Security in Computing & Information Technology

Lecture 5

Access Control

Lecture Schedule

Foundations

1. Introduction
2. Vulnerabilities, Threats, Attacks

Basic mechanisms

3. Security mechanisms, Elementary cryptography
4. Authentication
5. Access control

Major computing security areas

6. Operating systems
7. Databases
8. Networks
9. Web
10. Mobile computing

Applications

11. Privacy
12. Internet banking

Lecture Topics

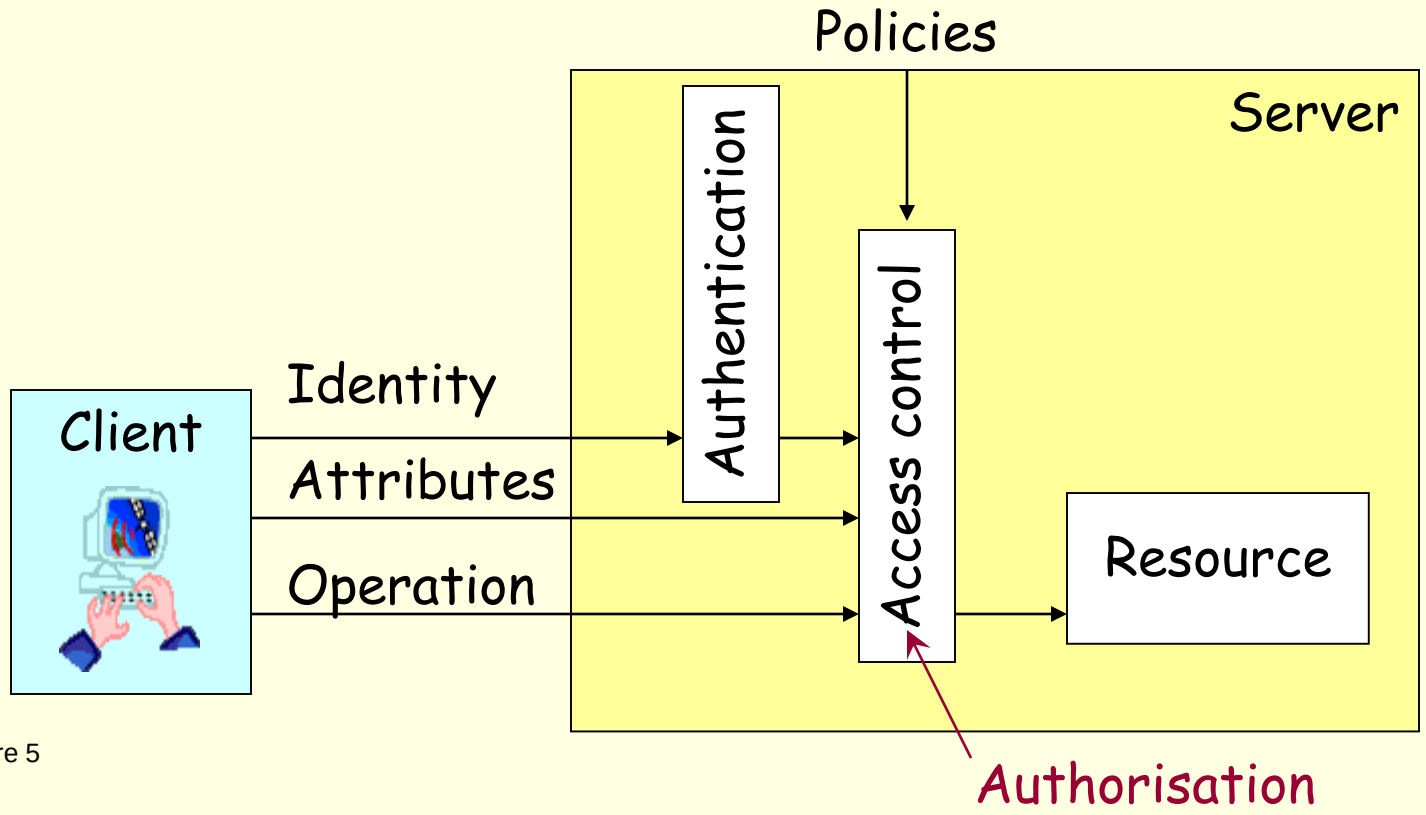
- Authorisation and access control
- Access matrix
- Access control methods

Authorisation

- Determines whether a user has permission to access (read, write …) certain resources
- Usage constraint
  - Can solve delegation
    - Alice authorises Bob to access her data (instead of e.g. disclosing her password to access the data)
- Based on
  - authentication and
  - privileges assigned to users
- Intended to refer to policy-definition
- Often used to describe policy-enforcement

Access Control

- A set of policies and mechanisms that permits authorised subjects to access restricted resources



Policies and Rules

- Access control policies
  - Conditions under which access is granted
  - Defined in fairly broad terms
  - Mostly application specific
- Access control rules
  - Wishes of the stakeholders of a resource
  - Define specific details
  - Formal rule specification is very difficult

Attributes for Access Control

- User attributes
  - Identity
    E.g. login username
  - Ticket, certificate testifying access rights
- Resource attributes
  - Name
  - Address
  - Operations that can be performed on them (read, write etc)
  - Access requirements / use conditions: restrictions on environmental conditions (time, weather …), user identities …

Access Control Participants

- Subjects
  - Active entities that perform operations, e.g. users
- Objects
  - Passive entities on which operations are performed, e.g. data
- Operation types
  - Observer
    - Does not modify the object (e.g. read)
  - Transformer
    - Alter content (e.g. write)
    - Alter existence (e.g. delete)

Access Matrix Model

- Protection states represented by a matrix
- Access rights: Kinds of accesses that may be performed on objects
  - Usual ones: Read, Write, Execute, Delete
- States:

Objects (o) / Subjects (s)	O1	O2	O3
S1	RWE		RW
S2	RE	RW	RE

- Decision rules
  - Data dependent / independent (e.g. who can access results)
  - Time dependent (e.g. results not available before announcement)
  - Context dependent (e.g. user cannot see names and results together)
  - History dependent (depends on previously accessed data)

Access Control Structures

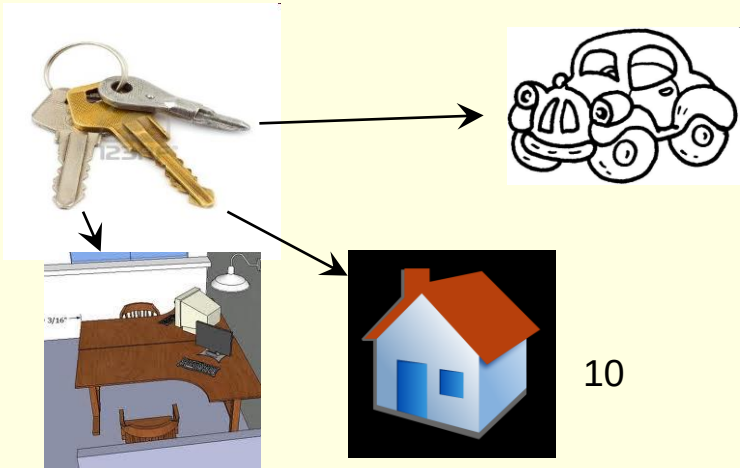
- Access control lists (ACLs)
  - A list of access rights attached to an object
  - Lists the users and their respective rights
  - Common in file systems (Windows, Unix)

ACL
UserID 1 Allow *Read Write Delete*
UserID 2 Allow *Read Write*
UserID 3 Deny *Read Write*

- Capability based security
  - Capability: a token allowing a subject (user, process) to access/use a resource

    E.g. Bunch of (encryption) keys,

    File descriptor [fd = open (file)]



ACLs vs Capabilities

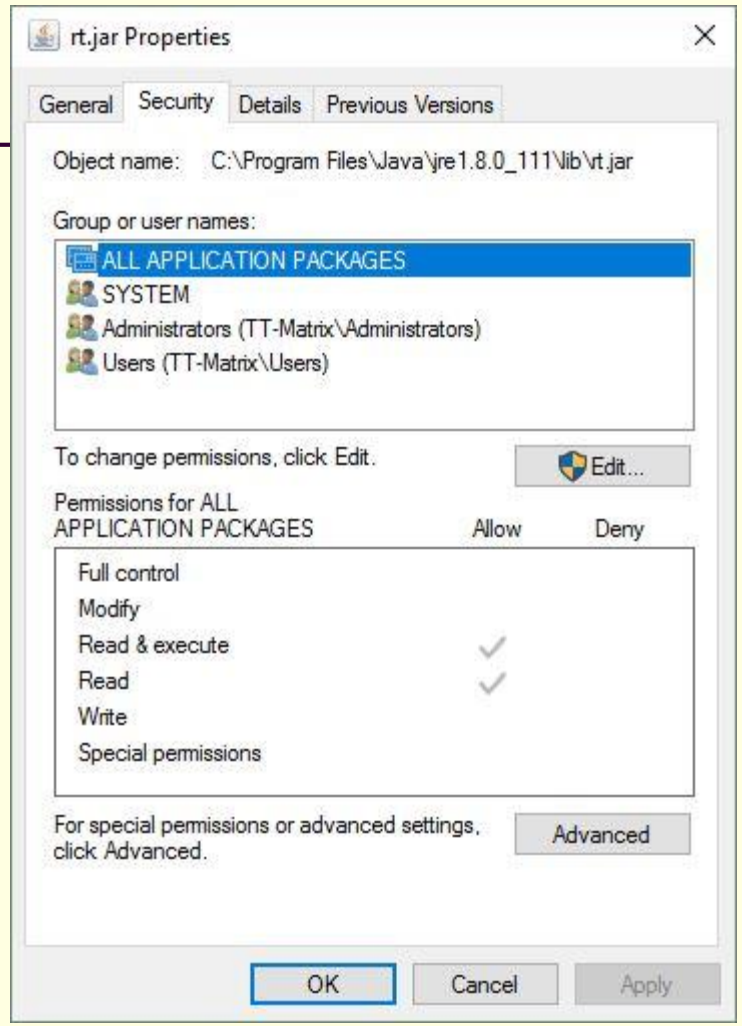
- Delegation of rights
  - ACL: needs interaction with administrator (owner), may be difficult during execution
  - Capabilities: can be passed from subject to subject
- Revocation
  - ACL: remove subject from the list
  - Capabilities: needs interaction with capability holder (needs proper administration to find the holder)

Access Rights (aka Privileges)

- Most common access rights
  - Read, write, append
  - Delete
  - Execute
- Access rights can be assigned to a group of users
  - Unix: user defined groups (listed in the `/etc/group` file)
  - Windows groups :
    - Built-in types: administrator, power user, …
    - User defined
- Positive & negative rights
  - Positive (what a user can do): the usual way
  - Negative rights (what the user is not allowed to do): used for overriding an other assignment (e.g. a right inherited from group membership)
- Object ownership
  - Subjects can own objects

File Example

- File permissions
  - Describe access rights to a file
  - Windows
    - Basic rights: Modify, Read, Write, Execute
    - Can define new rights
  - Unix
    - ACLs compressed into mode bits
    - basic rights: read, write, execute



Web Access Control (Apache)

- Web server uses ACL to control access to its web pages
  - By host

    Can be by (full or partial) domain name, IP address, network (with IP mask), e.g.

    ```
    Allow from apache.org
    Deny from 131.170.*.*
    ```
  - By environment variable

    E.g. user agent (that refers to browser, platform etc)

    ```
    SetEnvIf User-Agent BadBot GoAway=1
    Order allow,deny
    Allow from all
    Deny from env=GoAway
    ```
  - By arbitrary criteria

    E.g. time

    ```
    RewriteEngine On
    RewriteCond %{TIME_HOUR} >20 [OR]
    RewriteCond %{TIME_HOUR} <07
    RewriteRule ^/fridge - [F]
    ```

Network Access Control

Access to a network is controlled on device and user level

- Network admission (on-entry) control
  - Admission of
    - device to connect a computer to the network
    - user to access network resources (printers etc)
  - Identification of device, authentication of hosts or subject asking for admission
  - Pre-admission checks
    - Compliance with security policies
      E.g. Are anti-virus signatures up to date on the device?
  - Post-admission control
- Resource access control
  - Types of access to network resources

Mandatory Access Control

- The operating system prescribes and enforces users' access rights to resources
  (files, communication ports …)
- Features
  - Easy to manage
  - Suits scenarios with
    - central administration and control
    - hierarchical structure
- Was considered to be too restrictive, but is now gaining popularity
  E.g. assigning security levels and related rights to processes (Windows Mandatory Integrity Control)

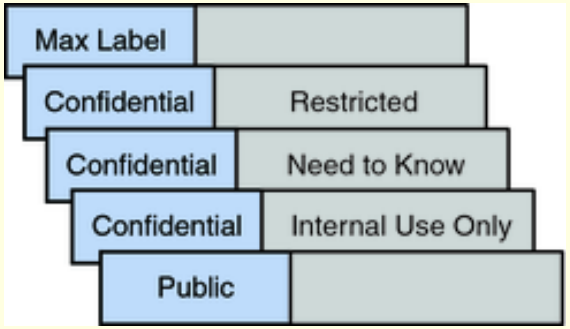


Image source: http://download.oracle.com/docs/cd/E18752_01/html/819-0868/ugintro-14.html

Discretionary Access Control

- Certain users can pass on certain rights to other users
- Features
  - More flexible
  - Difficult to enforce global rules
  - Typical scenario

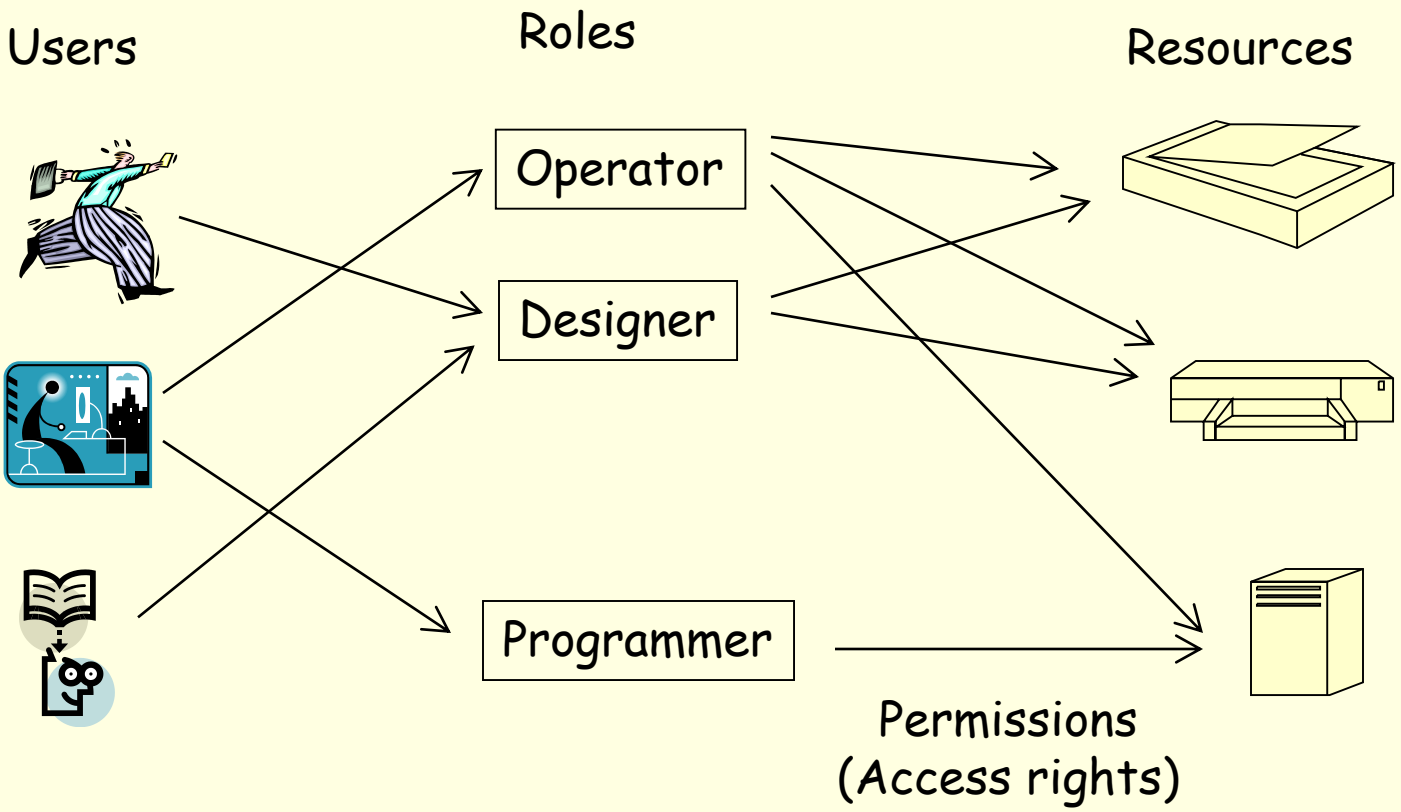
    Owners of objects can assign access rights to other users
- Most commercial systems support it to a variable degree

  E.g. Unix (**chmod**), Windows (File Properties → Security)

Role-Based Access Control (RBAC)

- Motivation
  - Large systems with large numbers of users
  - Many users have similar access rights
  - Operations can be assigned to certain roles (job functions)
  - Organisational policies have to be uniformly handled
- Requirements
  - Flexibility: Users and their access rights may change

RBAC Model


Users
Roles
Resources
Operator
Designer
Programmer
Permissions
(Access rights)

RBAC Components

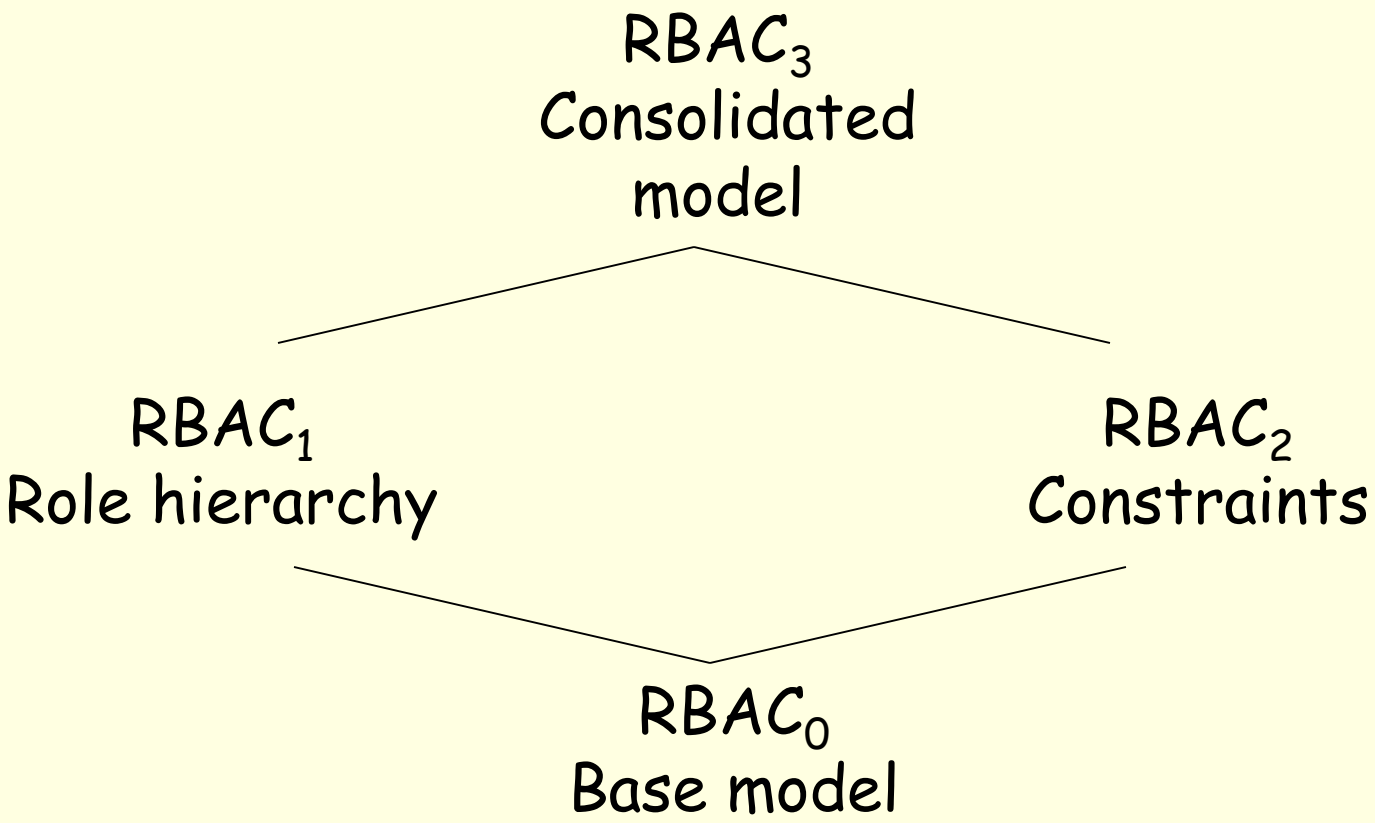
- Users
  - Collection of people, processes etc who use the system
  - Have possibly different sets of access rights
- Roles
  - Typical functions performed by users
  - Mediators between users and access rights
- Permissions (access rights)
  - Approval of a mode of access to a resource
- Role assignment
  - Set of roles the user may take on
- Role activation
  - Role the user is currently acting in

RBAC and Security Policies

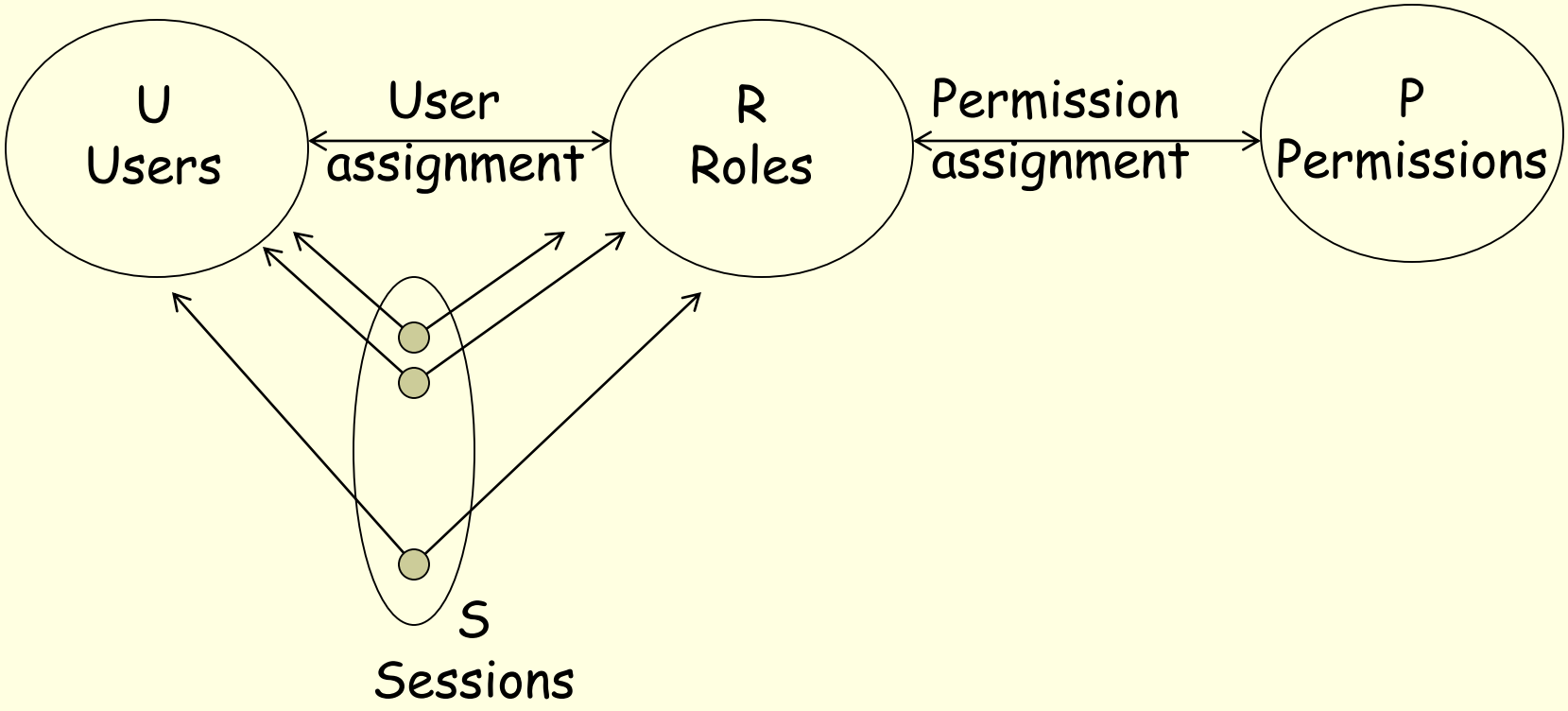
- Expresses organisational policies

  E.g.
  - The same person cannot have certain roles simultaneously
  - The number of users in a role is limited
  - Least privilege: a user must have the minimum set of access rights needed to perform the task
- Policy neutrality
  - RBAC provides a tool to express requirements
  - Configuration of RBAC implements the policies
- Independent of other access control methods (MAC, DAC)
  - But can coexist with them

The RBAC Conceptual Model



$RBAC_0$ Reference Model



$RBAC_0$

- Permissions
  - Positive permissions: ability to perform an action
  - Can apply to a single object or to many
  - Can be specific (read a file) or general (read all files of this department)
- User-to-role assignments
  - Many to many
    - A user can have a number of roles
    - A number of users can have the same role
- Role-to-permissions assignments
  - Many to many
    - A role can have a number of permissions
    - A number of roles can have the same permission

$RBAC_0$

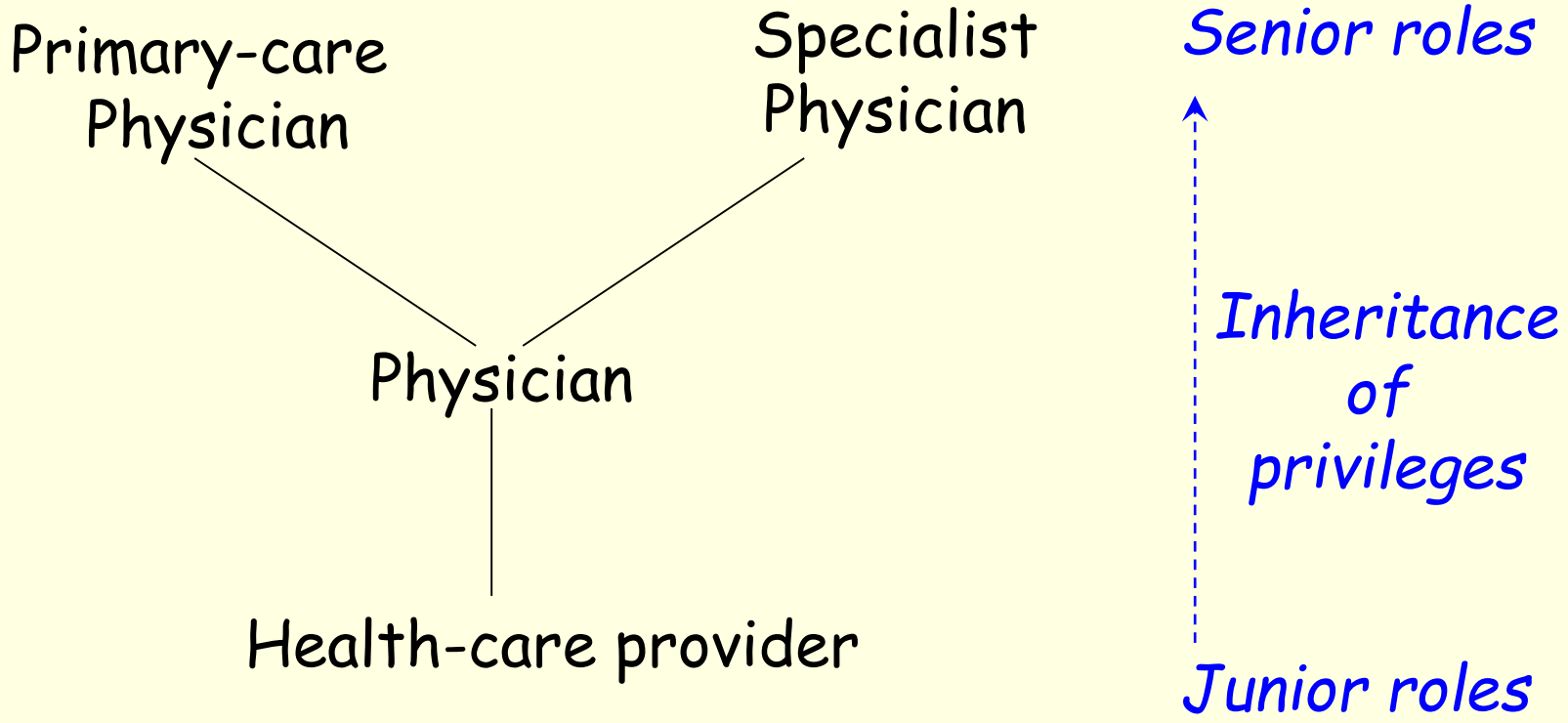
- Session
  - A mapping of a single user to one or more roles; activating a subset of all roles permitted for the user
  - Permissions: union of all permissions from all roles of the user
  - A session is associated with a single user
  - A user can have multiple active sessions simultaneously
  - A session is under the control of the user (e.g. the user can terminate it)

$RBAC_1$ – Role Hierarchies

- Structuring roles
  - Reflect the organisation's lines of authority and responsibility
  - More powerful (senior) roles can inherit permissions from less powerful (junior) roles
- Mathematically: A role hierarchy is a partial order
  - Reflexive: a role inherits its own permissions
  - Transitive: if *A* inherits a permission from *B* and *B* inherits that permission from *C*, then *A* also inherits it from *C*
  - Antisymmetric: roles cannot inherit from one another (roles would be redundant)

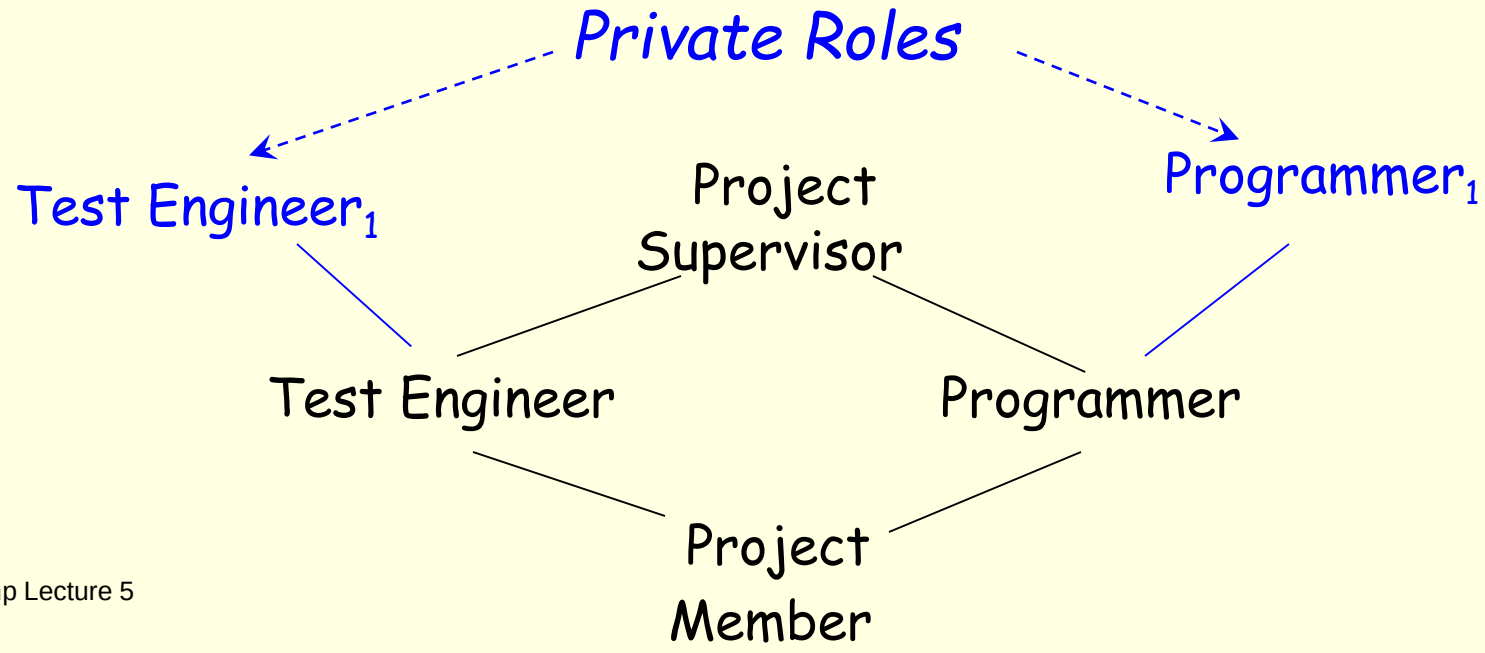
$RBAC_1$ – Role Inheritance

A Role Hierarchy



$RBAC_1$ – Limited Inheritance

- Sometimes it is useful to limit inheritance
  - E.g. access to incomplete work should be granted to developers only
  - Non-inheritable permissions can be assigned to private roles
  - Private roles can also form a hierarchy
    - Cross inheritance between private roles can make the hierarchy very complex



$RBAC_2$

$RBAC_2 = RBAC_0 + Constraints$

- Restrictions on roles and users
  - They define acceptable and non-acceptable permissions
- Enforces organisational policies
- Can apply to
  - Sessions
  - User and role functions

$RBAC_2$ – Constraints

- Separation of duty (mutually exclusive roles)
  - A user can assume a role only if it is not in conflict with other roles of the user
    E.g. account manager - purchasing manager, programmer - tester
  - Can be static (role assignment) or dynamic (role activation) separation
- Cardinality
  - Restriction on the number of users in a role
    - Maximum number: e.g. to enforce licence agreements
    - Minimum number: difficult to enforce (procedures need to be activated when a user leaves the system)
- Prerequisite roles
  - A user can be assigned to a role if the user is already assigned to another role
    E.g. Programmer must be a Project Member

$RBAC_3$ – The Consolidated Model

- Integrates $RBAC_1$ and $RBAC_2$ features into $RBAC_0$
- Constraints on role hierarchies
  - Limit the number of senior/junior roles of any given role
  - Certain roles may not have common senior/junior roles
- Private roles
  - Can be mutually exclusive (e.g. programmer and tester)
- Interactions
  - Constraints apply to direct membership, or carry on to inherited membership

Summary

- Access permissions are expressed in different representations of the access control matrix
- The actual access control method depends on the environment
- In large systems, role-based access control is the most frequently used method