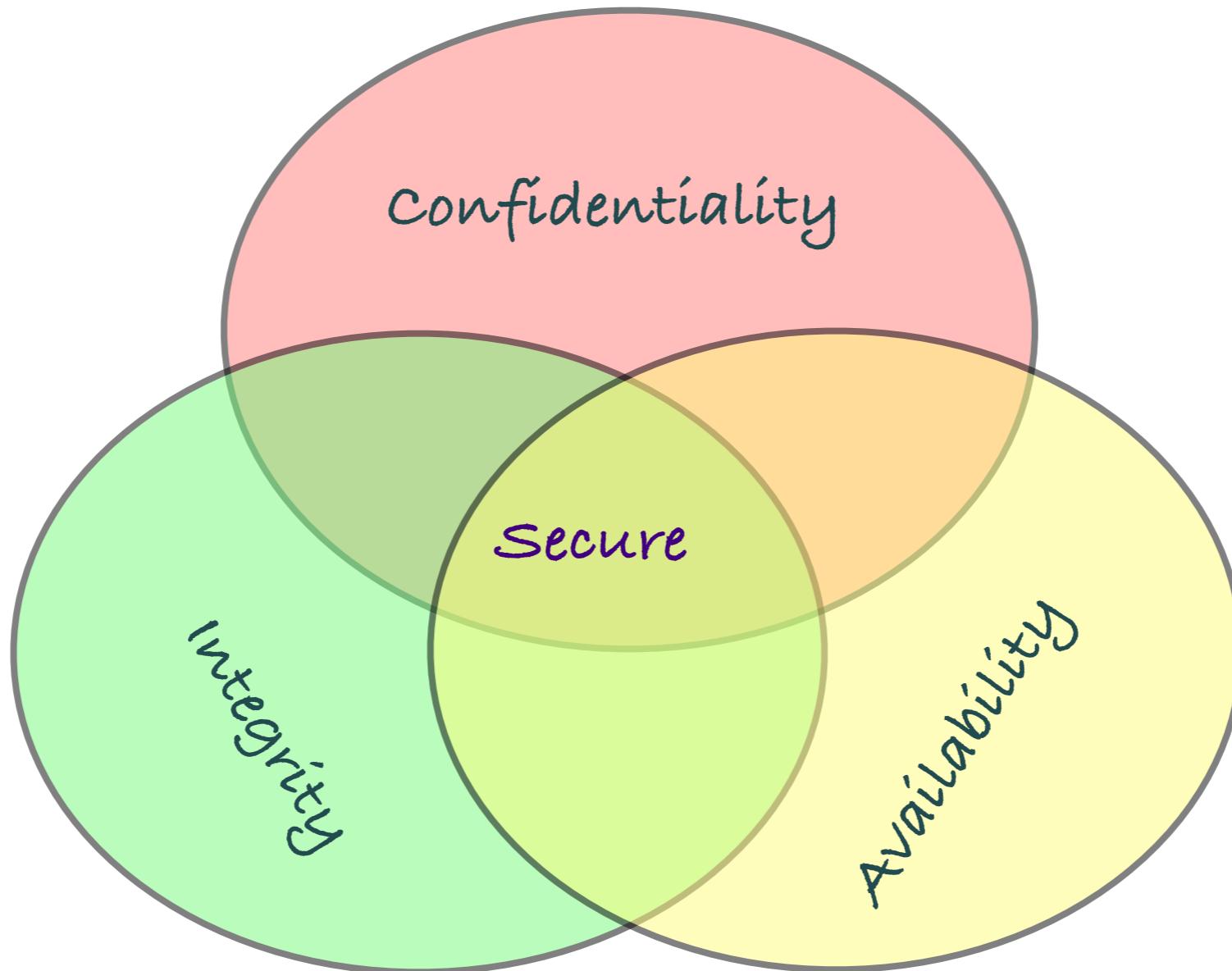


Security Policies

# Security Targets



**Security:** find the right balance among these objectives

# Policies vs Models

più sono formali,  
meno sono ambigue  
più definiscono quali che  
vegiono

rispetta agli aspetti indicati

\* A **security policy** defines what is allowed

- ✓ It characterizes the acceptable system's runs, or viceversa, those that are not acceptable
  - Analogous to a set of laws
  - It is defined in terms of high level rules and requirements that should be well defined, coherent and implementable

\* A **security model** provides a formal representation of a set of IT systems, underlying the needed security features

- ✓ It is an abstract description of system's behaviors and it is the formal base for designing policies

# Policies vs Models

- \* A system model describes what a user can do, not necessarily what it is authorized to do.

**Soundness:** The system satisfies the security properties (Are all reachable states authorized?)

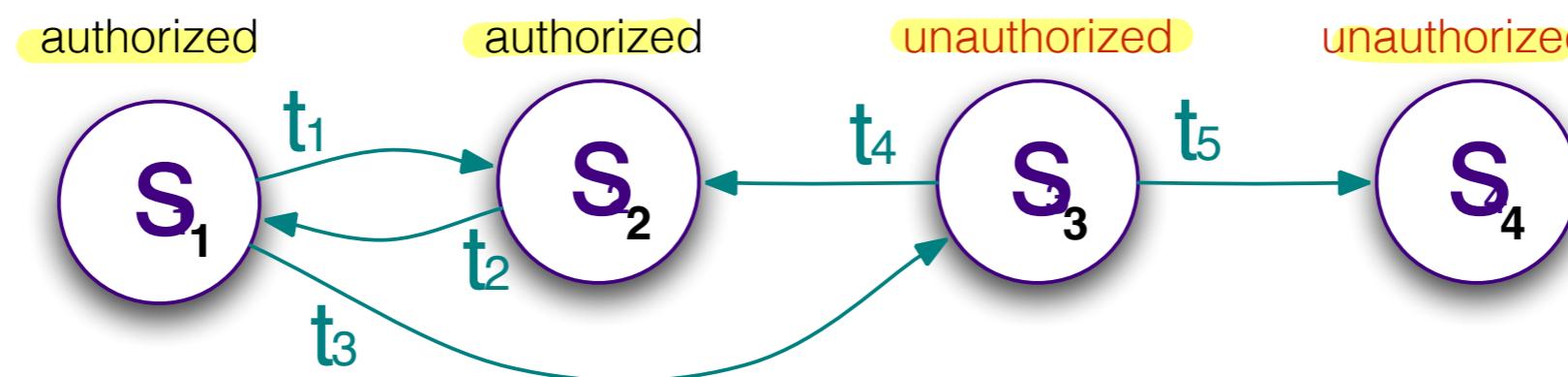
- \* When a model is sufficiently formal, soundness is well defined:

System  $\models$  Security property

# Secure System

\* **Secure system**: It is a system that starting from an authorized state can only enter in an authorized state

Il meccanismo deve impedire le transizioni non sicure



Automa che rappresenta gli stati in cui un sistema si può trovar con le transizioni

\* **Security breach**: occurs when a system enters an unauthorized state.

# Secure System

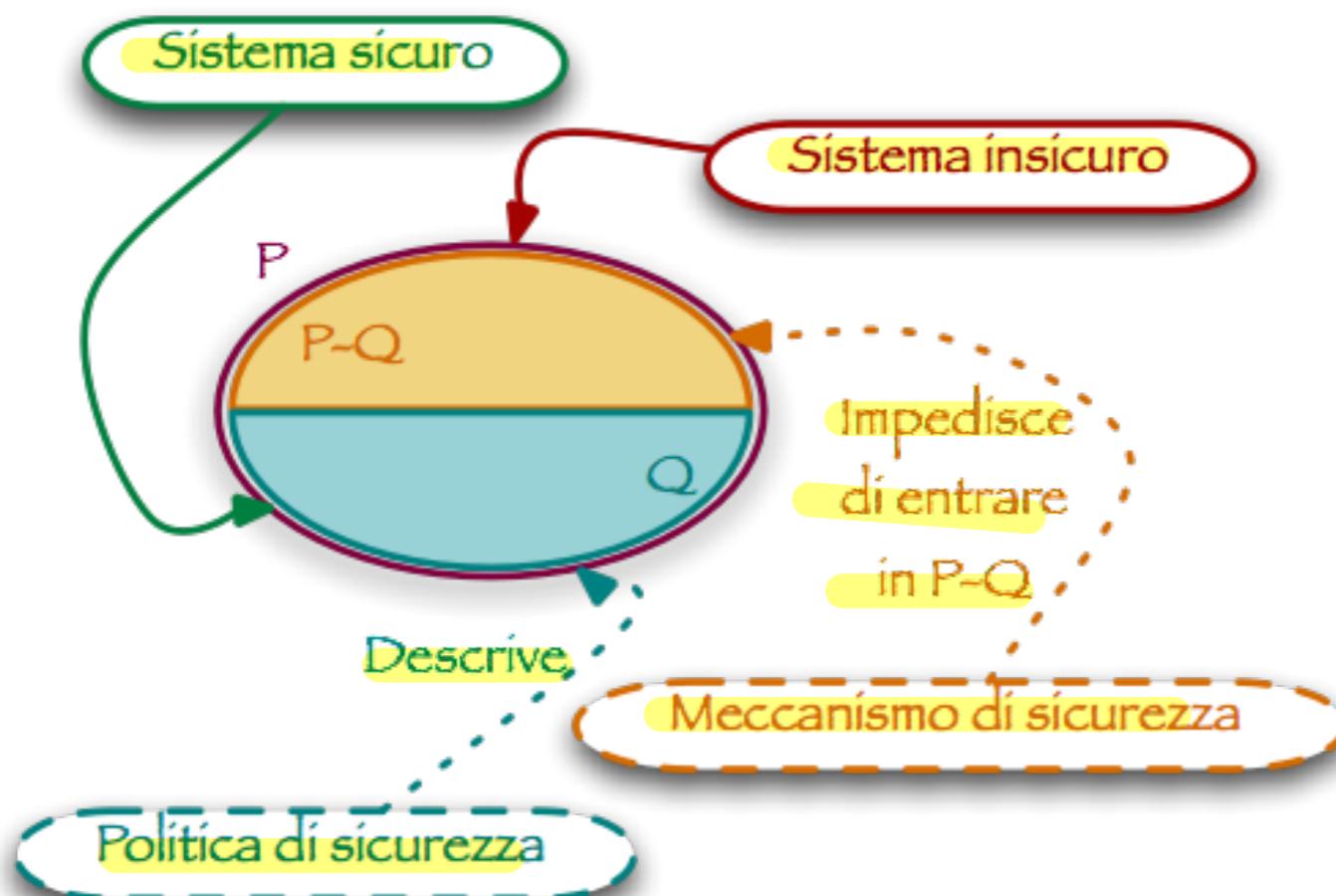
È un'astrazione e  
un'istruzione del sistema



- \* Policies and models are formalized in terms of **states** of the system
- \* A **state of a system** collects the current values stored in memory locations, registers and it provides a snapshot of the system in a certain instant of time
- \* The subset of information contained in a system state that regards protection is called **protection state**:
  - ✓ **File system**: portion of the system state that contains information on who is writing/reading a file, information on access control, etc...
  - ✓ **Program**: portion of the state system that provides a run-time image of program execution and contains information such as the value of the program counter, of the call stack, etc...
  - ✓ ...
- \* **Abstraction**: A computer system is represented as a transition system!

# Secure System

- \* Let  $P$  be the state space of the system, and let  $Q \subseteq P$  be the set of authorized system states
  - ✓ When the system is in a state in  $Q$  then the system is secure
  - ✓ When the system is in a state in  $P \setminus Q$  the system is not secure



- \* A security policy partitions the system states  $P$  in secure and insecure states
- \* A system is secure when starting from a secure state cannot enter an insecure state

# Secure System

- \* A policy is defined as a set of authorized/secure states
- \* Sometimes being authorized/secure may depend on the history of computation and therefore this cannot be decided by looking only at the current state
  - ✓ Example: a policy that does not allow to add an object that has been previously deleted
- \* To handle this kind of security requirements it is necessary to use a policy defined on system traces, thus a policy that defines the set of authorized/secure traces:

$S_0 S_1 S_2 S_3 \dots$

- \* A system is secure when every trace is authorized/secure

# Types of Security Policies

- \* It is important to identify the adequate security policy for a system
- \* **Military security policy** (or **governmental** security policy) is a security policy developed primarily to provide **confidentiality** where information is classified as **non-classified, limited, classified, secret, top-secret**.
- \* **Commercial security policy** is a security policy developed primary to provide **integrity**
- \* **Confidentiality policy** (**information-flow policy**) is a security policy dealing only with **confidentiality**
- \* **Integrity policy** is a security policy dealing only with **integrity**

# Examples of Policies

## Security policy for e-banking

A client of the bank can see its balance and the transactions he/she has made. He/she can transfer money from his/her account to another account for amounts less or equal to 10.000 Euro. For bigger amounts the approval of a manager is needed....

## Security policy for privacy

A user can access personal data only if this access is needed for accomplishing his/her current task and only if the user is allowed to execute such a task. Moreover, the goal of the current task has to correspond to the goals for which the personal data were originally collected, otherwise the user has to obtain an explicit permission.

It combines conditions with constraints on how data has to be used

Access Control

# What is access control?

- \* Central element of cyber security
- \* The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner
- \* Involves users and groups
  - ✓ authenticate to system
  - ✓ assigned access rights to certain resources on system

# Access control in use



# Access control in use

	Read	Write	Execute
Owner	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Group	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Others	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Octal: 755		Text: rwxr-xr-x	



# Access control in use



File1: Permissions

	Read	Write	Execute
Owner	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Group	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Others	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Octal: 755      Text: rwxr-xr-

Sharing settings

Link to share  
<https://drive.google.com/drive/folders/1lehTwOFR5c0-WaiMIOxQzseg5M1Z9aqW?usp=sharing>

Share link via:    

Who has access

 Anyone who has the link can view Change...

 Bojan Siljanovski (you) Is owner  
bojan.s@gmail.com

 help@coffeewithit.com X

Invite people:  
Enter names or email addresses...  

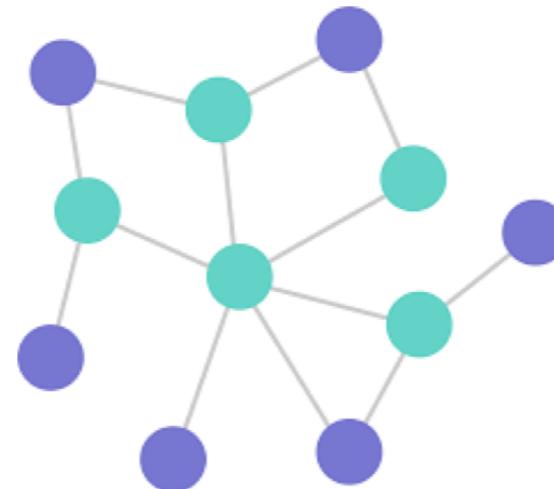
Owner settings [Learn more](#)  
 Prevent editors from changing access and adding new people

# Access Control system



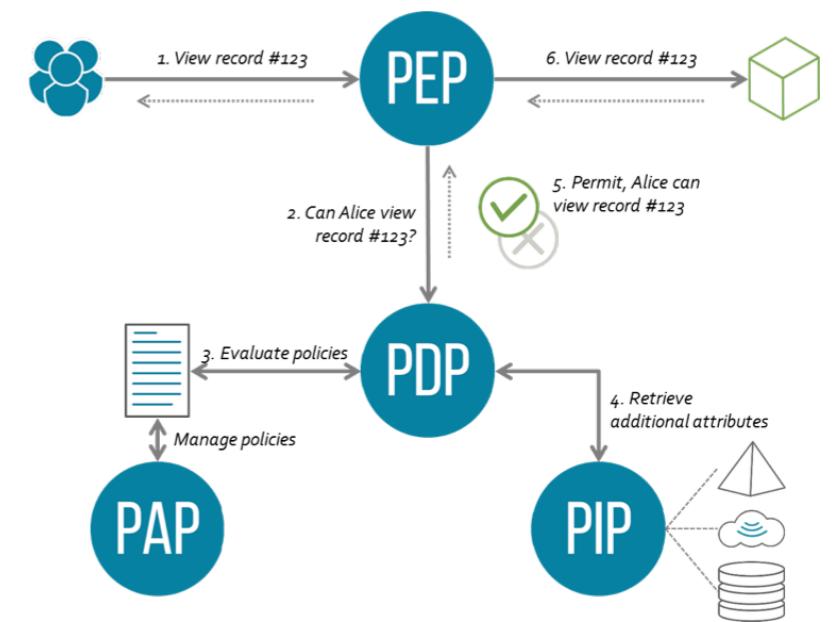
Access Control Policies

↓  
Policy



Access Control Model

↓  
modelli su  
ci definisco  
(a policy)

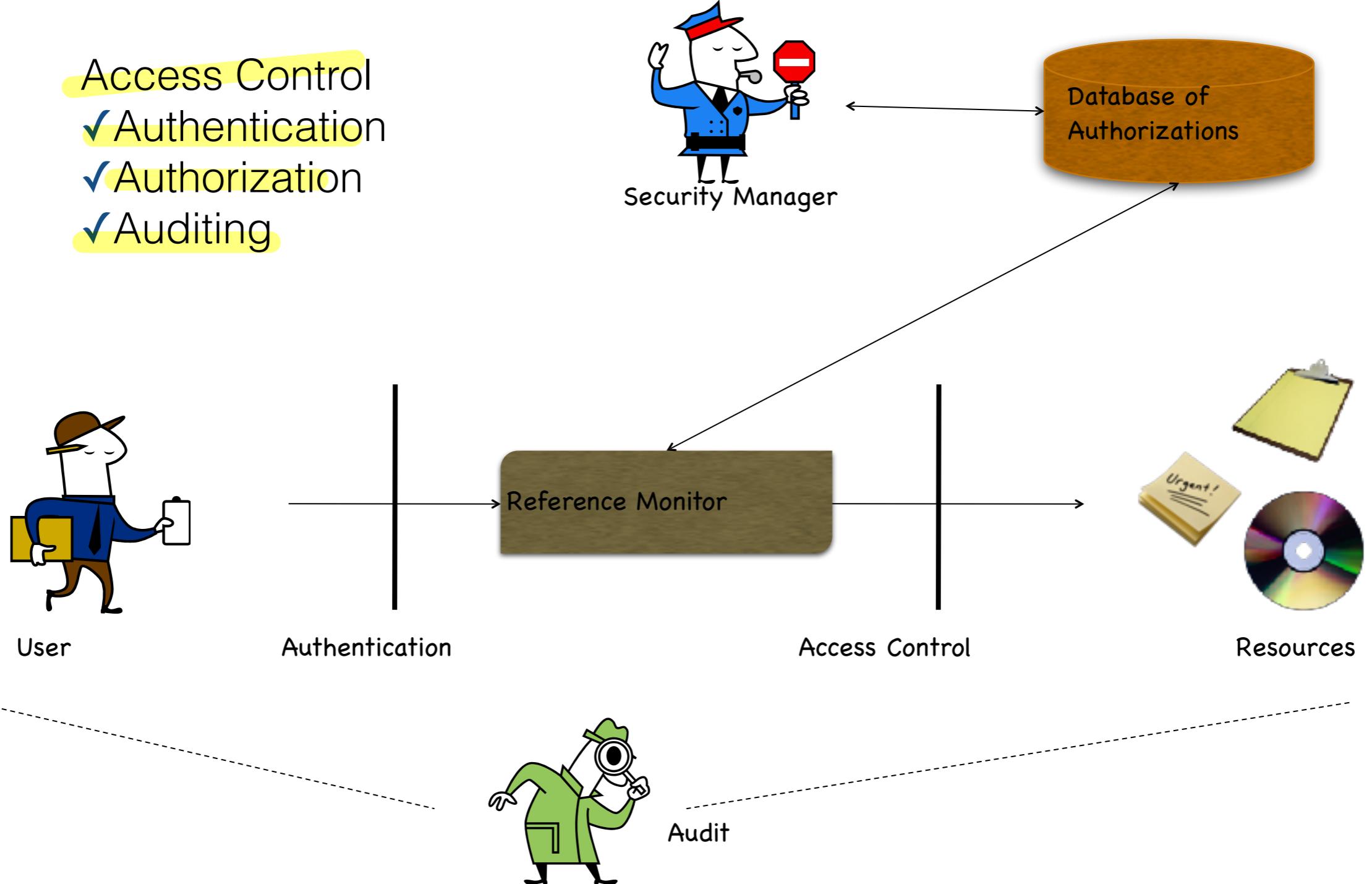


Access Control Mechanism

↓  
meccanismo che  
implementa le policy

# Access Control

- Access Control
- ✓ Authentication
- ✓ Authorization
- ✓ Auditing



# Access control principles

## \* Authentication

- ✓ The verification of an identity claimed by or for a system entity

## \* Authorization

- ✓ The granting of a right or permission to a system entity to access a system resource

## \* Audit

- ✓ The monitoring and processing of user accesses to system resources

# Auditing

## \* Two main components:

- ✓ *Collection* and organization of data for verification
- ✓ *Analysis* of the collected data to discover security violations

## \* Intrusion detection:

- ✓ *Passive*: offline analysis to detect intrusions already happened
- ✓ *Active*: real-time analysis for an immediate response for protection

## \* Problems

- ✓ Decide what has to be monitored
- ✓ How to automatically decide violations (offline or real-time)

## \* Solutions

- ✓ Anomaly detection: exploit of vulnerabilities typically corresponds to unusual system behaviors
- ✓ Detection of use of the system that is not allowed: rules that specify the allowed sequences of events of the system

# Access Control Requirements

- \* Reliable Input:
  - ✓ an access control system assumes that a user is *authentic*
- \* Support for fine and coarse specifications:
  - ✓ The access control system should support *fine-grained specifications*, allowing access to be regulated at the level of individual records in files.
  - Moreover, each individual access should be controlled rather than a sequence of access requests.
- \* Least privilege (*Need-to-know*): each system entity is granted the minimum system resources and authorizations that the entity needs to do its work
- Separation of duty: dividing the steps in a system function among different individuals
- \* Open and close policies: specify what is allowed (close) or not allowed (open)
- \* Policy combination and conflict resolution
- \* Administrative policies: Specifies who can add, delete, modify the authorizations rules

# Access control policy elements: subjects

**Subject** is an entity capable of accessing an object.  
Generally the concept of subject equates with that of  
process that actually accesses the object

- \* There are three different types of subjects
  - ✓ **Owner**: this may be the creator of a resource, i.e. a file
  - ✓ **Group**: a named group of users may be granted access rights. The membership to the group is enough to exercise access rights. A user may belong to multiple groups
  - ✓ **World**: the least amount of accesses and rights is granted to users that can access the system but that do not belong to the owners or to a group

# Access control policy elements: objects

**Object** is a resource to which access is controlled. It is an entity used to contain and/or receive information (records, pages, files, directories, mailboxes, messages, programs...)

- \* 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 trade-off between
  - \* security
  - \* complexity
  - \* ease of use

# Access control policy elements: access rights

**Access right** describes the way in which a subject may access an object

- \* Examples of access rights
  - ✓ **Read**: subject may view, copy and print information
  - ✓ **Write**: subject may add, modify or delete data, it includes read access
  - ✓ **Execute**: subject may execute certain programs
  - ✓ **Delete**: subject may delete certain resources
  - ✓ **Create**: subject may creare certain resources
  - ✓ **Concatenate**: (append or blind write)
  - ✓ **Search**: subject may list or search a resource

# Access Control Policies

obbligazione

- \* **Discretionary Access Control (DAC):** access control based on the identity of the requestor and on access rules
- \* **Mandatory Access Control (MAC):** access control based on comparison of security labels that indicate how sensitive the resources are with security clearance that indicate the resources that an entity can access
- \* **Role-based access control (RBAC):** access control based on the roles that users have within the system and on rules stating what accesses are allowed to users in a given role
- \* **Attribute-based Access Control (ABAC):** access is based on identity attributes of the subject, the object, and the context

Discretionary Access Control DAC

# Discretionary Access Control DAC

↳ Discrezionalità dell'utente

Users own resources and control the access to resources

- \* Access to data objects (files, directories, etc.) is permitted based on the *identity* of users.
- \* Owner controls the access rights of objects and of other users
- \* Discretionary: users can be given the ability of passing on their privileges to other users, where granting and revocation of privileges is regulated by an administrative policy
- \* Flexible mechanism prone to errors:
  - ✓ it is necessary that all users understand the mechanism and respect the security policy
  - ✓ there is no control on the flow of information
- \* often provided using an access matrix

# Access Control Matrix

- \* Provides the relations among **objects** (passive entities) and **subjects** (active entities) in terms of **access rights**. The two dimensions of the matrix represent:
  - \* Identified subjects that may attempt to access resources
  - \* Objects that may be accessed

matrix di controllo dell'accesso

↑  
subjects

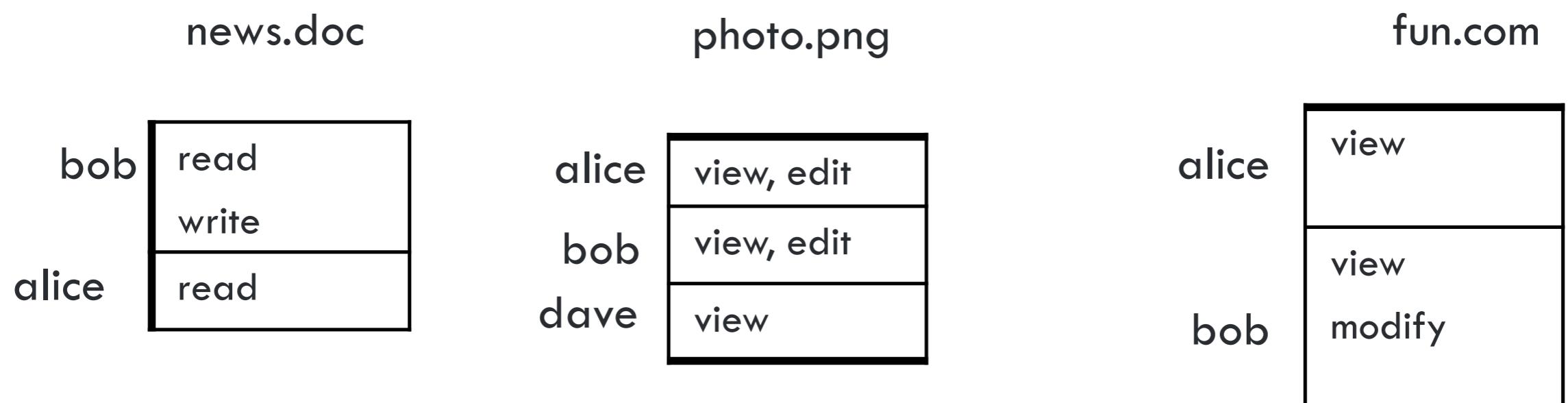
objects →

	news.doc	photo.png	fun.com
alice	read edit	view Cojo posso far	view
bob	read write	view edit	view modify
charlie			
dave		view	

non è scabbile

# Access Control List

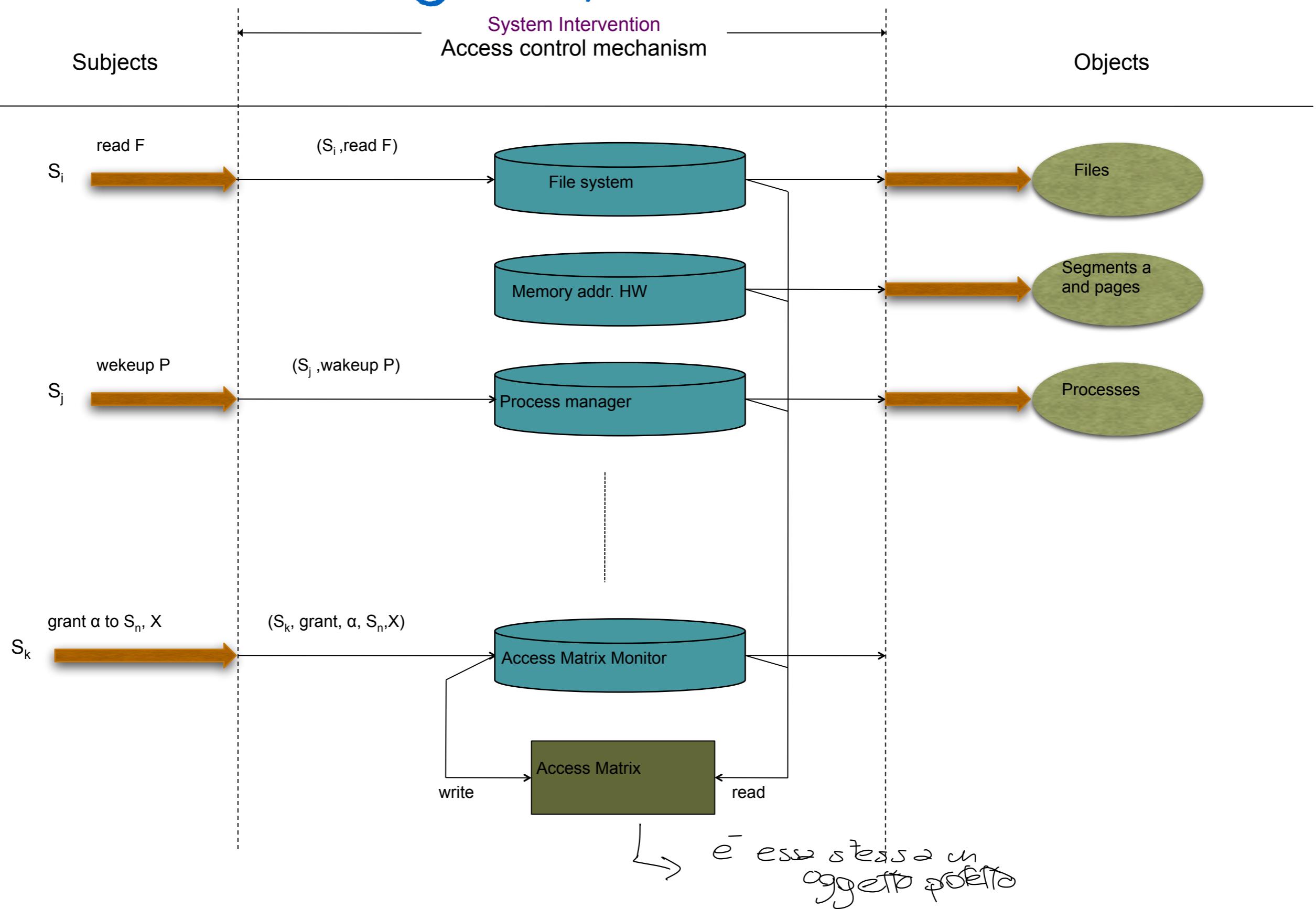
*Lista degli utenti che possono subire azioni su un oggetto*



# capability List

	news.doc	photo.png	tun.com
Alice's capability	read	view, edit	view
Bob's capability	read write	view, edit	view edit
Charlie's capability			
Dave's capability		view	

# verify the protection state



# Examples of DAC policies

- \* Graham-Denning Model (1972)
- \* Harrison-Ruzzo-Ullman Model (HRU)
- \* Take-Grant Model

# Graham-Denning Model (1972)

- \* DAC access control mechanism based on **protection rules**
- \* subjects S, objects O, rights R, matrix M
  - ↳ determinate privileges
  - ↳ riga per oggetto
  - ↳ colonna per oggetto
- \* Objects have **owners** with special privileges and subjects have **controllers** with **special privileges**
  - ↳ determinate privilege
- \* There are 8 protection rights:
  - ✓ create/delete objects
  - ✓ create/delete subjects
  - ✓ Read access rights
  - ✓ Grant access rights
  - ✓ Delete access rights
  - ✓ Transfer access rights

# Graham-Denning Model: RULES

regole che vanno a formare una politica

Discreta

↳ destra della linea

Rule	Command (from $s_0$ )	Condition	Operation
R1	create o	new owner	add column for o in M; insert 'owner' in $M[s_0, o]$
R2	delete o	'owner' in $M[s_0, o]$	deletes column for o in M
R3	create s	new controller	add a row for s in M; insert 'control' in $M[s_0, s]$
R4	delete s	'control' in $M[s_0, s]$	delete row for s in M
R5	read the access right of s on o può dare ad altri	'control' in $M[s_0, s]$ or 'owner' in $M[s_0, o]$	copy $M[s, o]$ in $s_0$ → new owner o controller
R6	grant r [r*] for s on o → può diritto	'owner' in $M[s_0, o]$	add r [r*] in $M[s, o]$ ↳ devo essere owner
R7	delete r from s on o	'control' in $M[s_0, s]$ or 'owner' in $M[s_0, o]$	remove r in $M[s, o]$ ↳ devo essere owner
R8	transfer r[r*] for s on o	'r*' in $M[s_0, o]$	add r[r*] in $M[s, o]$

# Harrison-Ruzzo-Ullman Model (HRU)

*S will access O?*

- \* Access control model based on commands, where every command involves conditions and primitive operations
- \* Commands determine the transition from one state to another one when specific conditions are satisfied
- \* Let  $X_0 = (S_0, O_0, M_0)$  be the initial state of a system, where  $M$  is the access matrix
- \* The set of state transitions is represented as a set of operations  $t_1, t_2, t_3$ , and successive states are represented as  $X_1, X_2 \dots$  where the notation

$$X_i \vdash_{t(i+1)} X_{i+1}$$

means that the state transition  $t_{i+1}$  moves the system from state  $X_i$  to  $X_{i+1}$

- \*  $\vdash^*$  denotes the transitive closure of  $\vdash$

- \* Every command corresponds to a sequence of transition operations

# HRU: Primitive Operations

- \* We define a set of primitive commands that alter the access control matrix **M**

- ✓ **create (subject) s**
- ✓ **create (object) o**
- ✓ **destroy (subject) s**
- ✓ **destroy (object) o**
- ✓ **enter (right) r into M[s,o]**
- ✓ **delete (right) r from M[s,o]**

# Transitions

**State transitions** are modeled by command execution, where S, O and M are the subjects objects and access matrix before execution and S', O' and M' are the subjects objects and access matrix after execution

Operation	Conditions	New State
<b>enter r into <math>M[s,o]</math></b>	$s \in S, o \in O$	$S' = S$ $O' = O$ $M'[s,o] = M[s,o] \cup \{r\}$ $M'[s_1,o_1] = M[s_1,o_1] \quad \forall (s_1,o_1) \neq (s,o)$
<b>delete r from <math>M[s,o]</math></b>	$s \in S, o \in O$	$S' = S$ $O' = O$ $M'[s,o] = M[s,o] \setminus \{r\}$ $M'[s_1,o_1] = M[s_1,o_1] \quad \forall (s_1,o_1) \neq (s,o)$
<b>if <math>s \notin S</math> or <math>o \notin O</math></b>		then $(S',O',P') = (S,O,P)$ and the request fails

# Transitions

Assumption: all subjects are objects  $S \subseteq O$

Operation	Condition	New State
<b>create (subject) <math>s'</math></b>	$s' \notin O$	$S'=S \cup \{s'\}$ $O'=O \cup \{s'\}$ $M'[s,o]=M[s,o] \quad \forall s \in S, o \in O$ $M'[s',o]=\emptyset \text{ per } o \in O$ $M'[s,s']=\emptyset \text{ per } s \in S$
<b>create (object) <math>o'</math></b>	$o' \notin O$	$S'=S$ $O'=O \cup \{o'\}$ $M'[s,o]=M[s,o] \quad \forall s \in S, o \in O$ $M'[s,o']=\emptyset \text{ per } s \in S$

# Transitions

Assumption: all subjects are objects  $S \subseteq O$

Operation	Condition	New State
<b>destroy (subject) <math>s'</math></b>	$s' \in S$	$S' = S \setminus \{s'\}$ $O' = O \setminus \{s'\}$ $M'[s,o] = M[s,o] \quad \forall s \in S', o \in O'$
<b>destroy (object) <math>o'</math></b>	$o' \in O$ $o' \notin S$	$S' = S$ $O' = O \setminus \{o'\}$ $M'[s,o] = M[s,o] \quad \forall s \in S', o \in O'$

If the conditions are not satisfied then  $(S,O,M) = (S',O',M')$  and the request fails

# Commands

\* A command is structured as follows:

**command** name ( $o_1, o_2, \dots, o_k$ )

**if**     $r_1$  in  $M[s_1, o_1]$

$r_2$  in  $M[s_2, o_2]$

...

$r_m$  in  $M[s_m, o_m]$

**then**

$op_1$

$op_2$

...

$op_n$

**end**

r = right

op = primitive operation

# HRU: Example

**command** create\_file (s,f)

create f  
enter 'owner' in M[s,f]  
enter 'read' in M[s,f]  
enter 'write' in M[s,f]

**end**

✓ When **s** creates a file **f**, **s** is the owner of **f** with read and write access on the file

**command** grant\_read (s,p,f)

**if** 'owner' in M[s,f]

**then**

enter 'read' in M[p,f]

**end**

Wiene eseguito solo se adeseguirà il comando  
e' l'owner

✓ The owner **s** of the file can grant read access to another subject **p**

# HRU: Definitions

- \* An authorization system is defined by a set of commands and by its state, captured by the access matrix
- \* The effect of a command is recorded as a change to the access matrix
- \* The HRU model can capture security policies regulating the allocation of access rights
- \* To **verify** that a system complies with such a policy, you have to check that there exists **no way for undesirable access rights to be granted**

# Take-Grant Systems

can the safety of a particular system, with specific rules, be established?

\* Consider the protection models Take-Grant

\* Four primitive operations:

- ✓ **create**
- ✓ **remove**
- ✓ **take (t)**
- ✓ **grant(g)**

\* Subjects **S** (denoted ), objects **O** (denoted ) , rights **R**

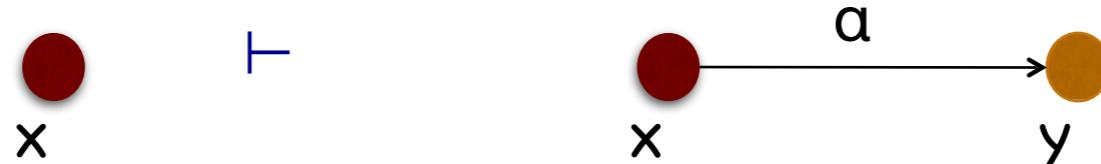
\* Graph: Nodes = **S<sub>u</sub>O** (the nodes that can be both objects and subjects are denoted as ) , Edges = **SxO**, label(**s,o**) = **M[s,o]**

Select objects some definite  
in memory form the eis system

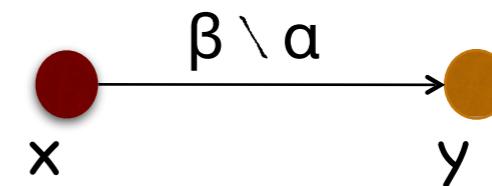
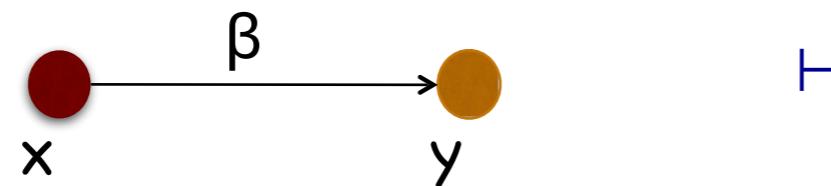
selection is possible

defining tells do the is possible the future situation

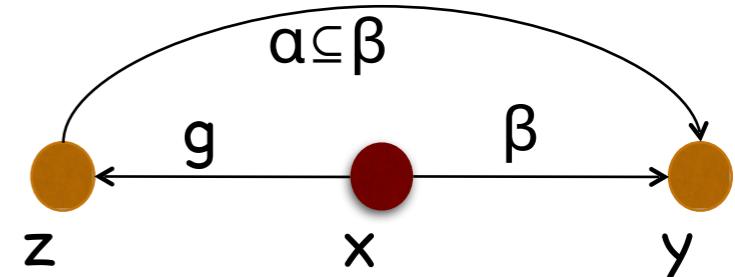
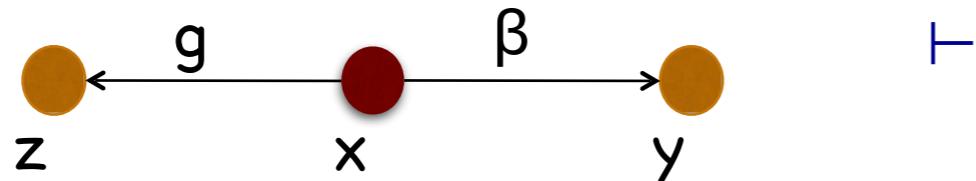
**Creation of a  
subject/object :**  
**create(y,x)**



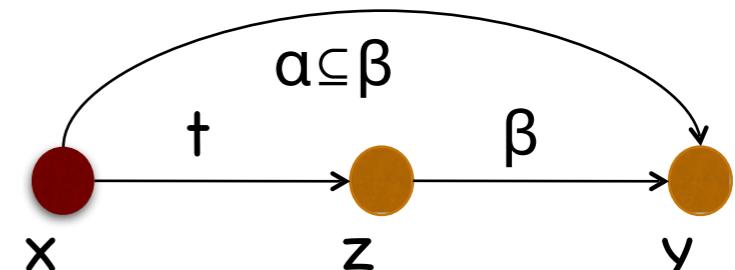
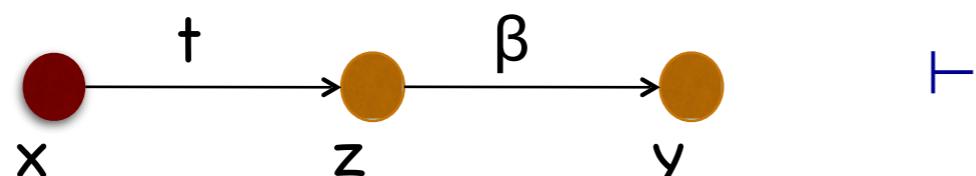
**Remove access  
rights:**  
**remove(y,x)**



**Grant access  
rights:**  
**grant(z,y,x)**



**Take access  
rights:**  
**take(z,y,x)**



# Decidability

- \* Let us define the predicate **can-share**
- \* **can-share (a,x,y,G<sub>0</sub>) = true** iff exists a sequence of protection graphs  $G_1, \dots, G_n$  such that  $G_0 \vdash^* G_n$  using only the 4 rules of the take-grant model, and in  $G_n$  there is an edge from **x** to **y** with label **a**.

## Theorem

There exists an algorithm of complexity **O(|V|+|E|)** that verifies the predicate **can-share**, where **V** is the set of nodes and **E** the set of edges of  $G_0$ .

- \* This means that it is possible to decide if:
  - \* **s** can share **o** with **s'**
  - \* **s** can take the access to **o** from **s'**

Quali sono gli approcci al controllo degli accessi?  
Dove andrebbero essere

## Limits of DAC

- \* Managing a policy is a complex task in a large system
- \* Set of subjects or objects is large
- \* Set of subjects or objects change frequently
- \* **Flexible** mechanism prone to errors:
  - ✓ it is necessary that all users understand the mechanism and respect the security policy
  - ✓ there is no control on the flow of information
    - non scalabile
    - può andare fuori controllo

Il potere è fatto di sistemi  
il soggetto non ha potere

## Mandatory Access Control: MAC

↓  
"mondo welfare"

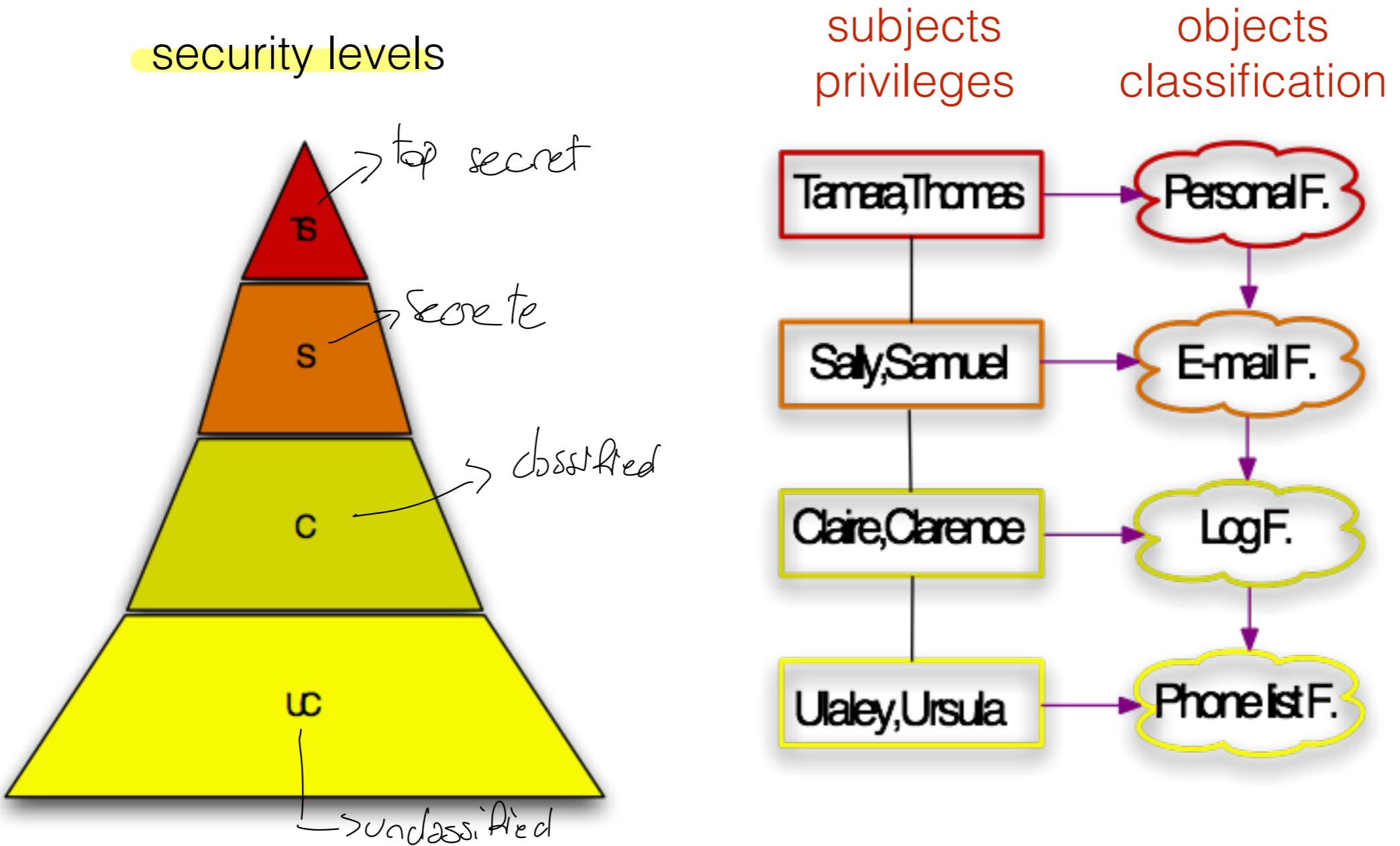
# Mandatory Access Control: MAC

↳ Decide if system  
→ Discretionali dell'utente

- \* Decision on access control are made based on the **labels of objects** that represent how critical the objects are and on the **levels of authorizations of the subjects**
- \* MAC is often identified with **multi-level security**
- \* **Access rights to objects are established by a central authority and not by the owner**
  - ✓ the access control is **mandatory**
  - ✓ moves the control from the users to the system
- \* Less flexible than DAC but more secure

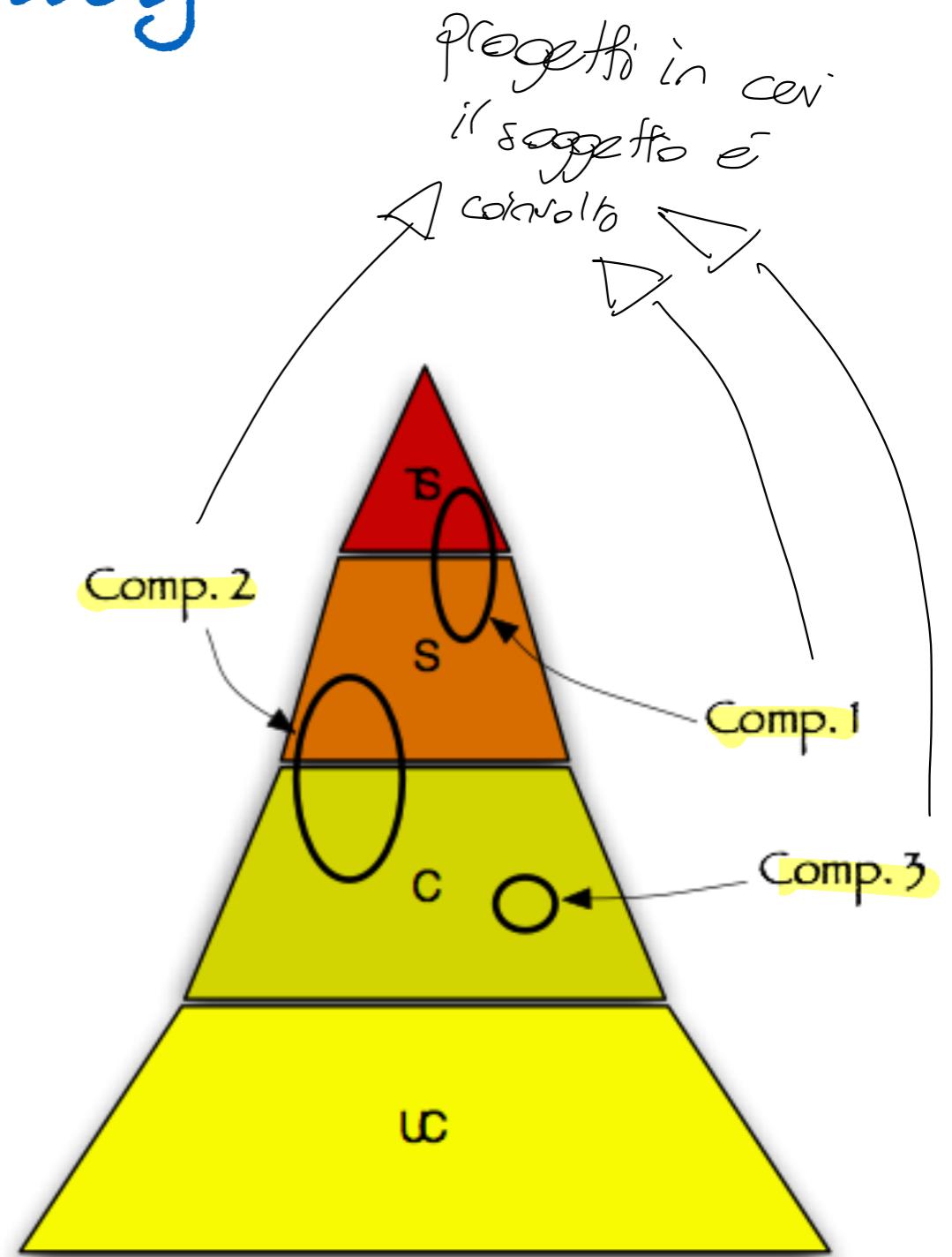
# Multi-level Security

É :/ systems de sécurité



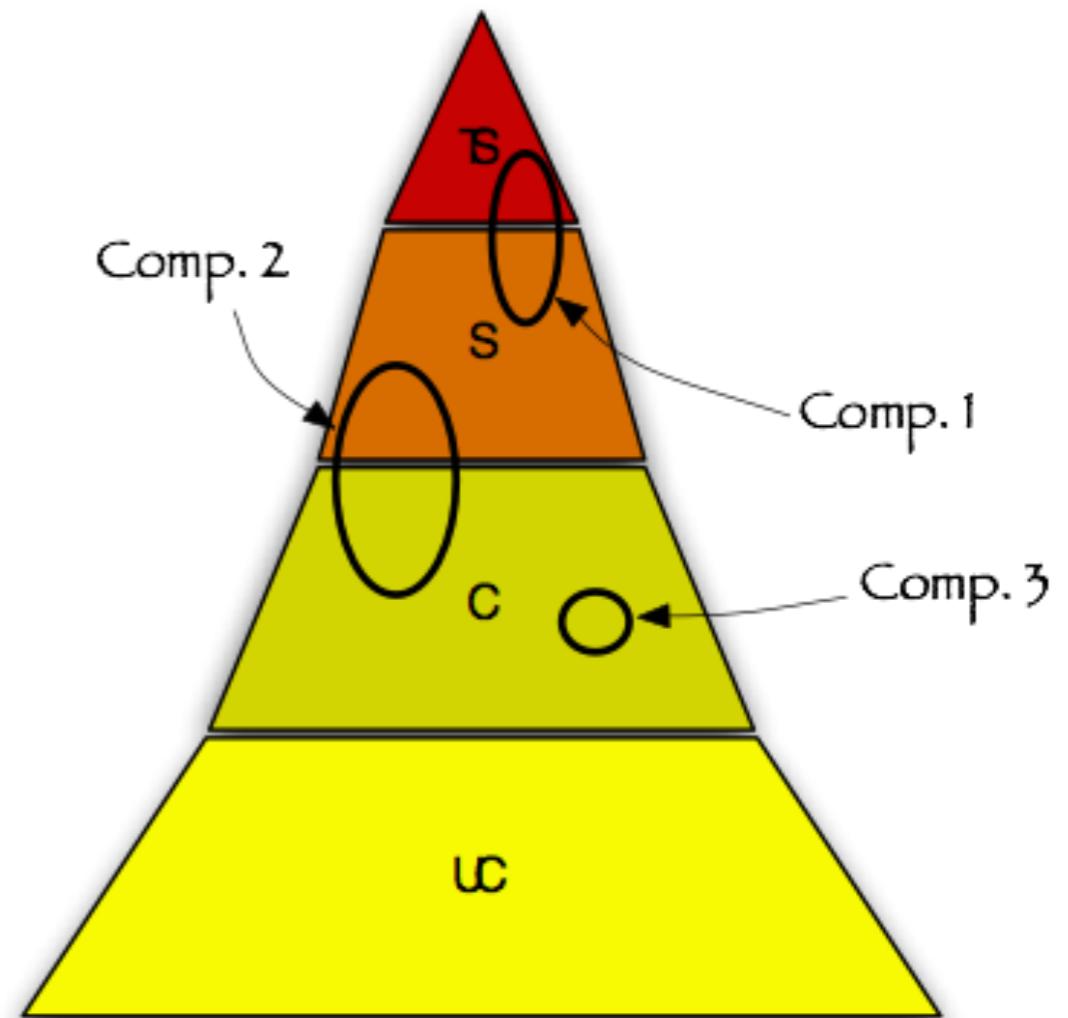
# Military Policy

- \* Access to information is based on the **need-to-know** principle
- \* Classified information is associated to one or more **compartments** that describe the subjects  
*↳ pic simple implementare  
il need to know*
- ✓ we could have two projects A and B that need to access secret information but their access to secret information is limited
- \* Compartments can include information at different levels in the classification
- \* A single element can belong to more than one compartment



# Military Policy

- \* Class = (level,compartment)  
defines the sensitivity of the information.
- \* Access has to be authorized
- \* Rights = (level,compartments)
- \* Used to limit access, s read o iff:
  - ✓ the security level of s is at least the one of o
  - ✓ s needs to know all compartments to which o belongs



# Lattice and Security Levels

- In a policy where a subject may access object only if the subject's security level is higher than the object's security level, we wish to have **unique answer** to the following questions:
  - \* Given two objects at different security levels, what is the **minimal** security level a subject must have to be allowed to read both objects?
  - \* Given two subjects at different security levels, what is the **maximal** security level an object can have so that it still can be read by both subjects?
  - \* **Lattice of security levels**
    - ✓ greatest lower bound
    - ✓ least upper bound

# Lattice and Security Levels

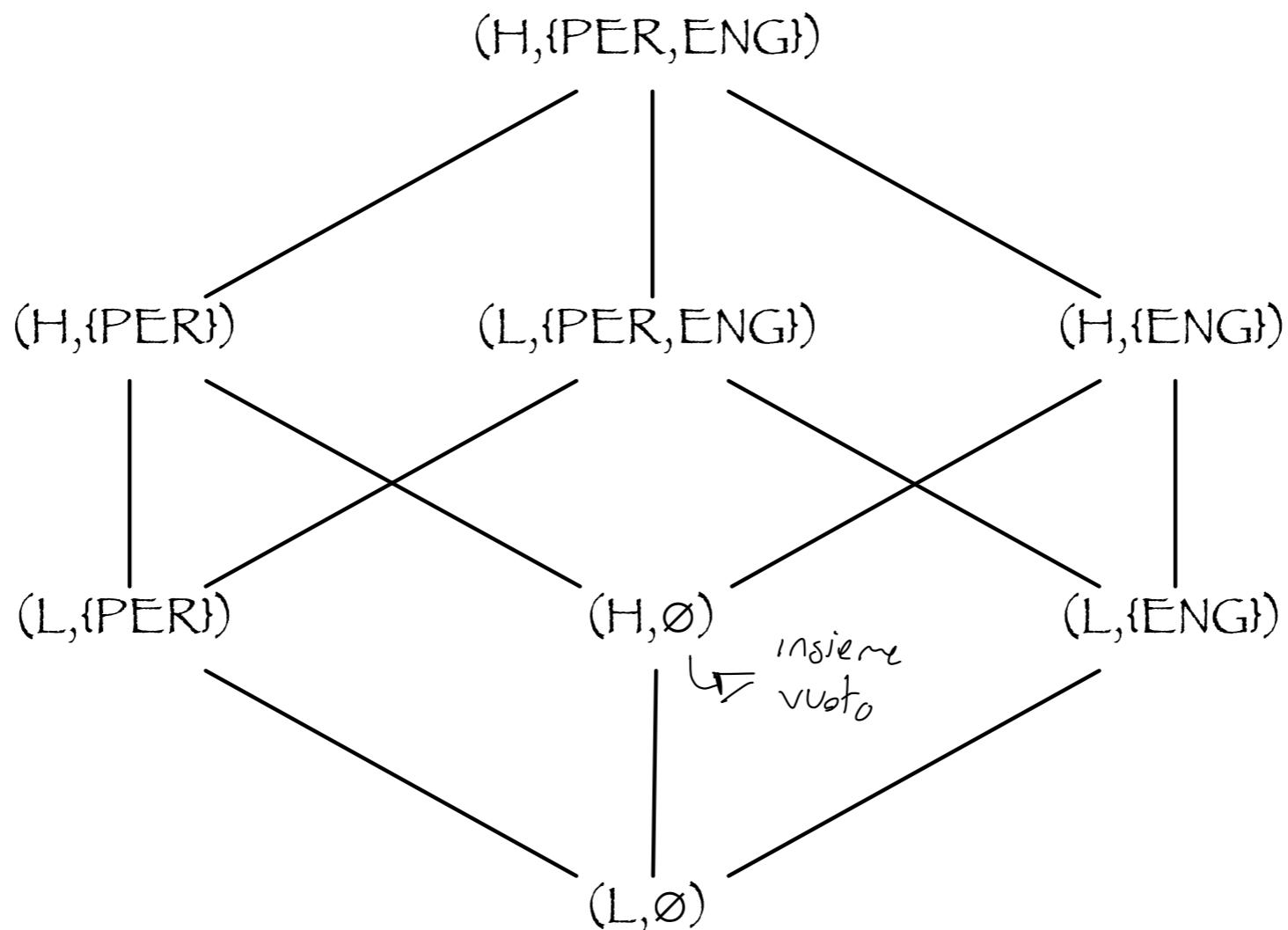
Indeed, with linear ordering you can only express a limited set of security policies. In order to limit the access to the information of a secret project X only to people working on X we need the following security lattice:

- \* Let  $\mathbf{H}$  be a set of security levels with hierarchical ordering  $\leq_{\mathbf{H}}$ ;
- \* Let  $\mathbf{C}$  be a set of compartments (project names, company division, academic departments, ...)
- \* A security label is a pair  $(\mathbf{h}, \mathbf{c})$  with  $\mathbf{h} \in \mathbf{H}$  and  $\mathbf{c} \subseteq \mathbf{C}$
- \* We define a partial order over security labels as follows:

$$(\mathbf{h}_1, \mathbf{c}_1) \leq (\mathbf{h}_2, \mathbf{c}_2) \quad \text{iff} \quad \mathbf{h}_1 \leq_{\mathbf{H}} \mathbf{h}_2 \quad \text{and} \quad \mathbf{c}_1 \subseteq \mathbf{c}_2$$

# Example of Lattice of Security Labels

- \* Security levels: public (L), private (H)
- \* Compartments: personell (PER), engineering (ENG)
- \* Ordering relations:  
 $(L,\{PER\}) \leq (H,\{PER\}), (L,\{PER\}) \leq (L,\{PER,ENG\}), (L,\{PER\}) \leq (H,\{ENG\})$



# Access Control Models

- \* Access control models can be classified according to different features:
  - focus solo confidenzialità → chi le avverte
- \* The capture confidentiality policies (**Bell-LaPadula**), integrity policies (**Biba**, Clark-Wilson) or both (Chinese Wall)
- \* There are models that can be applied to environments with static policies (Bell-LaPadula), and others that allow dynamic changes in the access rights (Chinese Wall)
- \* Some models apply to military environments (Bell-LaPadula) others to commercial environments (Clark-Wilson, Chinese Wall)
- \* Security models can be informal (Clark-Wilson) or formal (Bell-LaPadula, Harrison-Ruzzo-Ullman).

# The Bell-Lapadula Model

- \* Typical MAC model based on multilevel security .

- ✓ Security clearance of subjects
- ✓ Security classification of objects
- ✓ Control access matrix

Combina MAC e DAC  
I  
N' d'ce  
di ha  
accesso  
cosa  
I  
posso  
cedere accesso  
I  
Ne rispetta  
di MAC e DAC

## GOAL

prevent read access to objects with security classification higher than the subject's clearance

Information cannot flow from high to low

- \* Discretionary vs Mandatory: BLP combines mandatory and discretionary access controls. MAC is defined based on security levels of the objects and clearance of the subjects. Moreover, a subject may grant another subject access to an object constrained by the MAC rules.

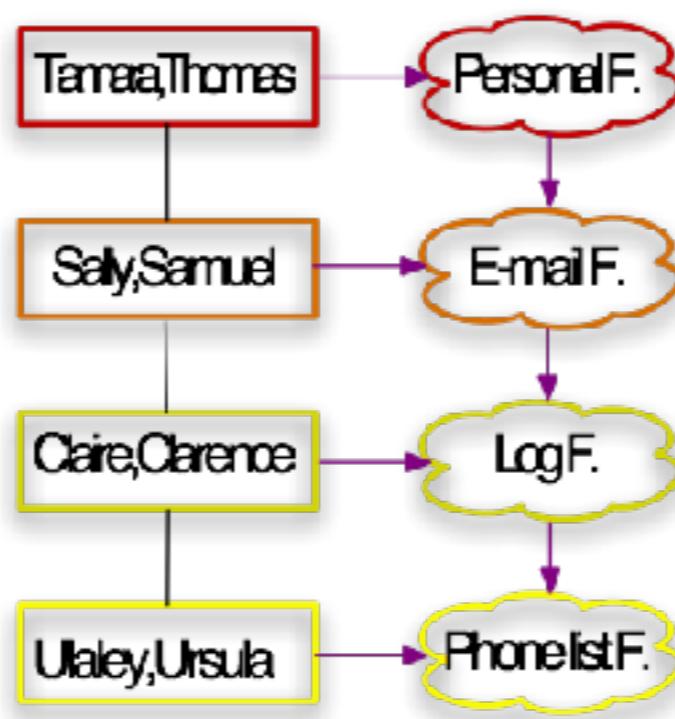
# BLP: Simple Security Property

Simple Security Property (no reads up)

**S** can read **O** iff  $h_o \leq h_s$  and **S** has read DAC on **O**

↳ sono proibite letture verso l'alto, sono proibite letture verso l'alto e scritture verso il basso  
Ex: Claire and Clarence cannot read personal files  
Es: Tamara and Sally can read log files

GARANTISCE  
CONFIDENZIALITÀ



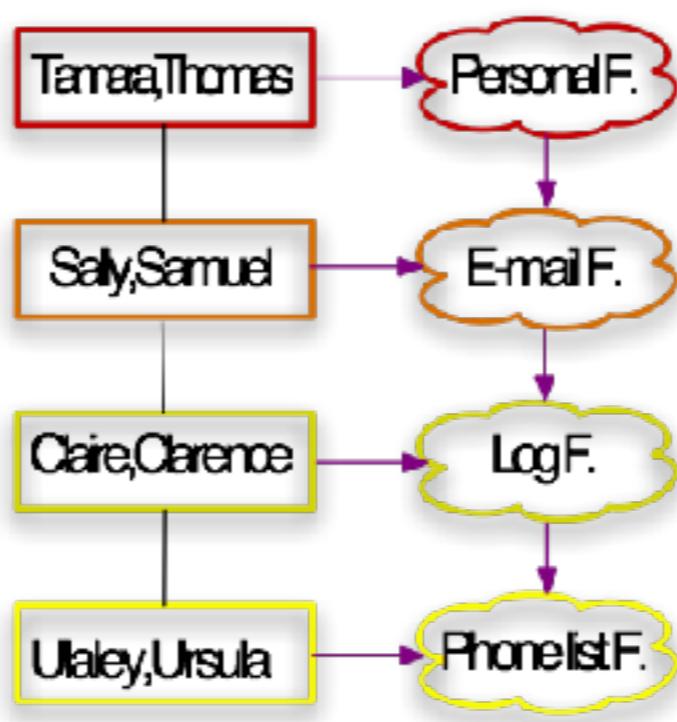
- \* Problem: Tamara could copy the content of personal file into log files, thus allowing Claire to access personal data!

# BLP Star-Property

Star-Property, Preliminary Version (no write down)

**S** can write **O** iff  $h_s \leq h_o$  and **S** has write DAC on **O**

Ex: Tamara cannot write on log files since  $h_{Tamara} > h_{Log\ F.}$



**Secure system** A system is secure if both ss-property and star-property hold.

**State** representation of subjects that have current access to objects and the current access rights

**Basic Security Theorem, Preliminary Version:** If a system has a secure initial state and every possible state transition preserves the ss-property and the star-property preliminary version, then every possible state of the system is secure.

Possé trasferire un diritto  
solo se non viola le  
MAC rules

## BLP ds-property

- \* DAC: subjects that can access an object can transfer the access rights of that objects to other subjects
- \* In BLP this is expressed by the ds-property

↳ Bell - LAPADULA

### ds-property

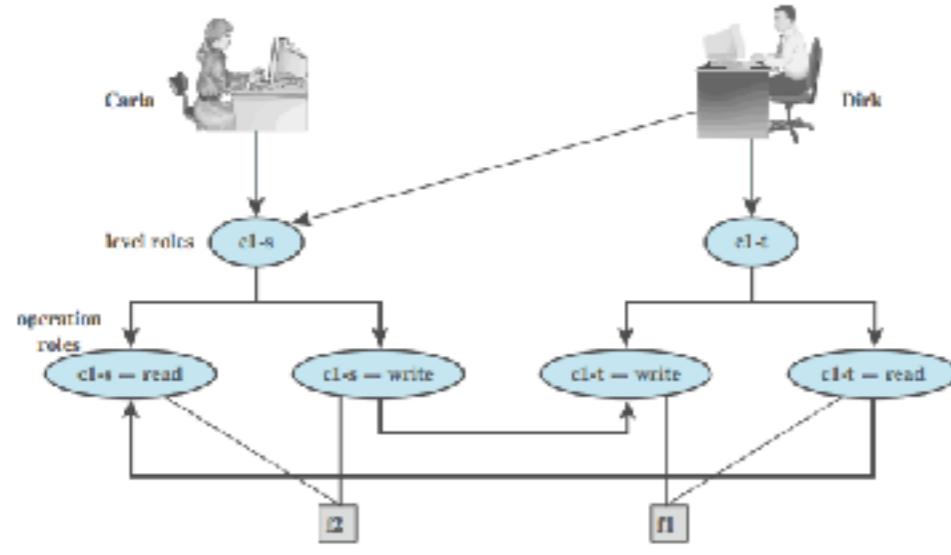
A subject may grant to another subject access to an object (on the owner's discretion), constrained by the MAC rules. Thus, a subject can exercise only accesses for which it has the necessary authorization and which satisfy the MAC rules.

A system is **secure** if it satisfies the ss-property, the star-property and the ds-property.

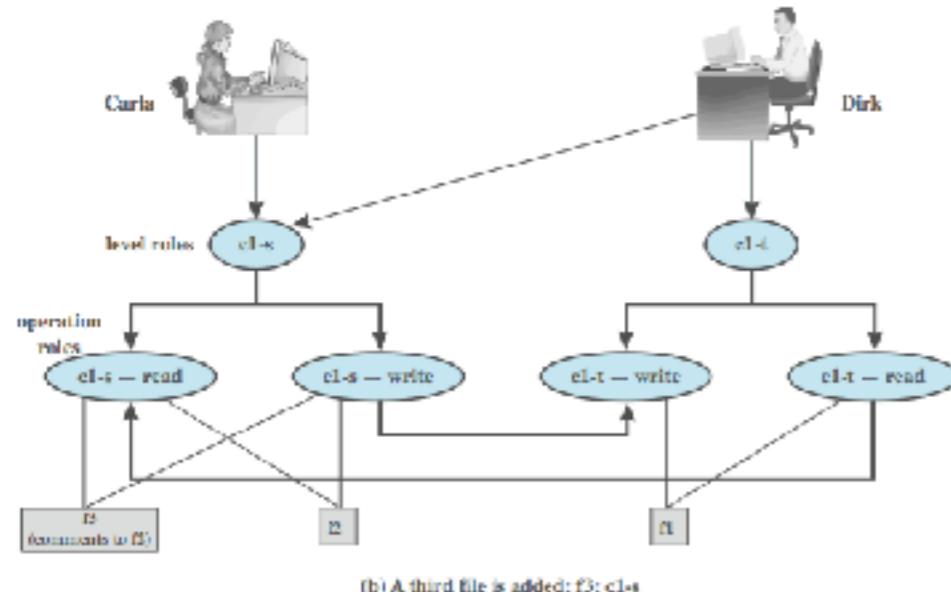
secretaire > secrete student

# BLP: Example

Carla reads and writes f2, Dirk reads and writes f1 and reads f2  
if Carla allows him, but he cannot write f2 for the \*-property



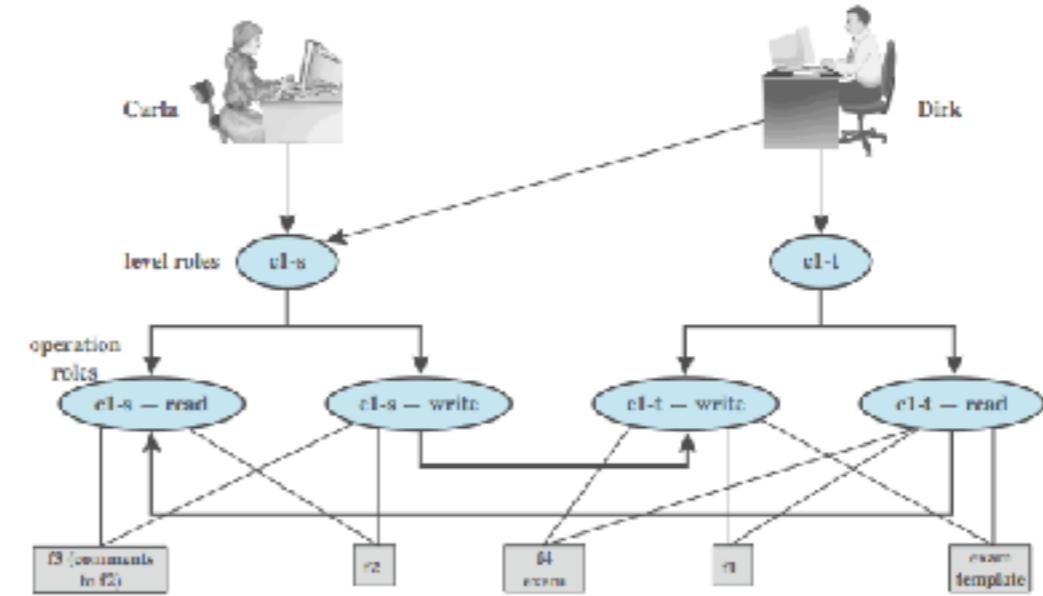
(a) Two new files are created:  $f1$ :  $c1-t$ ;  $f2$ :  $c1-s$



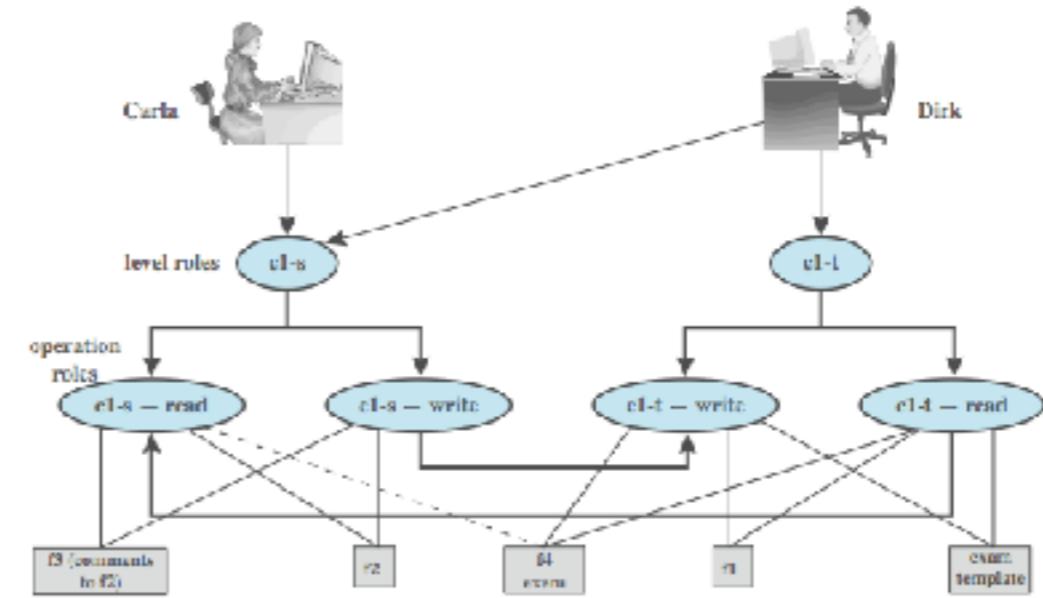
(b) A third file is added:  $f3$ :  $c1-s$

Figure 10.2 Example of Use of BLP Concepts (page 1 of 3)

Dirk writes an exam  $f4$  based on the exam templates, so he has to do it as a teacher



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



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

Dirk reads  $f2$  and wants to create a file with the comments for Carla  
Dirk logs as a student and creates  $f3$  so that Carla can read it

Carla cannot read the exam  $f4$  because this violates the ss-  
property. Dirk has to **declassify**  $f4$  so that Carla can read it.  
The system administrator should do this downgrade

→ da direttore all' amministratore del sistema

# BLP: Example

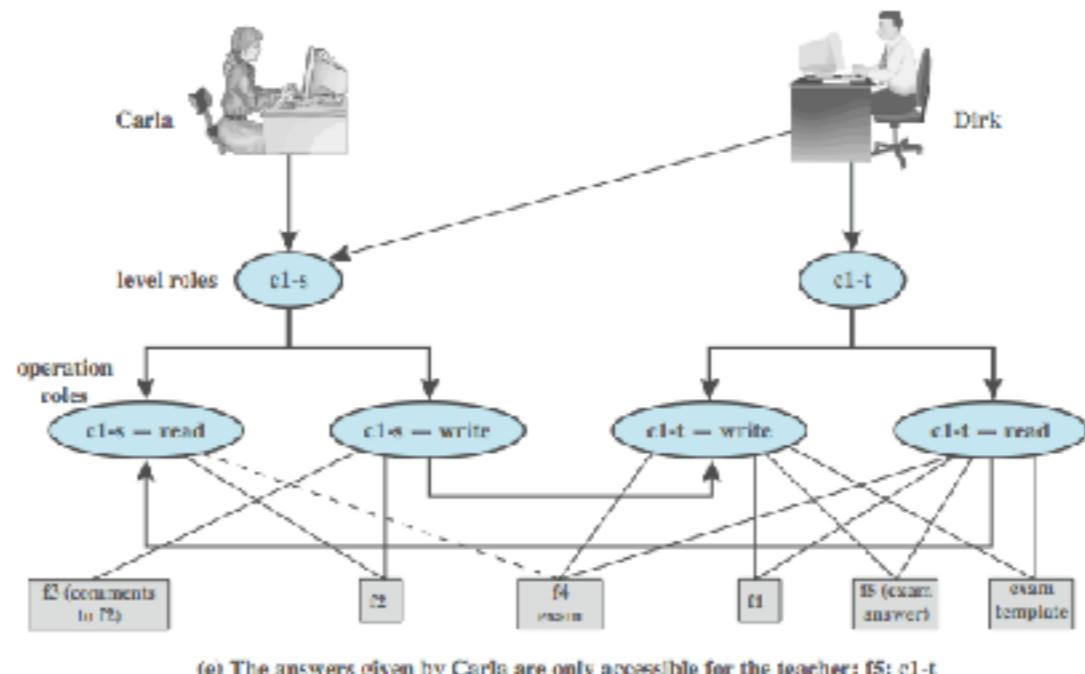


Figure 10.2 Example of Use of BLP Concepts (page 3 of 3)

Carla writes the results of the exam in f5, she has to do it at the teacher level so that only Dirk can read it. This is allowed since BLP allows writing up.

# BLP: Advantages and limitations

\* BLP is a very significant security model

- ✓ Descriptive capabilities: the BLP state describes the current access operations and all current access permissions
- ✓ The security policies are based on security levels and access control matrix. It is easy to change structures for new policies.
- ✓ By changing the properties we can handle integrity (Biba model)

\* BLP does not cover all aspects of security

- ✓ It only deals with confidentiality (no integrity)
- ✓ It does not address the management of access control
- ✓ It contains covert channels
  - ✓ A low-level subject creates an object dummy.obj at its own level
  - ✓ Its high-level accomplice (or trojan horse) upgrades the security level of dummy.obj to high or leaves it unchanged
  - ✓ Later the low-level subject tries to read dummy.obj. Success or failure of this request reveals the action of the high level subject. One bit of information has been transmitted from high to low.

# Modello di Biba (1977)

Biba studied the **integrity** of systems and he proposed three policies, one of which is the mathematical dual of the Bell-LaPadula Model

- \* Subjects **S**, objects **O** and integrity levels **I** ordered ( $\leq$ )
- \*  $i: S \cup O \rightarrow I$  returns the integrity level of subjects and objects
  - ✓ Data at higher level is **more accurate** or reliable (with respect to some metrics) than data at a lower level
  - ✓ Integrity labels **inhibit data manipulation**
- \* Three relations:
  - ✓  $r \subseteq S \times O$ , **s** reads **o**
  - ✓  $w \subseteq S \times O$ , **s** writes/modifies **o**
  - ✓  $x \subseteq S \times S$ , **s** invokes **s'**

# Biba: Integrity Policy

- \* Integrity  $\neq$  security
  - ✓ **Security** labels primarily limits the **flow of information**
  - ✓ **Integrity** labels primarily inhibit the **modification of information**
- \* This policy is the dual of the BLP, and it **prevents that trusted objects and subjects are corrupted by untrusted information**
  1. **Simple integrity property (no write up)**  $s \in S$  can write  $o \in O$  iff  $i(o) \leq i(s)$
- \* This first rule forbids write up
  - ✓ If a subject can alter an object with a higher degree of trust, he/she could inject false or corrupted data

# Biba: Integrity Policy

- \* This policy is the dual of the BLP, and it prevents that trusted objects and subjects are corrupted by untrusted information
  - 1. Simple integrity property (no write up)  $s \in S$  can write  $o \in O$  iff  $i(o) \leq i(s)$
  - 2. Integrity star-property (no read-down)  $s \in S$  can read  $o \in O$  iff  $i(s) \leq i(o)$
- \* The second rule forbids to read down
  - ✓ it requires that a subject can read only objects of higher integrity in order to disallow a subject to contaminate a higher object with data from a lower object
- \* These rules implies that a subject  $s$  can read and write an object  $o$  iff  $i(s) = i(o)$ .

# Biba: Integrity Policy

- \* This policy is the dual of the BLP, and it prevents that trusted objects and subjects are corrupted by untrusted information
  - 1. Simple integrity property (no write up)  $s \in S$  can write  $o \in O$  iff  $i(o) \leq i(s)$
  - 2. Integrity star-property (no read-down)  $s \in S$  can read  $o \in O$  iff  $i(s) \leq i(o)$
  - 3. Invoke property  $s_1 \in S$  can invoke  $s_2 \in S$  iff  $i(s_2) \leq i(s_1)$
- \* The third rule forbids to a subject to invoke a subject of higher integrity level, otherwise he/she would control more reliable subjects

# Biba: Low-watermark policy dynamic integrity levels

- \* The following integrity rules automatically adjust the integrity level of an entity if it comes into contact with dirty information.
  1. Simple integrity property:  $s \in S$  can write  $o \in O$  iff  $i(o) \leq i(s)$
  2. Subject low watermark property: A subject  $s$  can read objects at any integrity level. If  $s \in S$  reads  $o \in O$  then  $i'(s) = \min(i(s), i(o))$
  3. Invoke property:  $s_1 \in S$  can invoke  $s_2 \in S$  iff  $i(s_2) \leq i(s_1)$
- \* The second rules lowers the integrity level of a subject to the one of the objects that he/she reads
  - ✓ The subjects accesses data that are less reliable than what he/she is

Role based Access Control: RBAC

# DAC do not scale

	objects		
	news.doc	photo.png	fun.com
alice	read	view edit	view
bob	read write	view edit	view modify
charlie			
dave		view	

- \* **Access control matrix**

	news.doc	photo.png	fun.com
bob	read write		
alice		view, edit	
bob		view, edit	
dave		view	

- \* **Access control list**

	news.doc	photo.png	fun.com
Alice's capability	read	view, edit	view
Bob's capability			

	news.doc	photo.png	fun.com
Bob's capability	read write	view, edit	view edit
Charlie's capability			

Charlie's capability

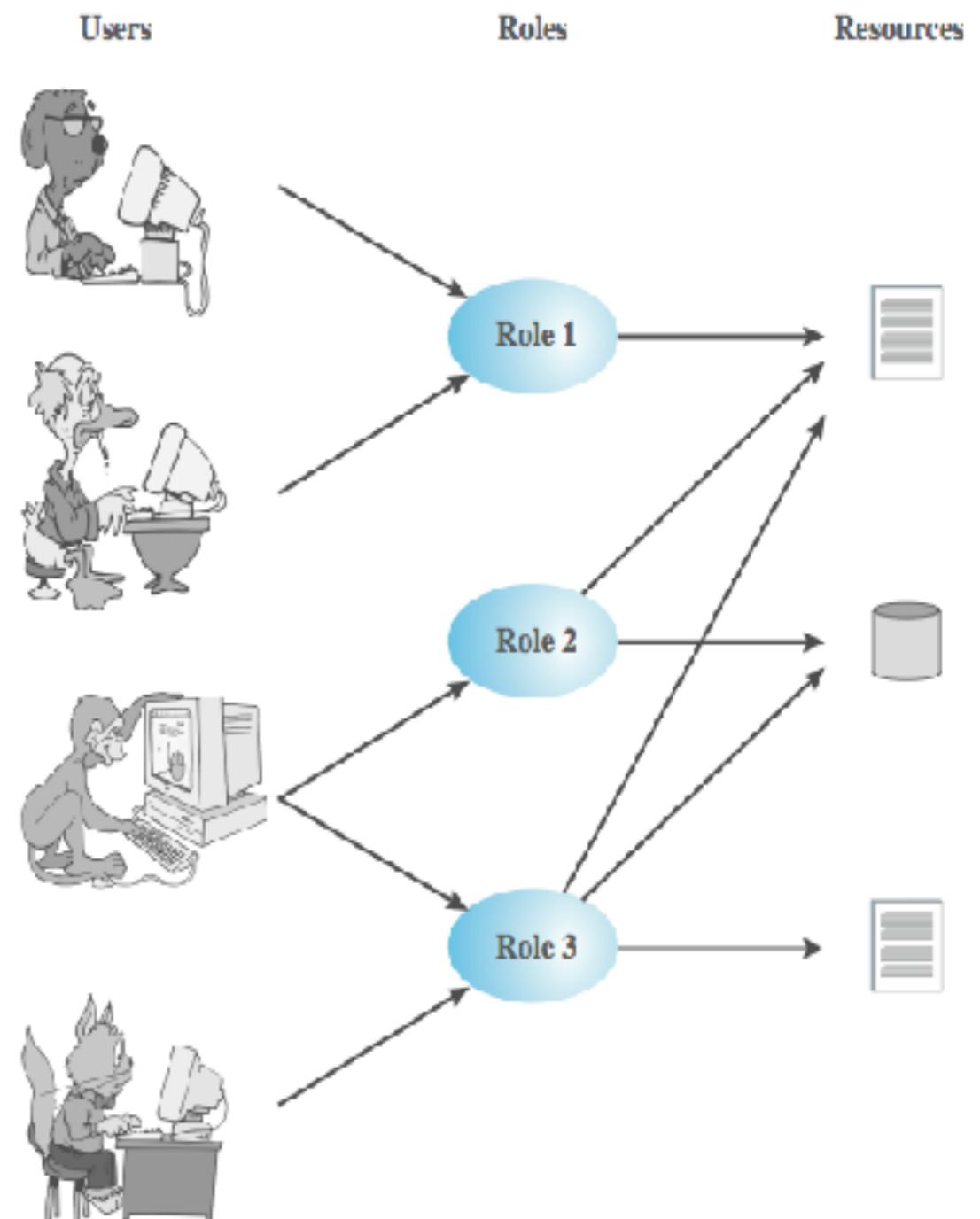
	photo.png
Dave's capability	view

Dave's capability

Idea utilizzare i ruoli e  
non assegnare permessi singolarmente

## RBAC

- \* DAC systems define the access rights of individual users and groups of users
- \* RBAC systems assign access rights to roles instead of individual users. Users are assigned to different roles, either statically or dynamically.
- \* Least privilege / need to know

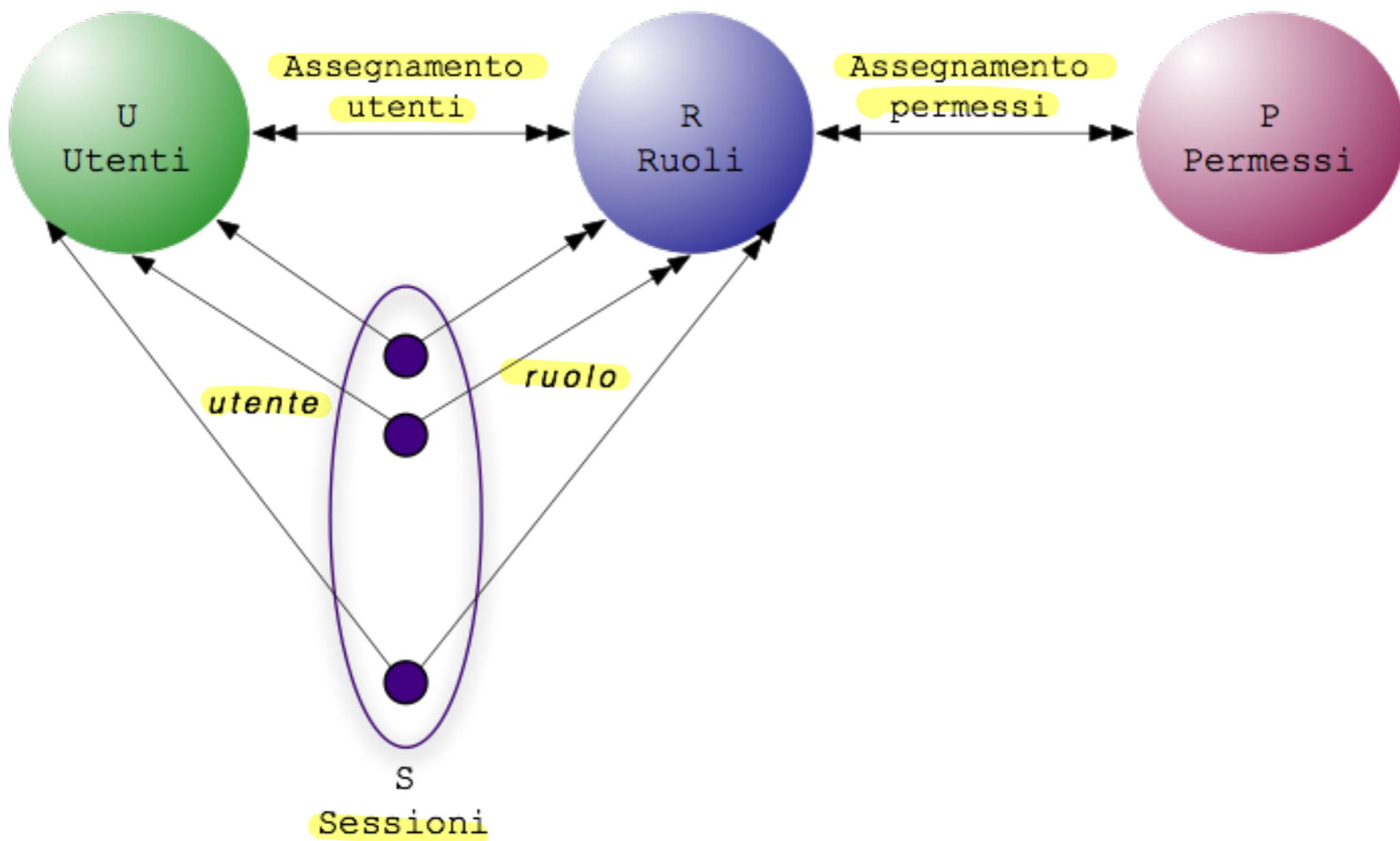


# RBAC Family

- \* Defined by Ravi Sandhu in 1996
- \* ANSI standard since 2004
- \* **RBAC0**: Core model: users, roles, permissions, sessions
- \* **RBAC1**: RBAC0 + Role Hierarchies
- \* **RBAC2**: RBAC1 + Constraints

वर्गीकृति  
अंकोली

# Base Model - RBAC<sub>0</sub>



# Example of roles

## \* Roles

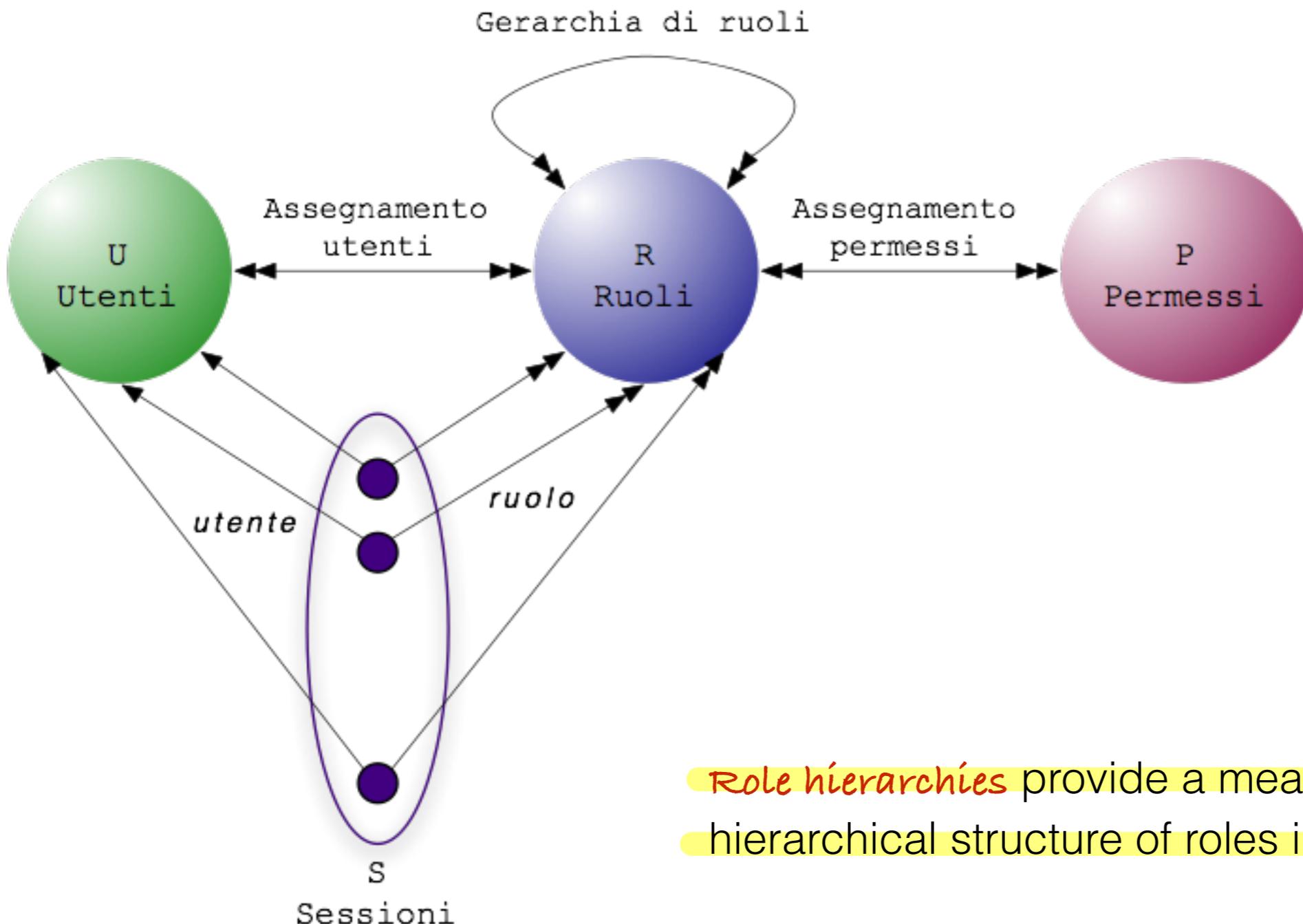
- ✓ Lecturer
- ✓ PhD student
- ✓ Associate Professor
- ✓ Full Professor
- ✓ Head of Department

## \* Objects

- ✓ Exam paper
- ✓ Coursework
- ✓ Module Marks

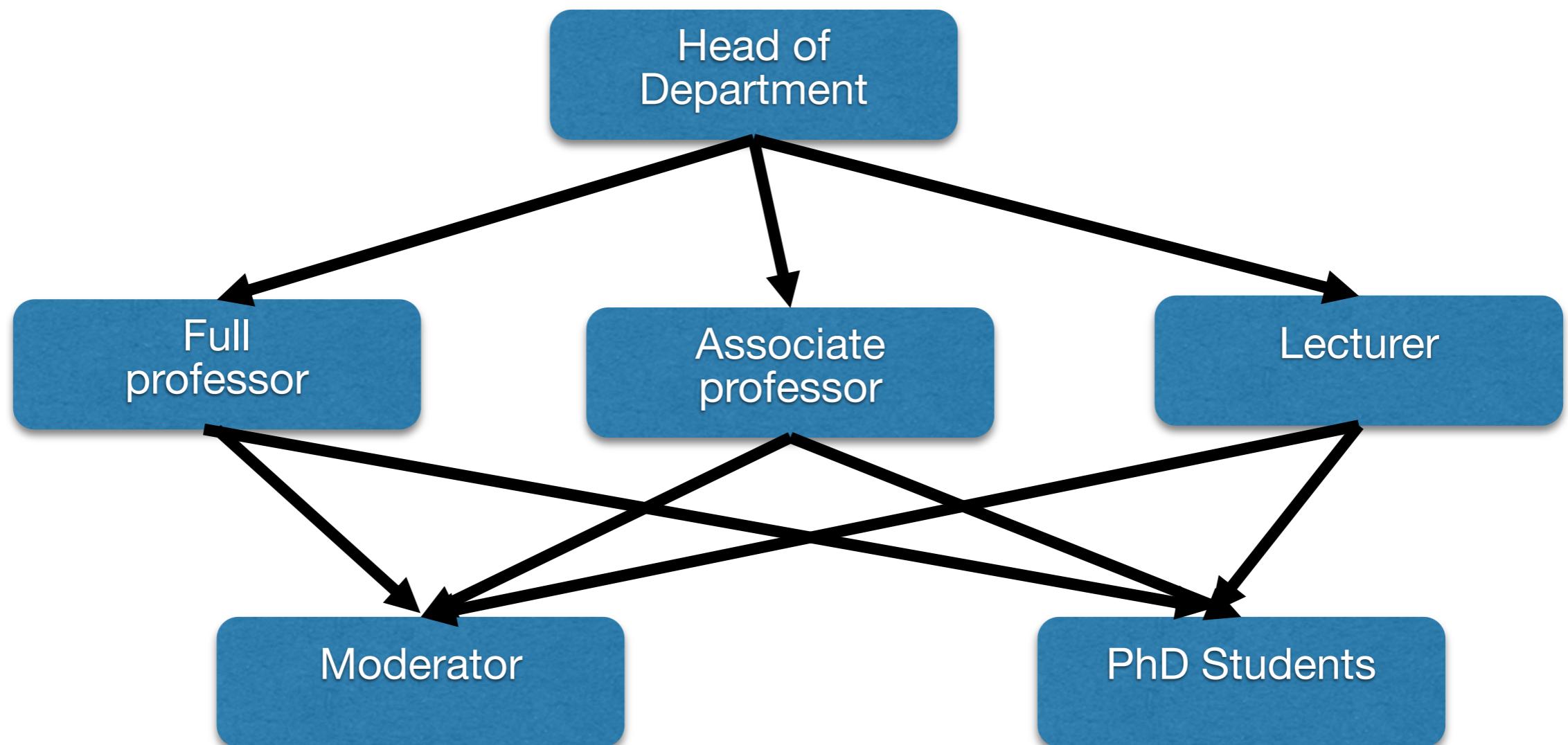
# Role Hierarchies - RBAC<sub>1</sub>

potrai avere gerarchia fra ruoli per editore  
permessi dal figlio al padre

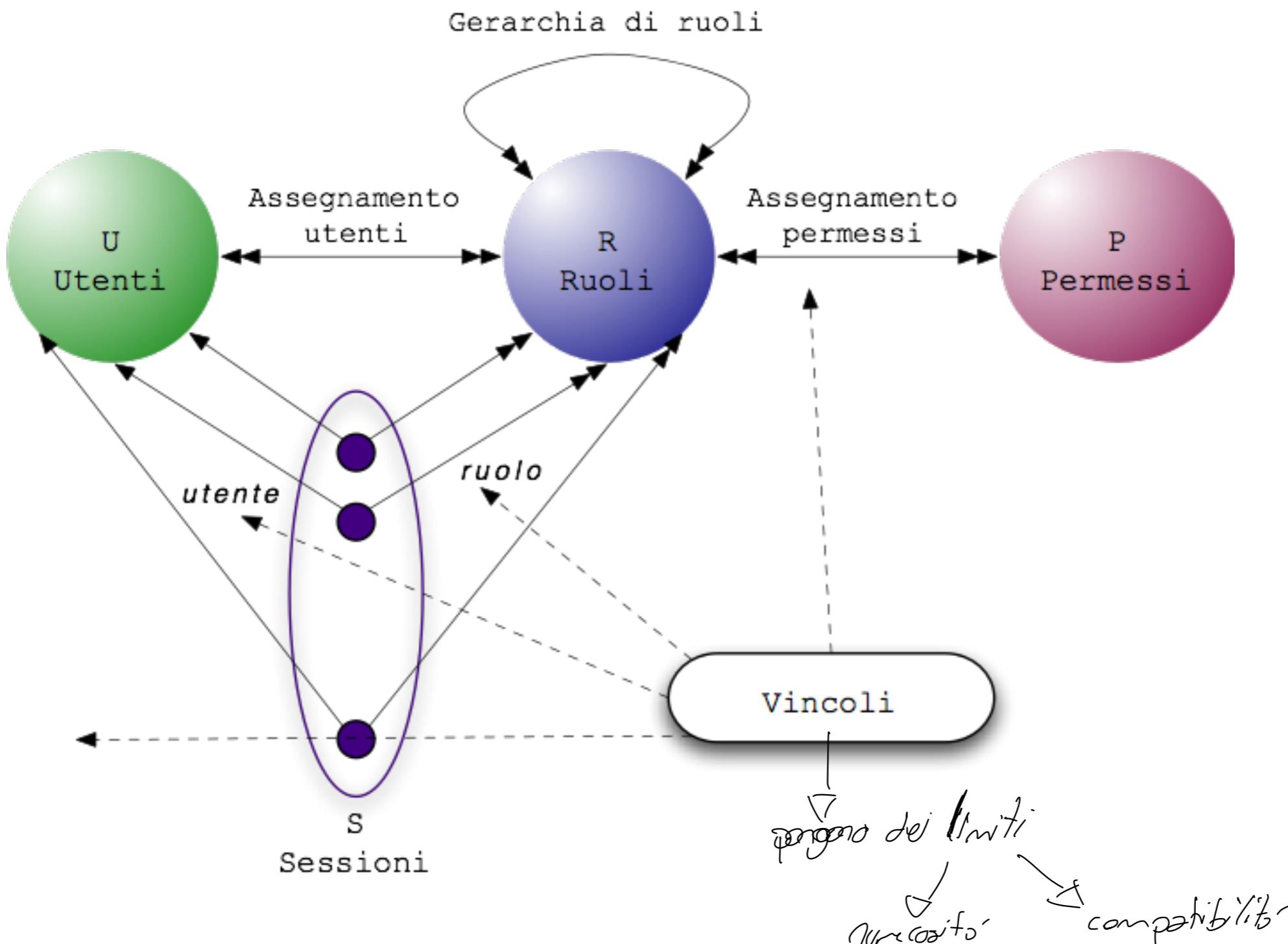


Role hierarchies provide a means of reflecting the hierarchical structure of roles in an organization.

## Role Hierarchies - RBAC<sub>1</sub>



# Consolidated RBAC<sub>2</sub>

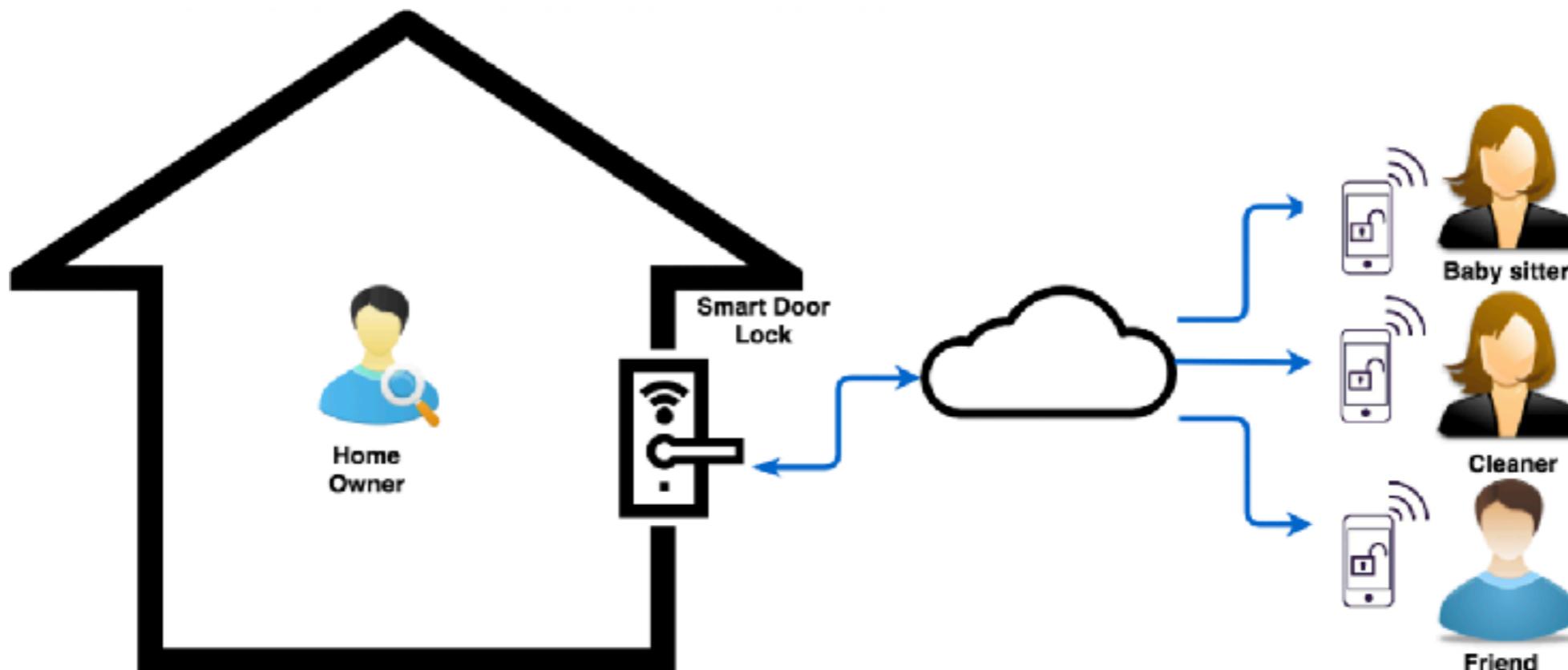


# RBAC: Advantages

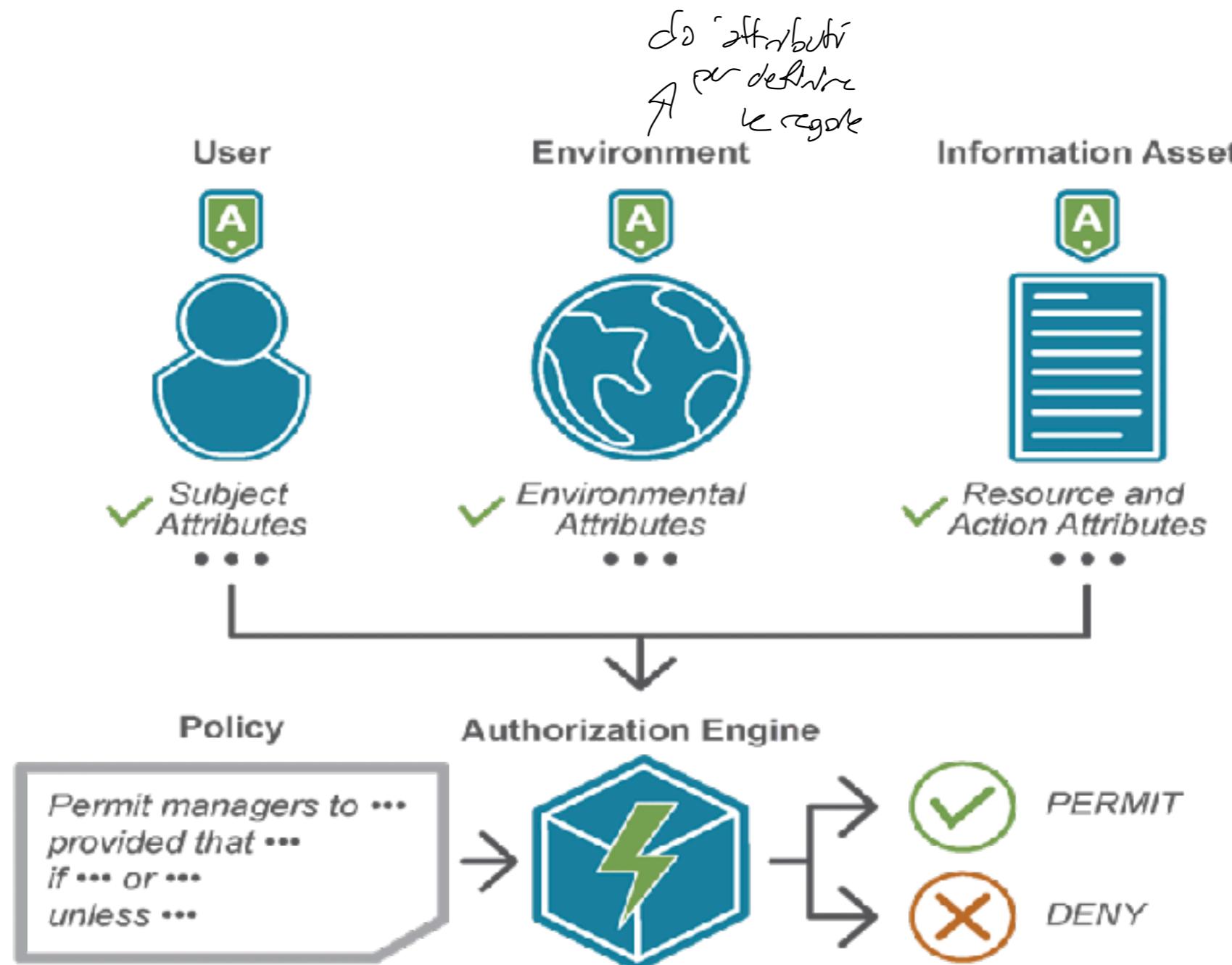
- \* Roles are abstractions of the function present in an organization (it can be seen as collection of procedures)
  - ✓ Roles are different from groups
  - ✓ Roles are more related to responsibility and access rights associated to responsibilities
- \* Abstraction makes access control policies more tractable
- \* When the number of roles is low there is a big advantage in the amount of information that needs to be stored wrt the standard access control matrix
- \* Efficient administering and monitoring of permissions
  - \* No need to manually assign users to permissions
  - \* Automatically done by assigning users to roles
- \* Used in: healthcare (NHS-GB), Banks and insurance policies, commercial products, cloud platforms

# Attribute based access control ABAC

↳ ha anche attributi dell'ambiente e device



# Attribute based access control ABAC



# Summary

- \* Access control specifies who or what may have access to a system resource and the type of access that is permitted
- \* Four types of access control policies
  - ✓ Discretionary
  - ✓ Mandatory
  - ✓ Role-based
  - ✓ Attribute-based
- \* Access control is very hard to implement correctly
  - ✓ Employees are often over-entitled