  - a table of an operating system, instructions (especially privileged instructions).
  - memory, I/O device, host machine, private network, etc.

Access Control Basic Concepts

- Access operations (or access types/modes): permitted operations; resources dependent, e.g.
  - for file access: read, write, execute, append (does not imply read access), delete, …
  - for network access: grant, deny, redirect, …

Access Control Basic Concepts

- Design principles of access control systems:
  - Check every access: every access **by a subject to an object** should be checked; granting the subject the privilege previously does not mean that the subject should retain this privilege indefinitely.
  - Allow least privilege: subjects should be given just enough privileges to perform their tasks; unnecessary access should not be allowed even if this extra access is useless or harmless.
  - Verify acceptable usage: invalid operations should be denied; it is important to check that an operation to be performed on an object is a valid one.

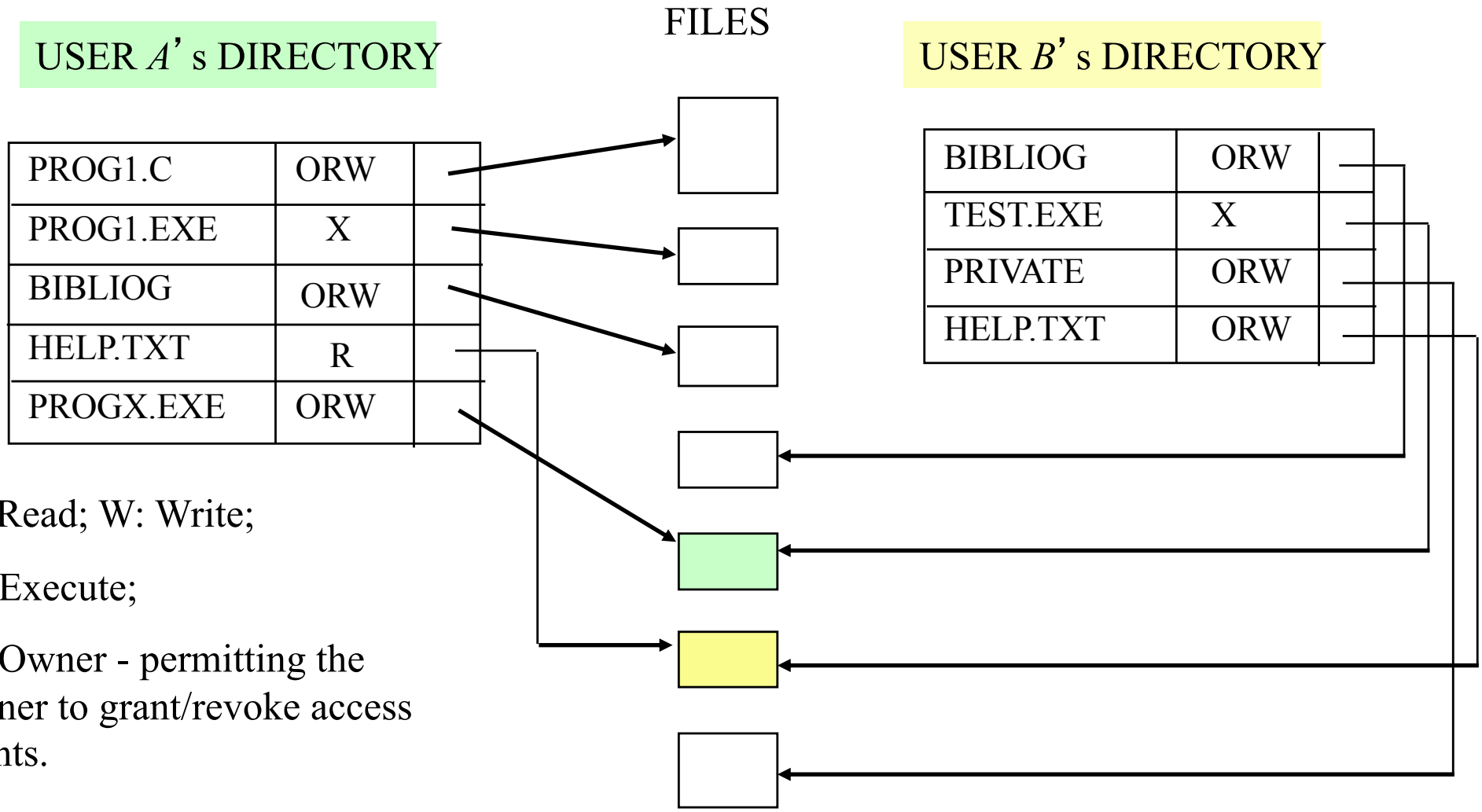
Access Control Basic Concepts

- Access control mechanisms/models
  - Discretionary Access Control (DAC)
    - Identity-based access control
  - Mandatory Access Control (MAC)
    - Classical model: clearance-based access control
  - Role Based Access Control (RBAC)
    - A user's permissions are determined by users' roles, rather than identity or clearance level.
    - Roles can encode arbitrary attributes.
    - Scalable solution by resembling organisational structures in large organisations.

Part 2 Overview

- DAC mechanisms
  - Part 2.1
    - Directory access: lists all the files a user (subject) can access
  - Part 2.2
    - Access control matrix: table of subjects, objects & rights
    - Access control list: an object versus subjects
    - Capability list: a subject versus objects
  - Part 2.3
    - Procedure-oriented access control: information hiding.

DAC - Directory Access



R: Read; W: Write;

X: Execute;

O: Owner - permitting the owner to grant/revoke access rights.

DAC - Directory Access

- Directory access
  - One list per subject, showing all the objects that are allowed to access by the subject;
  - The list can become too large if there are many shared objects;
  - Revocation of access rights can be time-consuming;
  - Pseudonyms or file naming inconsistency;
  - The approach can not efficiently address most object protection scenarios.

DAC - Access Control Matrix

- An entry in the matrix specifies the set of access operations a subject *s* can perform on an object *o*.
- Each column: ACL for one object - defines who can perform what operation.
- Each row: A subject's capability list – defines, for a given subject, what operations allowed on what objects.

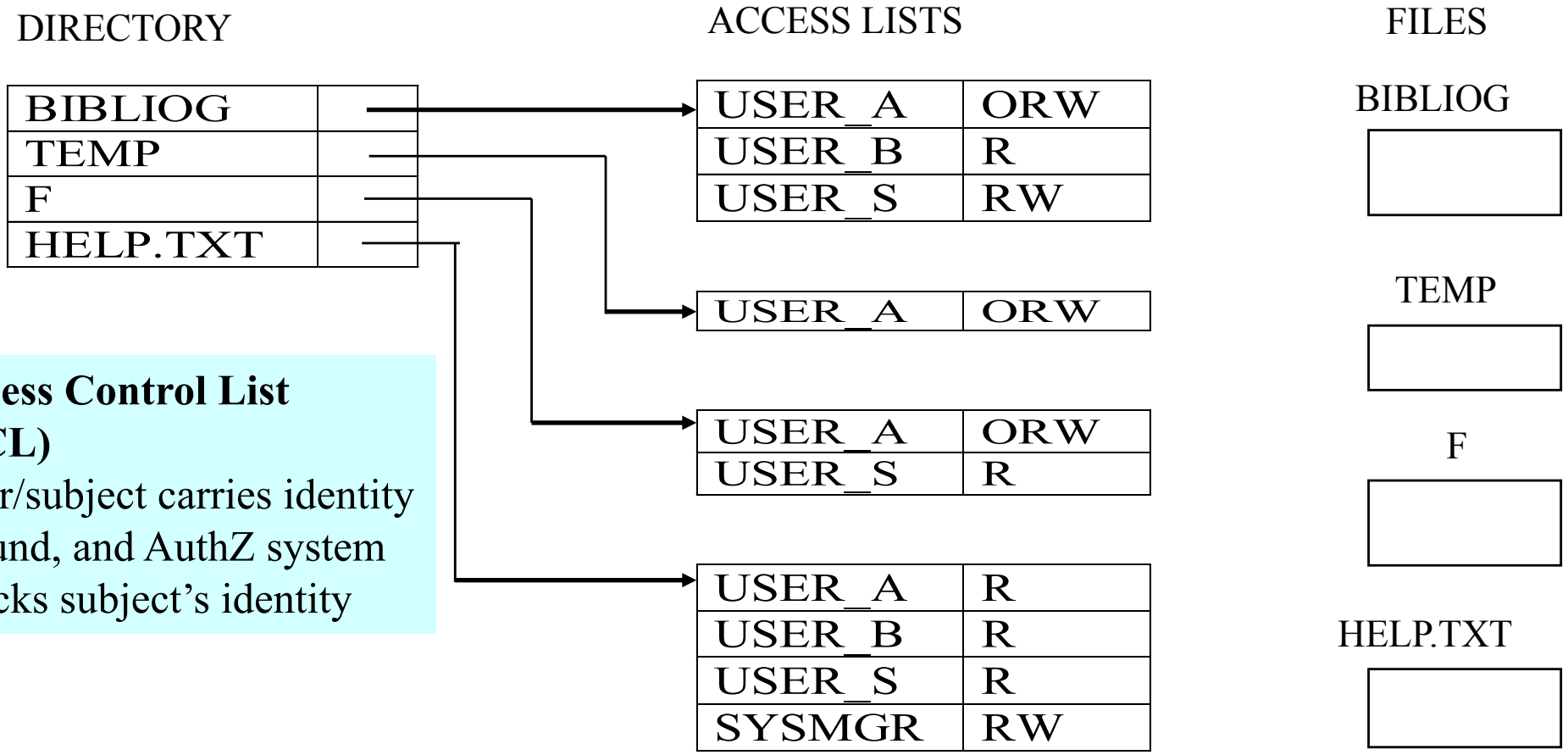
User_B's Capability List

	BIBLIOG	TEMP	F	HELP.TXT
USER_A	ORW	ORW	ORW	R
USER_B	R	-	-	R
USER_S	RW	-	R	R
USER_T	-	-	-	R
SYS_MGR	-	-	-	RW
USER_SV	-	-	-	O

Access Control List (ACL) for F

DAC - Access Control List

- An Object List lists *subjects* and their *access rights*



Access Control List (ACL)
User/subject carries identity around, and AuthZ system checks subject's identity

DAC - Access Control List

- ACL is a decomposition of the matrix by columns.
- One list per object, showing a set of pairs, {*Subjects, Rights*}, indicating the named *subject* can access the associated *object* using any of the *rights*.
- Usually, if a subject is not named, s/he has ***no*** rights over file – the principle of Fail-Safe Default.
- A shared public object can have a very short access list, explicitly naming the few subjects who should have access rights different from the default.
- To shorten a list, specific users can have explicit rights and all other users (defined using 'wildcard') can have a default set of rights.

DAC - Access Control List

- If there are many subjects, you may use groups or wildcards in ACL to shorten the list, e.g.
  - UNICOS:
    - Entries are (*user, group, rights*)
    - If *user* is in *group*, has rights over file
    - '*' is wildcard for *user* and *group*
      - (holly, *, r): holly can read file regardless of her group.
      - (*, groupA, w): anyone in groupA can write file.
  - UNIX
    - Three classes of users: owner, group, others (the rest)
- Rights revocation
  - Owner deletes subject's entries, or rights from subject's entry, in ACL.

DAC – Capability

- A Capability list/ticket is a row from the Access Control Matrix
  - An example:

USER_S	BIBLIOG {RW}	F {R}	HELP.TXT {R}

  - A capability ticket specifies a given subject can perform what operations on what objects.
  - User carries around tickets; each user may have a number of tickets.
  - User may grant rights to another user, and also grant the right to grant rights.
  - Excellent for implementing 'temporary access' - access rights given to a subject are only valid for a certain duration.

DAC – Capability

- Issues
  - Capability revocation
    - Scan all C-lists, remove relevant capabilities – too expensive.
    - Other methods – temporary access, etc, more in the research domain.
  - Must protect capability tickets against unauthorised alterations either by users or processes
    - as otherwise, subject could change rights encoded in a capability ticket or object to which they refer.
    - often done by using a cryptographically protected checksum using a key known to OS, or by having a trusted entity to hold/store the tickets.

DAC – Capability versus ACL

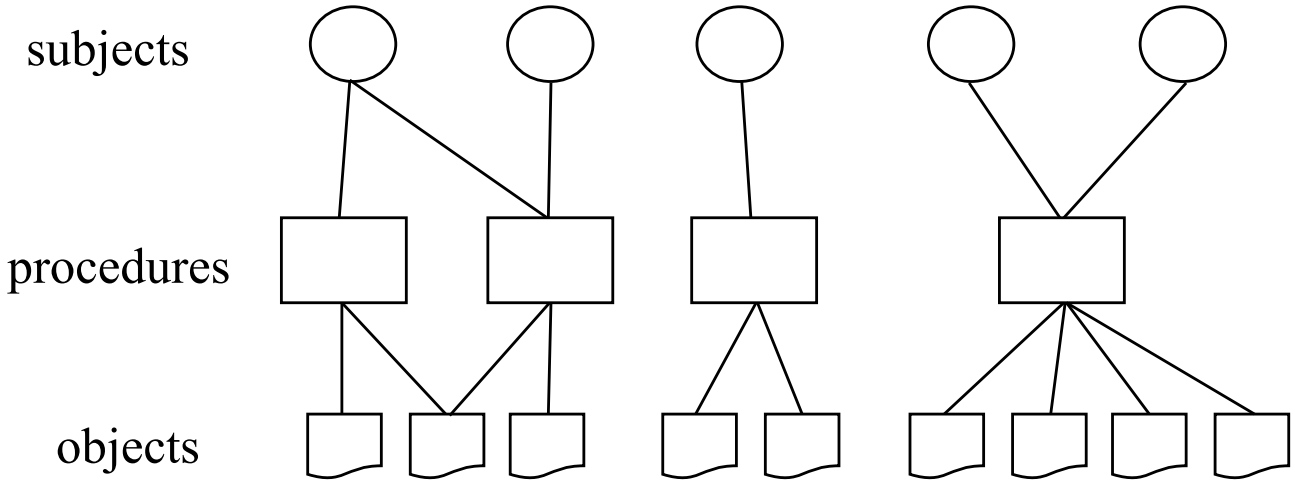
- Both theoretically similar:
  - Q1: Given a subject, what objects can it access, and what operation?
  - Q2: Given an object, what subjects can access it, and what operation?
  - C-List answers Q1, and ACLs answers Q2, better.

- In the past, more attentions have been on Q2 (largely because the use of non-distributed multi-user systems), which is the reason ACL-based systems are more common than C-based systems.

- Q1 is becoming more important in some applications, e.g. incident response and large-scale distributed systems, so there is a possibility 'Capability Lists may come back'.

DAC - Procedure-oriented

- Objects are encapsulated, permitting only certain specified accesses via program execution (e.g. you can only access a web server via a browser).



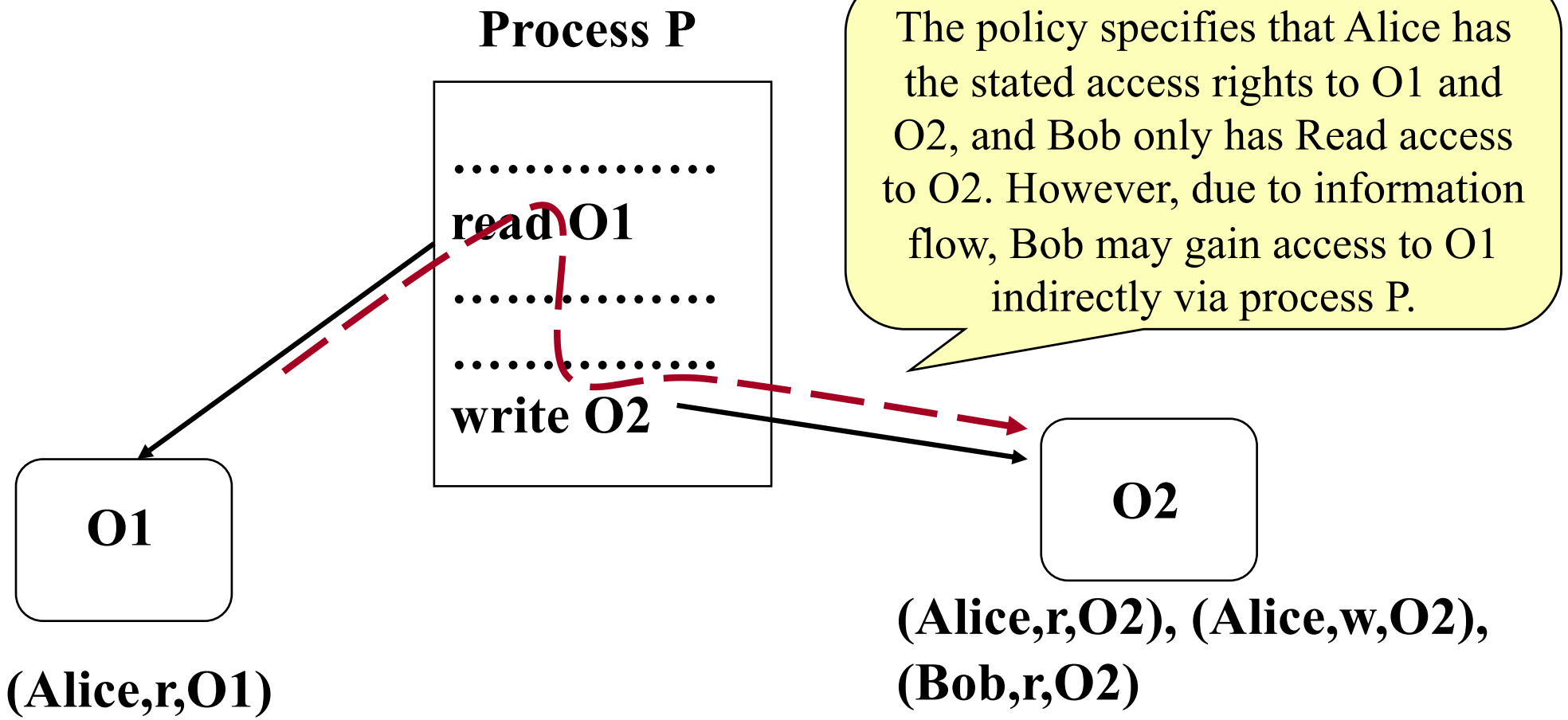
- Accesses to an object are done through a software procedure - a trusted interface.
- The penalty is performance cost.

Part 3 Overview

- Mandatory Access Control (MAC)

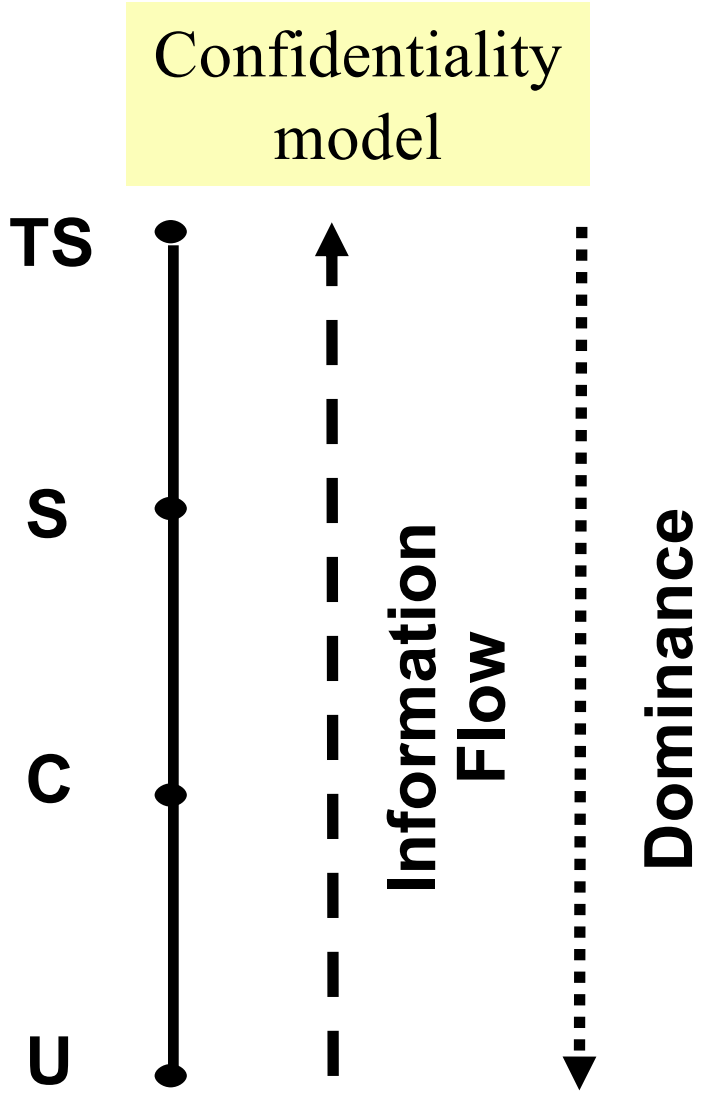
Mandatory Access Control (MAC) – Motivation

- DAC does not have information flow control -> Risk of Trojan Horse (covert channel).



MAC

- DAC cannot protect data against Trojan Horses embedded in programs, but MAC can, preventing unauthorised information leakage (ensuring confidentiality).
- Rights are determined by security labels:
  - For objects: **classification** (security/sensitivity).
  - For subjects: **clearance** (trust) levels.
- *C* is no more restrictive than *S*; *C* is dominated by *S*; information may flow from *C* to *S*.
- Also referred to as *multilevel security*.



{Top Secret (TS), Secret (S), Classified (C), Unclassified (U)}, where TS > S > C >U.

MAC

- A system-wide policy decrees who is allowed to access what.
- Used to control information flow in highly confidential environments such as military contexts and/or to prevent malicious software from modifying highly sensitive information such as access control policies.

- Subjects and objects are classified into different levels of trust and sensitivity:
  - Subjects are assigned clearance levels.
  - Objects are classified into categories each assigned with a classification level.
  - Only the subjects with a certain clearance level are granted with access to objects with a given security level.

MAC – BLP Model

- Mandatory control rules for **secrecy/confidentiality**
  - No read up (**read down**): object's classification must be below (or equal to) subject's clearance.
  - No write down (**write up**): object's classification must be above (or equal to) subject's clearance.
- The above rules can be formally expressed as: each subject s in S and each object o in O has a fixed security level $C(s)$ and $C(o)$ (denoting clearance and classification level), we have:
  - Subject s may have *read* access to an object o only if $C(s) \geq C(o)$.
  - Subject s can write to an object o only if $C(o) \geq C(s)$.
  - Subject s who has *read* access to an object o may have *write* access to an object p only if $C(p) \geq C(o)$.
- This is also referred to as **Bell-LaPadula (BLP) Confidentiality Model**

MAC – Biba Model

- Mandatory control rules for **integrity** (here security=integrity)
  - **Read up**: object's classification must be above (or equal to) subject's clearance.
  - **Write down**: object's classification must be below (or equal to) subject's clearance.
- The above rules could be formally expressed as: each subject s in S and each object o in O has a fixed integrity level $I(s)$ and $I(o)$ (denoting clearance and classification level), we have:
  - Subject s can read an object o only if $I(o) \geq I(s)$.
  - Subject s can modify (have *write* access to) object o only if $I(s) \geq I(o)$.
  - If subject s has *read* access to object o with integrity level $I(o)$, s can have *write* access to object p only if $I(o) \geq I(p)$.
- This is also referred to as **Biba Integrity Model.**

Part 4 Overview

- RBAC (Role Based Access Control)
  - Part4.1: Background
  - Part4.2: RBAC
- Conclusion

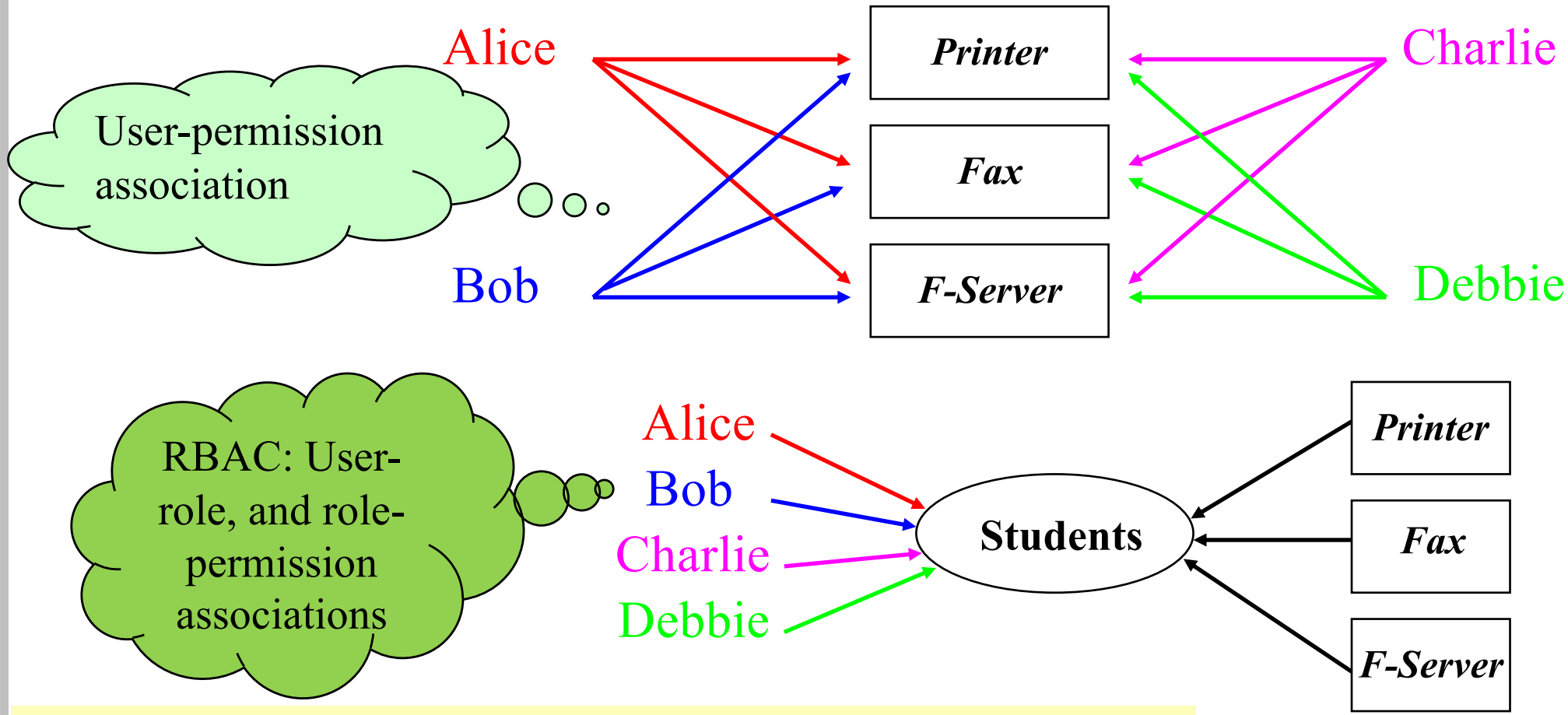
RBAC – Why do we need it

- Managing access control in a large organisation can be challenging
- When the numbers of subjects and objects are high, permission assignments can be complex and error-prone.
- If user population is highly dynamic, managing *grant* and *revoke* operations can be time-consuming.
- End users are often not the owners of resource objects being managed; the organisation is.
- Control is often based on employee functions rather than data ownership.

RBAC – Why do we need it

- RBAC provides a valuable level of abstraction - **roles**
- The idea is
  - Set permissions based on **roles**
  - Assign users to a role or a set of roles
- Because the roles within an organisation are relatively stable; they change less frequently than user turnover, RBAC can
  - simplify the task of access control management
  - reduces cost and potential errors in administrative process

RBAC - Why do we need it



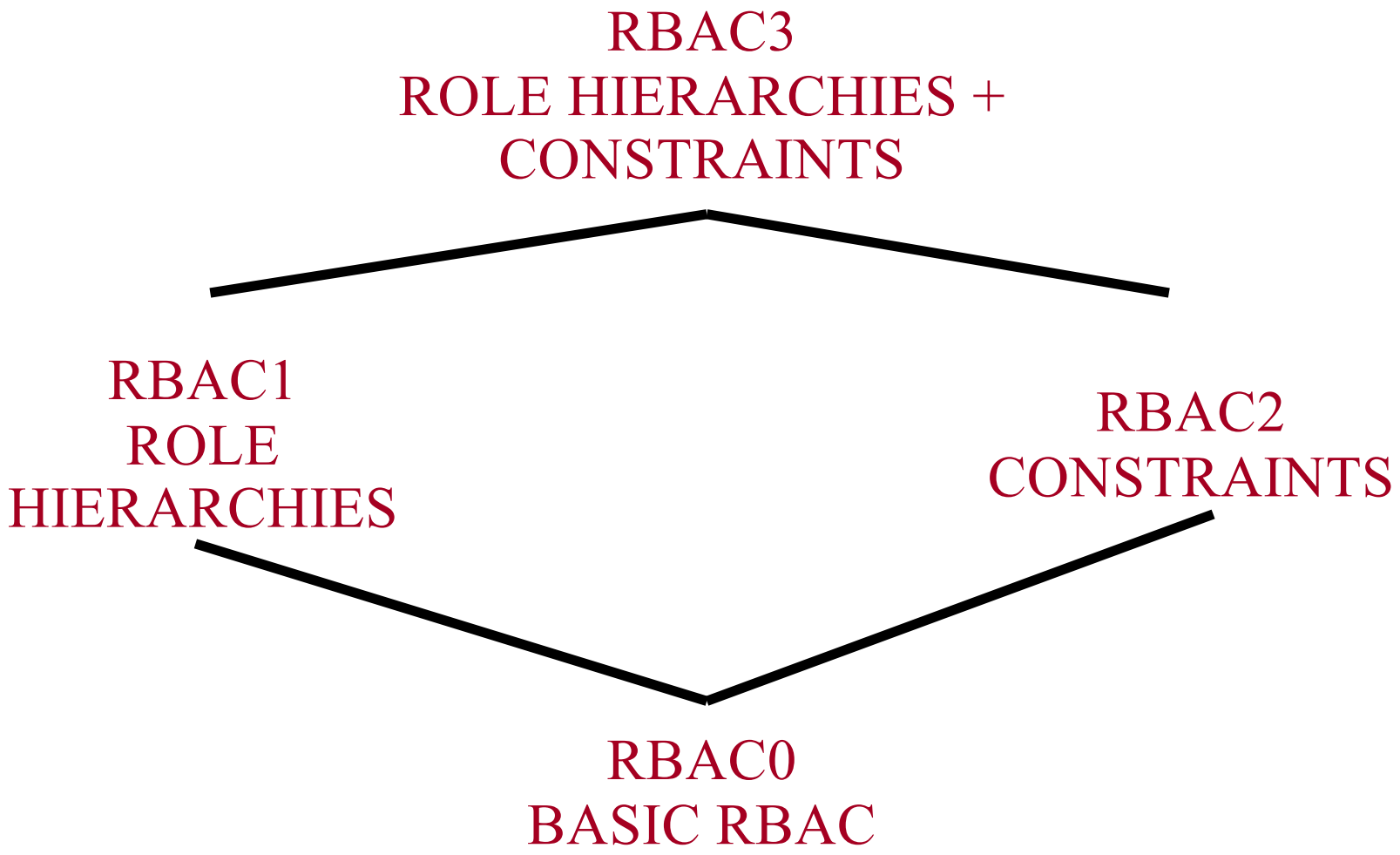
If (the number of roles) < (the number of users), the number of operations necessary to grant and revoke access privileges is greatly reduced.

RBAC – Security principles

- Least privilege
  - No more privilege than necessary to perform his/her job function.
- Separation of duties (conflict of interest constraints)
  - Static separation of duty: a user should not be authorized for both roles, e.g., student and staff.
  - Dynamic separation of duty: a user should not act simultaneously in both roles, e.g., casher and customer.

RBAC – Functional capabilities

- RBAC96 Family of Models



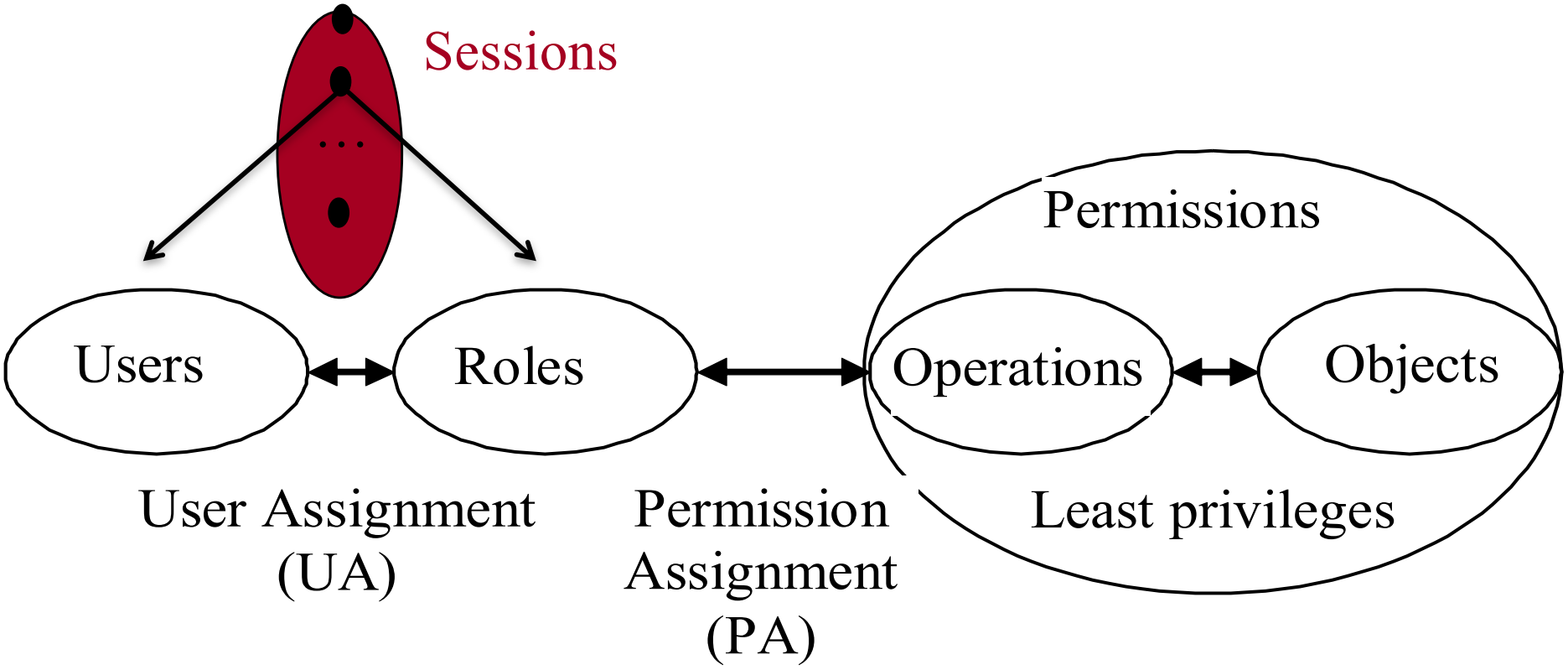
RBAC – RBAC0

- Users are assigned to roles, permissions are granted to roles, and users acquire permissions by being members of roles.
- A role is
  - a collection of users and
  - a collection of permissions

- Permissions are positive
- No negative permissions or denials
  - negative permissions and denials can be handled by constraints

RBAC – RBAC0

- A user can invoke multiple sessions; in each session a user can invoke any subset of roles that the user is a member of.

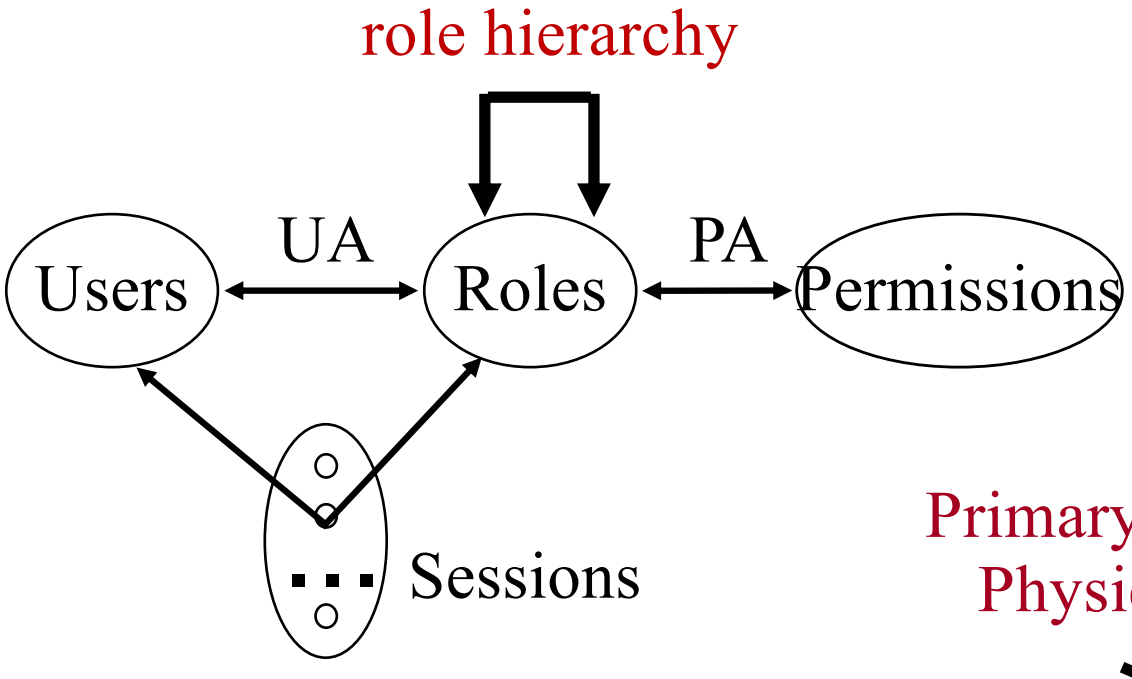


RBAC – RBAC0

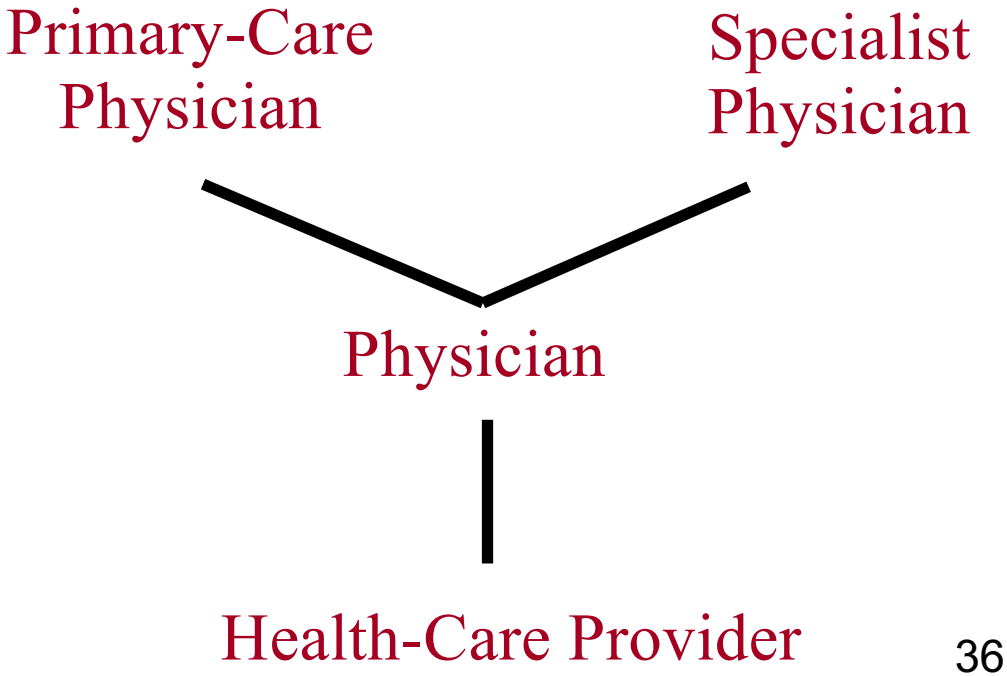
- Configuring RBAC involves the following tasks
  - Determining permissions.
  - Determining functional roles based upon tasks, responsibilities, and qualifications, etc.
  - Assign users to the roles (UA)
  - Assign permissions to roles (PA)

- Ability to support many-to-many UA and PA relations
  - A user can have many roles; a role can have many users.
  - A permission can be assigned to many roles; each role can have many permissions.

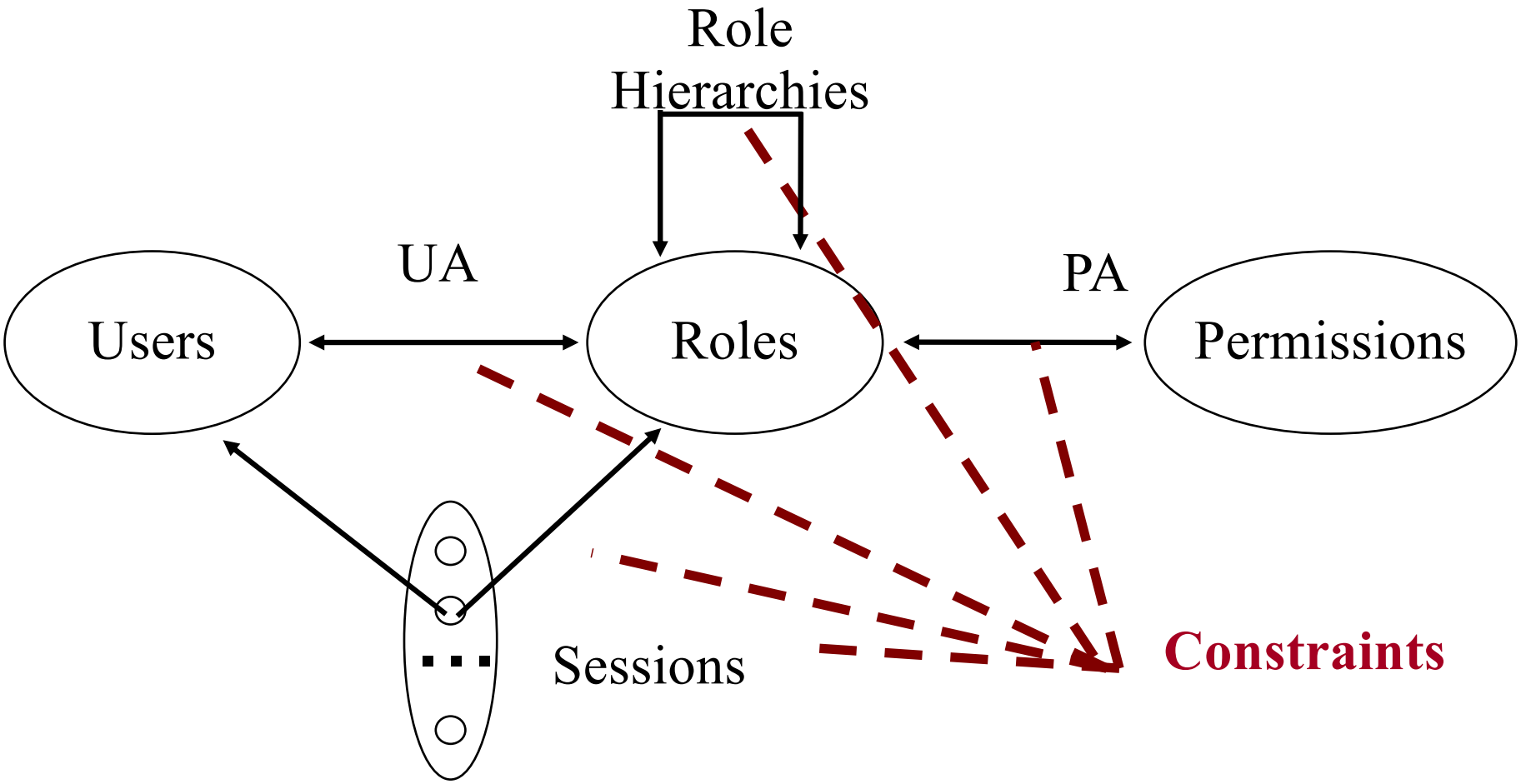
RBAC – Role Hierarchy



Hierarchical Roles



RBAC – Constraints



RBAC – Constraints

- Mutually exclusive roles
  - Static exclusion: the same individual can never hold both roles
  - Dynamic exclusion: the same individual can never hold both roles in the same context
- Mutually exclusive permissions
  - Static exclusion: the same role should never be assigned both permissions
  - Dynamic exclusion: the same role can never hold both permissions in the same context

RBAC – Constraints

- Cardinality constraints on User-role Assignment (UA)
  - At most k users can belong to the role
  - At least k users must belong to the role
  - Exactly k users must belong to the role

- Cardinality constraints on Permissions-role Assignment (PA)
  - At most k roles can get the permission
  - At least k roles must get the permission
  - Exactly k roles must get the permission

Exercise Question – E11.1

Contrast the following AC mechanisms, Directory, Access Control List, Capability and Procedure-oriented AC, **in terms of how easy it is** (for each case, you should consider what the system needs to do to accomplish the operation):

i) to determine authorised access during execution.
ii) to add access for a new subject.
iii) to delete access by a subject.
iv) to create a new object to which all subjects by default have access.
v) to revoke partial rights of a subject in the system.

Exercise Question – E11.2

- Rewrite an AC policy using MAC to mitigate the risk of trojan horse shown in the MAC-Motivation slide.

Conclusions

- Several mechanisms can be used to enforce access control, organised on a per-object or per-subject basis - they may be suited for different scenarios or application contexts.
- DAC models are flexible in terms of policy specification, and are typically used by OS and DBMS in commercial world. However, they do not provide information flow control, thus vulnerable to trojan horse attacks.
- MAC can protect against Trojan Horse attacks, and is typically used in military applications, and/or for constructing trusted computing systems.
- RBAC resembles the management structures of large organisations; it defines a user's permissions based on the user's roles rather than his identity (DAC) or clearance (MAC).