Resource Access Control (Authorization)

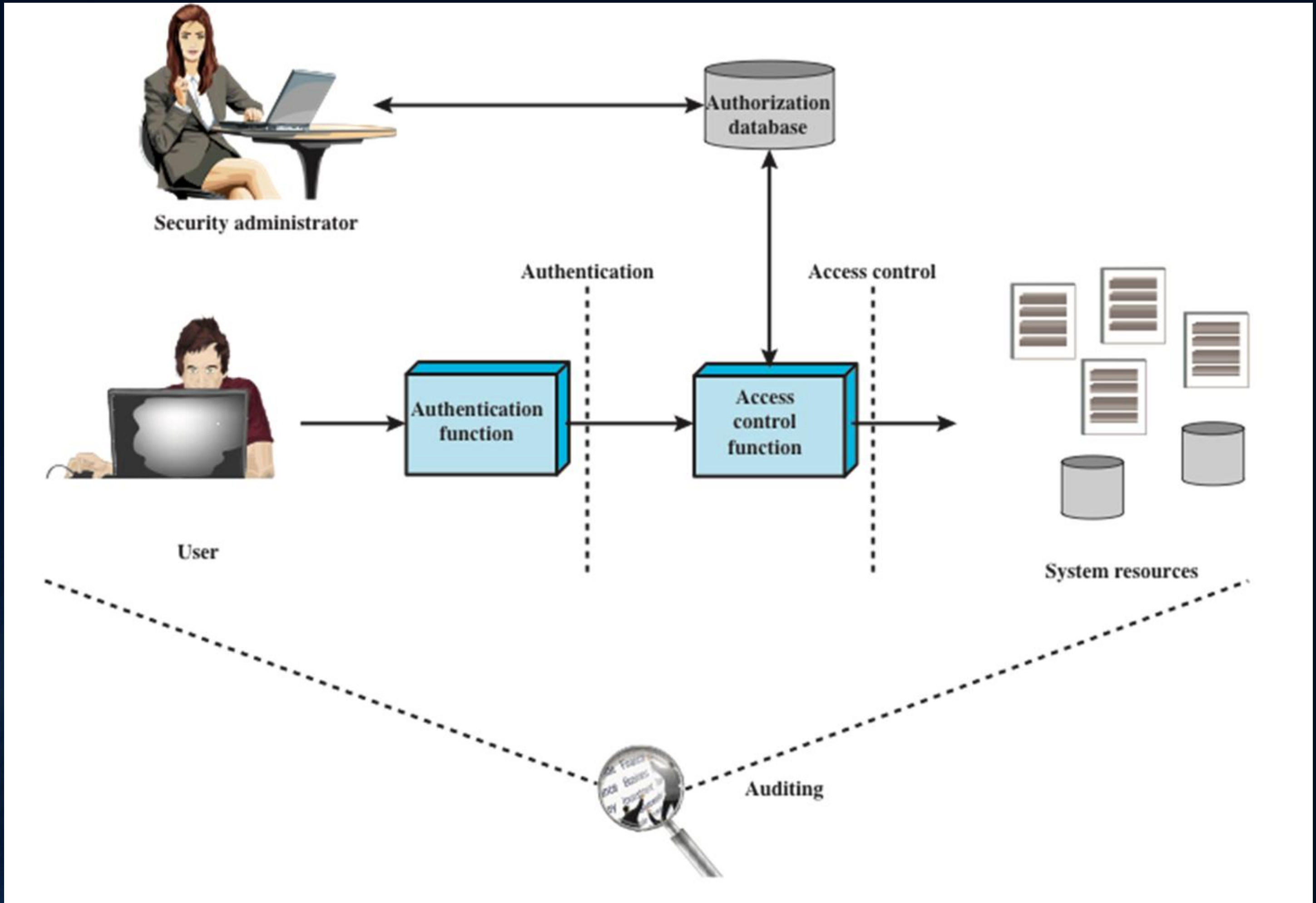
PRINCIPLES
MECHANISMS
MAIN IMPLEMENTATIONS

APM@FEUP

Operations

- RFC 4949 refers access control as
  - specification of who or what may have access to a specific system resource and the type of access that is permitted (e.g., read, write, …)
- Involves
  - Authentication – Identification of users or other entities in the system
  - Authorization – The granting of permission to a system entity to access a resource and how (who is trusted for executing that operation)
  - Auditing – examination of user activities to test and verify policies, access adequacy and security breaches
  - Security administration – Maintenance of the authorization database that specifies what type of access to which resources each user is allowed
  - Operating systems implement some sort of access control function through native components
  - Resources comprehend the file system (directories, files), devices, system databases, other OS managed objects (processes, synchronization, policies, …)

The access control system



Relationship Among Access Control and Other Security Functions

Access control policies

- Authorization function determines the access to resources
  - from authenticated entities (subjects) determines who can access what (objects) and in what way (permissions or access rights to operations)
- Generally, four categories of access control policies can be considered
  - Discretionary access control (DAC)
    - for each protected object, a list of subjects exist (allowed or not) stating what operations can be performed (permissions)
    - It's a complete matrix of permissions with an entry for every combination of subject and object
  - Mandatory access control (MAC)
    - Decides based on the security sensitivity of resources and the security clearance of subjects (military).
    - It is called mandatory because it do not allow any rules to change security levels or clearances
  - Role based access control (RBAC)
    - Accesses are granted based on the role of the subject (there is a mapping between subjects and the roles they have)
  - Attribute based access control (ABAC)
    - Access is granted based on assigned attributes to subjects and objects

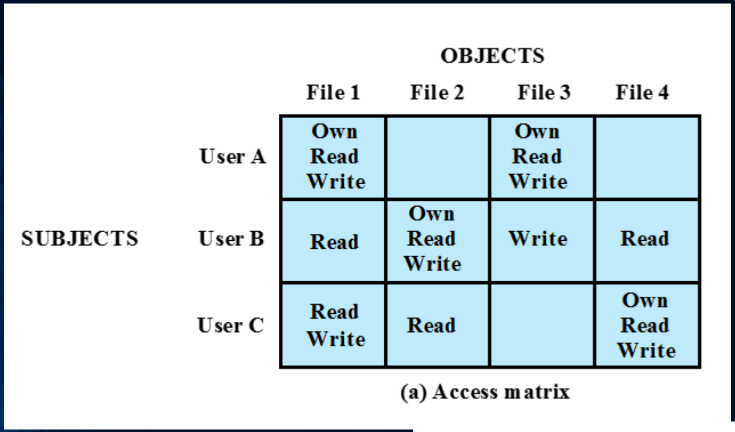
Access control elements

- Subjects
  - Authenticated entity that can access objects
    - Can be a process or thread representing a user or an application duly authenticated
    - They can be divided in classes
      - Owner – usually the creator of the resource or an administrator
        - the owner can grant new access rights to other subjects (in DAC or RBAC)
      - Group – a set of subjects
      - World – all subjects
- Objects
  - Access controlled resources or assets
    - files, directories, data bases, applications, devices, kernel objects (mutexes, semaphores, …), API calls, …
- Access rights
  - Way in which a subject accesses an object (the allowed operations)
    - read, write, execute, delete, create, search, traverse, …

Discretionary access control (DAC)

- Is represented by an access matrix
  - lists subjects in one dimension (rows)
  - lists objects in the other dimension (columns)
  - each cell or entry specify the access rights
- The access matrix is often sparse
  - Empty cells can represent some default (usually only owner allowed)
- The access matrix is often decomposed
  - Column decomposition
    - Each object has a list of accessing (or denying) subjects associated with it
    - It is known as the access control list (ACL)
    - For each subject in the list, his access rights are listed
  - Row decomposition
    - Each subject maintains a list of allowed objects
    - For each object, the list of access rights (capabilities) are recorded

Examples of an access matrix

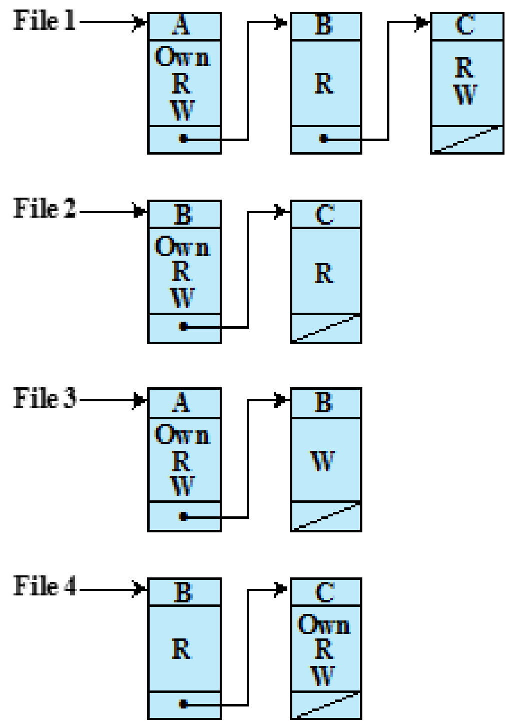


		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

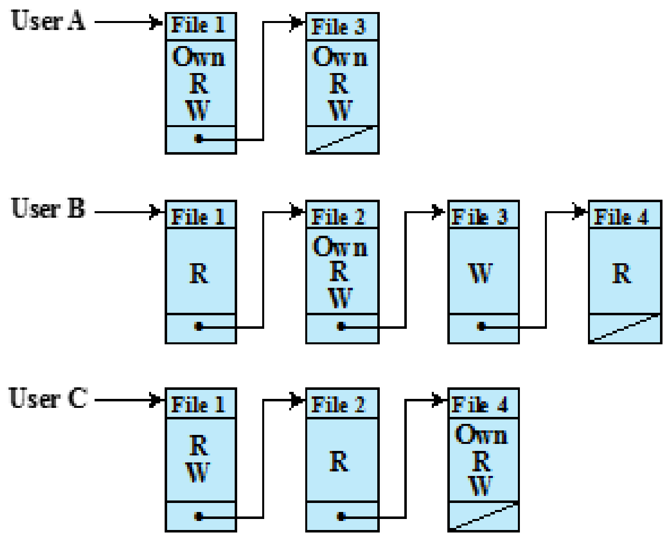
(a) Access matrix

access matrix

decompositions


File 1 → A: Own R W → B: R → C: R W
File 2 → B: Own R W → C: R
File 3 → A: Own R W → B: W
File 4 → B: R → C: Own R W

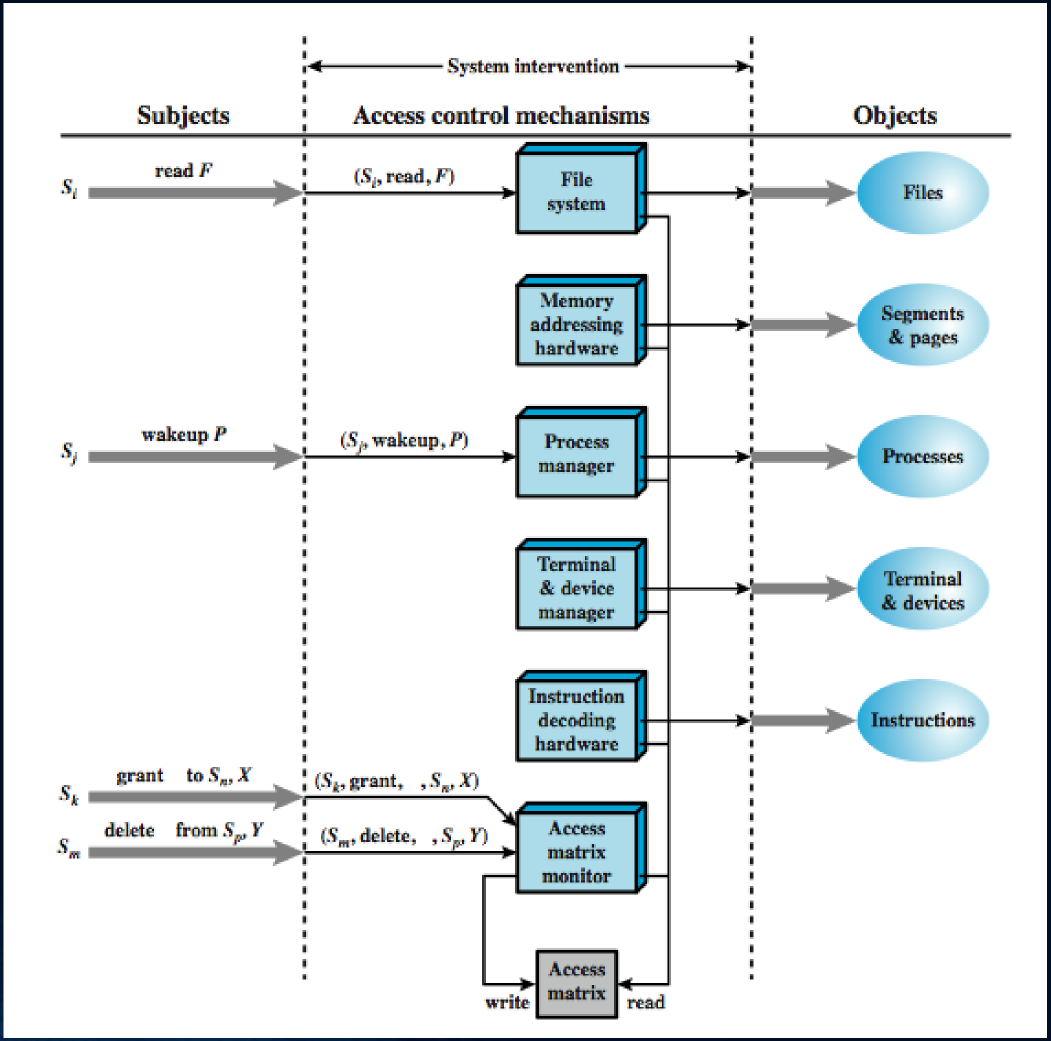
(b) Access control lists for files of part (a)


User A → File 1: Own R W → File 3: Own R W
User B → File 1: R → File 2: Own R W → File 3: W → File 4: R
User C → File 1: R W → File 2: R → File 4: Own R W

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

Access control function

- Specialized OS modules to apply the DAC policies
  - When some subject requests (OS API) an operation on an object
  - A special access matrix manager can modify the matrix entries

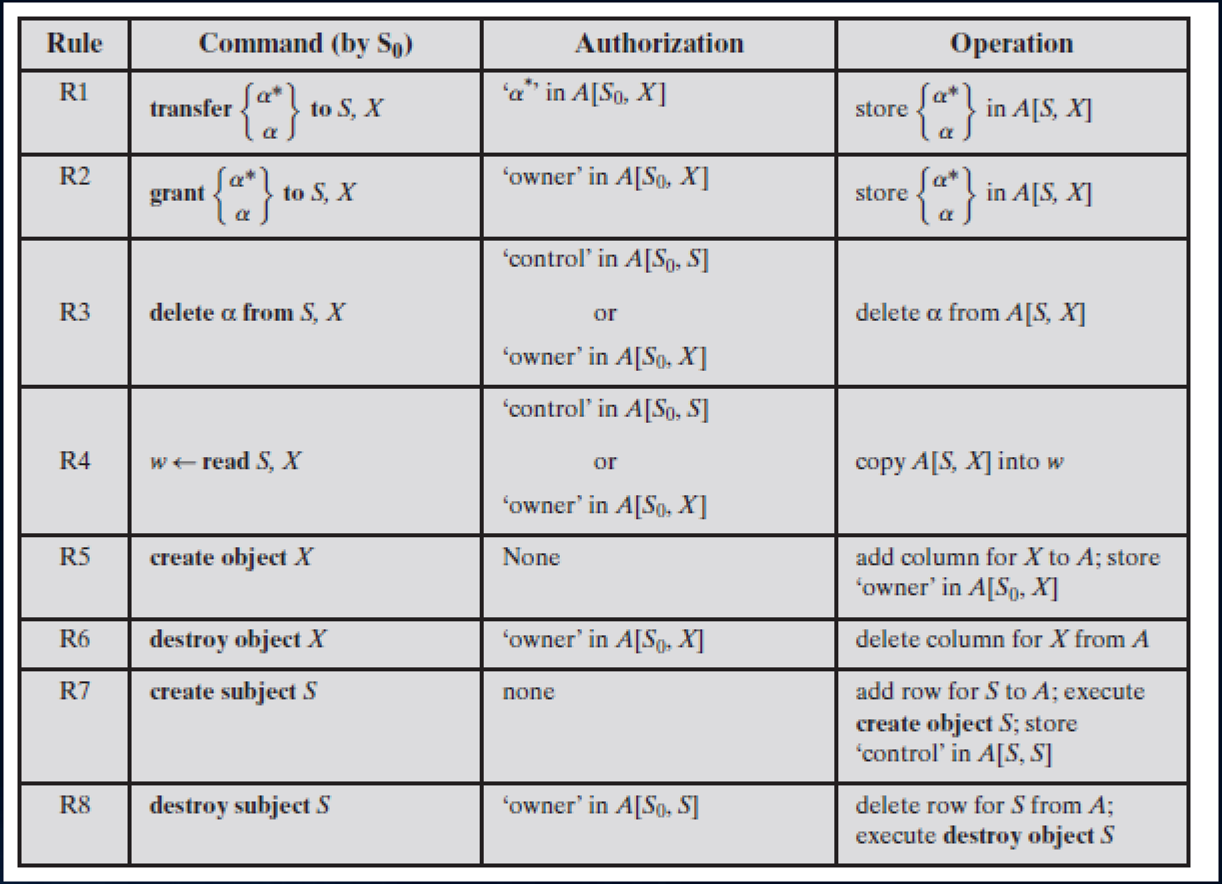


Allowed users can change the access matrix

Rules (policies) for modifying matrix entries

- **The system defines a set of rules for controlling the access control matrix, (the name discretionary come from these rules)**
  - Examples
    - Rules
    - Commands
    - Authorization condition
    - Result

examples issued by subject S_o

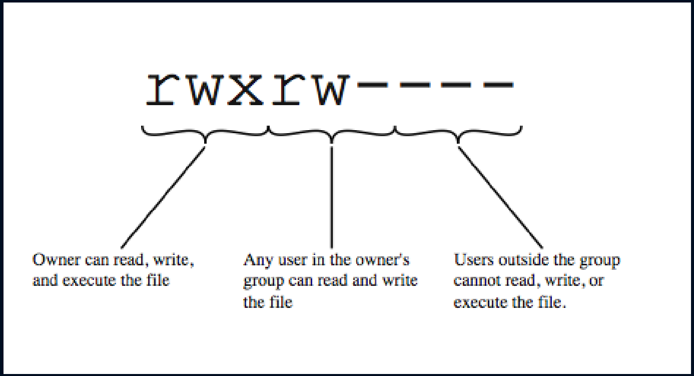


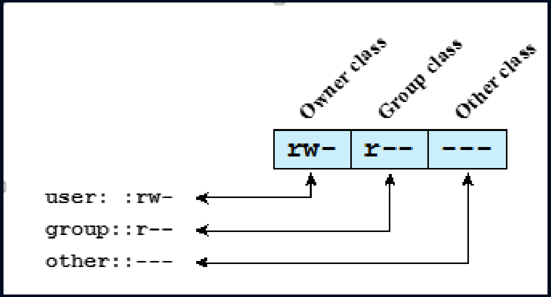
Rule	Command (by S_0)	Authorization	Operation
R1	transfer $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ to S, X	'α^*' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R2	grant $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ to S, X	'owner' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha^*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

α^* - means access right (α) with propagation permission

UNIX object access control

- Each user has a unique identifier (user ID)
- He is also a member of a primary group (has also a group ID)
- 12 protection bits per object
  - 9 of them specify read, write and execute (traverse for directories) permissions
  - owner, members of the group, all others
  - 2 other bits specify SetUID and SetGID
    - The SetUID bit means that when executed, the effective ID of the process is the executable file owner (and group)
    - the SetGID bit, when applied to a directory, makes any new file in it, to have the same group
  - the 12th bit is the sticky bit
    - when applied to a directory means that only the owner of each file inside it, can rename, move or delete that file


rwxrw----
Owner can read, write, and execute the file
Any user in the owner's group can read and write the file
Users outside the group cannot read, write, or execute the file.



User with ID=0 is exempt from any control

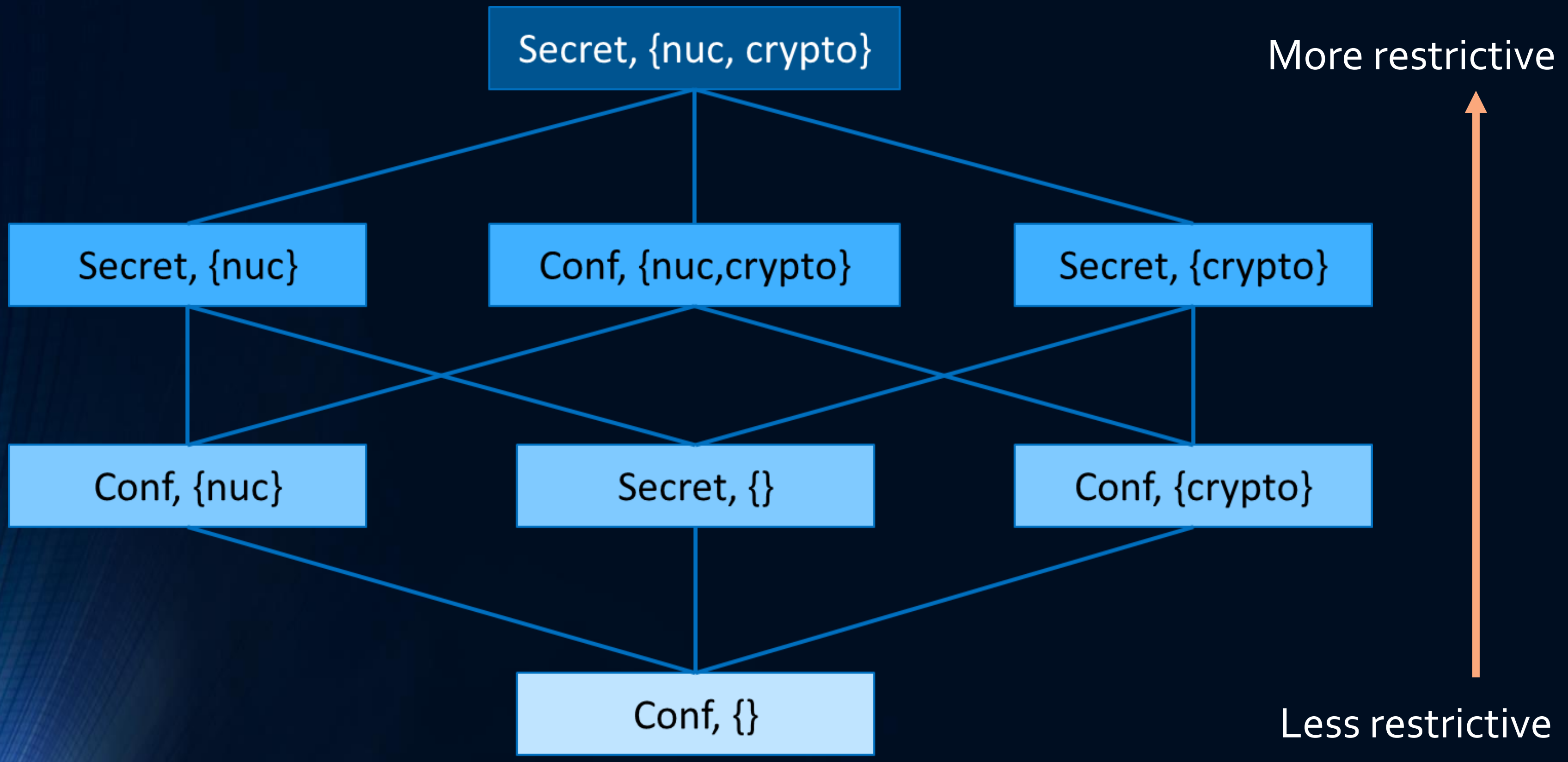
Mandatory access control (MAC)

- There is a central authority that *mandates* policy
- Information (objects) belong to that authority, not to users
- Both subjects and objects are assigned a security label
  - Those labels belong to a well-defined hierarchy
  - Heavily influenced by the US DoD "Orange Book"
    - DoD Trusted Computer System Evaluation Criteria document
- The hierarchical label system is also known as Multi-Level Security (MLS)
  - Usually, the main concern is confidentiality protection
  - The label hierarchy can have several components (dimensions)
    - Two common: confidentiality sensitivity level (e.g., top secret > secret > confidential > unclassified), and
    - Compartments: categories of information
      - (e.g., cryptography, nuclear, biological, satellite reconnaissance)

Hierarchy of labels (levels)

- Label: pair of a sensitivity level and a set of compartments
  - For a subject the sensitivity level is the level of trust of the authority on him, and the compartments are based on the need to know to perform his duty on the system
    - The sensitivity level of a subject is also known as the security clearance level
    - The sensitivity level of a document is its security classification
  - Examples of labels
    - (Top secret, {crypto, nuclear})
    - (Unclassified, { })
  - Labels are imposed by the security authority (organization)
- The hierarchy
  - Notation: $L(X) \sqsubseteq L(Y)$
    - Means L(X) is no more restrictive than L(Y), where X and Y could be subjects or objects
    - Definition: $\boxed{L_1 \sqsubseteq L_2 \text{ iff } S_1 \leq S_2 \wedge C_1 \subseteq C_2}$
      - L is a label, S a sensitivity level, and C a set of compartments

Example of labels

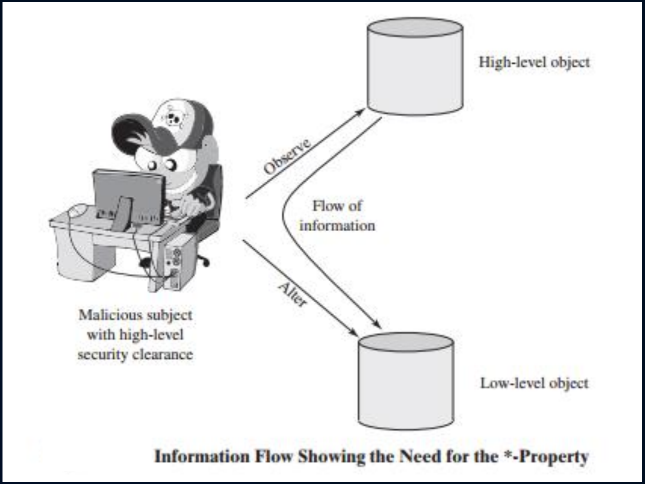


Labels on the same row are ‘incomparable’, not on the same hierarchical level (we cannot establish the ⊑ relationship between those labels)

Here, we can establish that (Conf, {}) ⊑ (Conf, {nuc}) ⊑ (Secret, {nuc}) ⊑ (Secret {nuc, crypto})

Access control in MLS

- **When may a subject read an object ?**
  - Goal is to prevent access to information for which a subject has not clearance
  - Rule (policy): S may read O iff $L(O) \sqsubseteq L(S)$
    - Object classification must be below (or equal to) the subject clearance (label)
    - Also known as "no read up" rule (aka the security property)
- **When may a subject write to an object ?**
  - Subjects must not be allowed to *launder* information, meaning that information on a higher level is conveyed to lower levels
  - Rule (policy): S may write O iff $L(S) \sqsubseteq L(O)$
    - Object classification should be above (or equal to) the subject clearance
    - Also known as "no write down" rule (aka *-property)
  - Write means 'append' or create a new document
  - For edit, read and write permissions are required



Information Flow Showing the Need for the *-Property

Examples

- **Scenario**
  - A Colonel has clearance (Secret, {nuclear, Europe})
  - DocA has classification (Confidential, {nuclear})
  - DocB has classification (Secret, {Europe, US})
  - DocC has classification (Top Secret, {nuclear, Europe})
- **Which documents can the Colonel read ?**
  - Only DocA (DocB includes another compartment, and DocC belongs to a higher level)
- **Which documents can the Colonel write ?**
  - Only DocC (may write (append) but not read)
- **From these 3, there is no document he can edit (read and write)**

Formalization of MLS

- MLS was proposed and formalized in an article in 1973
  - By Bell and LaPadula[1] (Secure Computer Systems: Mathematical foundations)
  - It was formalized as a mathematical model, within the restrictions and assumptions stated on that paper
  - A proof of confidentiality was demonstrated (demonstrating no leak to subjects without the needed clearance)
  - Influenced strongly the Orange Book of the US DoD
    - The "no read up" (security property) and "no write down" (*-property) are the basis
- SELinux (Security Enhanced Linux) is a kernel extension from 2000 (merged in some distributions after Linux 2.6)
  - Separates security policy from access control engine
  - Supports MAC with MLS enabled
  - Subject and objects are assigned a 'context' (L( ))
  - MLS rules are enforced for those subjects and objects

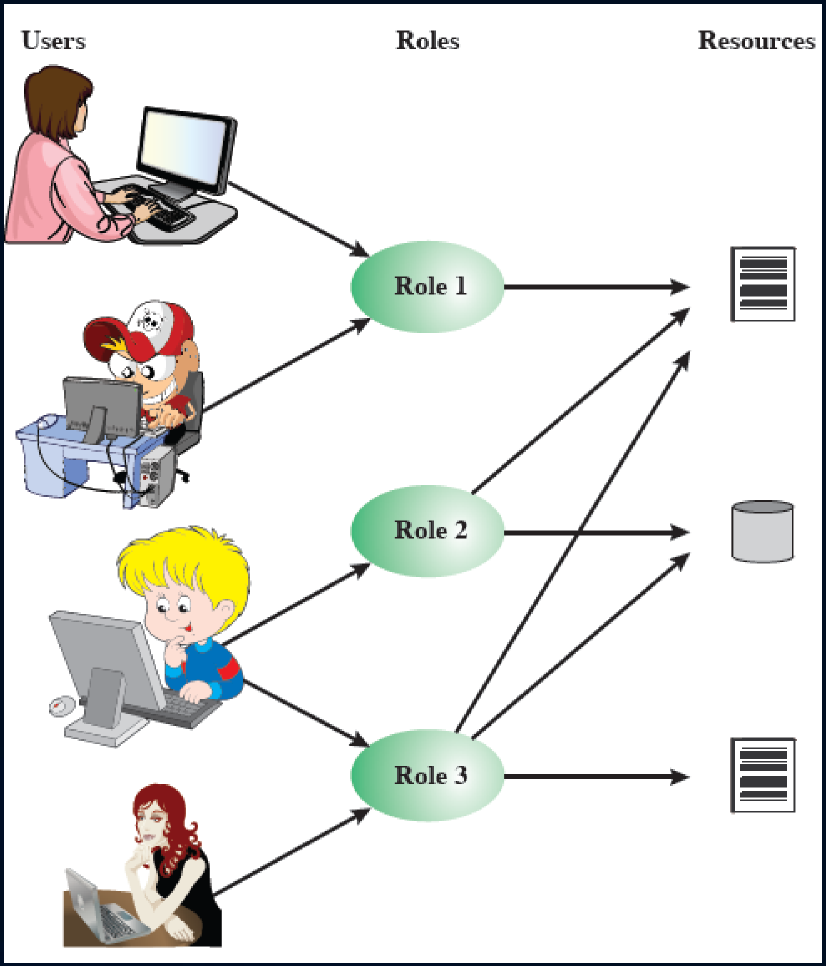
[1] revisited in 2005 in another publication by the authors (Looking back at the Bell-LaPadula model)

Beyond MLS (or extensions of Bell-Lapadula)

- **Biba Model**
  - Is the application of Bell-LaPadula to integrity instead of confidentiality
  - Was adapted to integrity by Biba in 1977
    - Same rules, different lattice (definition of labels and hierarchy)
    - Is a dual lattice (high vs. low is flipped)
    - Biba: Low integrity sources may not flow to high integrity sinks
- **Mandatory access control can be defined in different forms (not MLS)**
  - Brewer-Nash model, also known as the Chinese Wall model
    - Developed to solve conflicts of interest inside organizations
      - Consulting companies with concurrent products or enterprises
      - Law firms involved in conflicting litigations
  - Clark-Wilson model, developed for business information systems, and formulated in 1987
    - E.g., regulating transactions and other state changes with model integrity rules

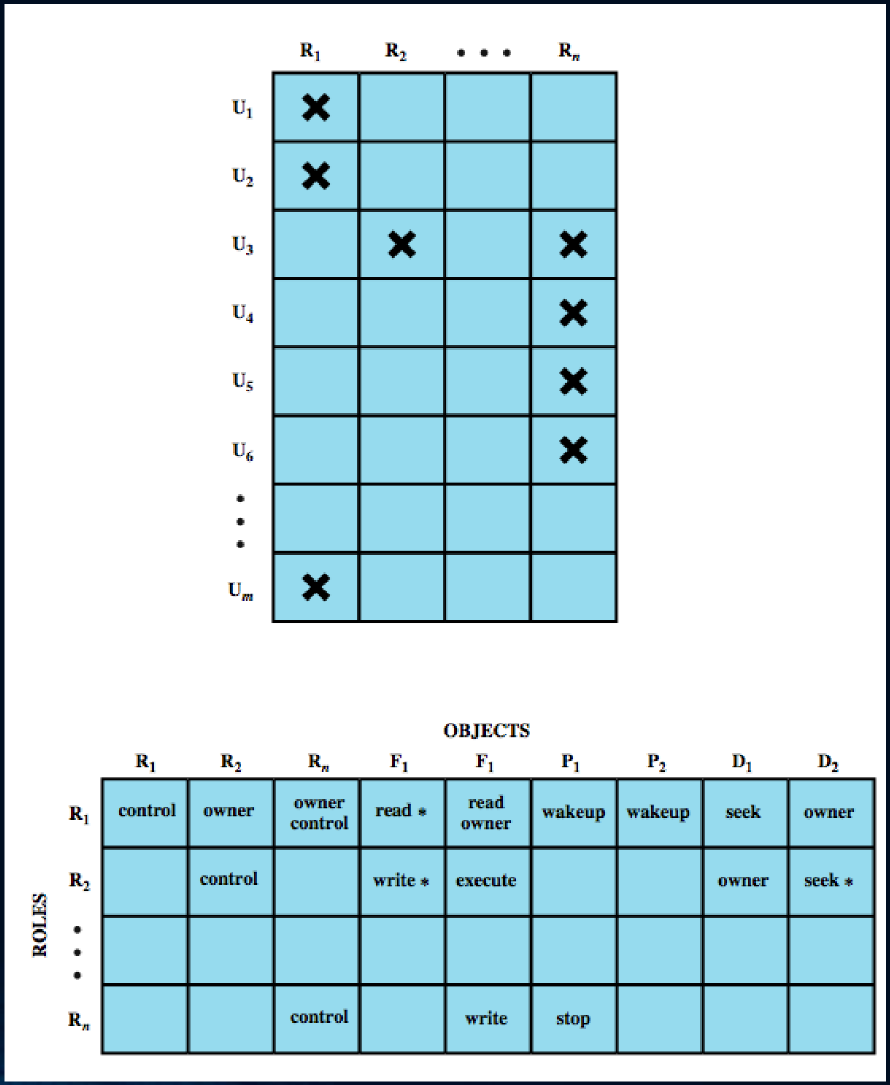
Role based access control (RBAC)

- Access based on role, not identities
- A many-to-many relationship between users and roles is established and stored
- Roles are often static
  - Can also be defined as a mapping between sets of users (groups) and roles



Mappings and permissions

- Relationships between users, roles and access rights are matrices
- Two of them are needed
- Execution control
  - Sometimes RBAC is also used for controlling the execution of certain methods or functionalities in applications
  - This control can be checked and enforced programma-tically
    - E.g., is the process owner in this role ?



	R_1	R_2	• • •	R_n
U_1	✗			
U_2	✗			
U_3		✗		✗
U_4				✗
U_5				✗
U_6				✗
⋮				
U_m	✗			

OBJECTS

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

Role constraints

- Some implementations allow the definition of role constraints
- Some examples
  - Mutually exclusive
    - role sets where users can only be assigned to one of the roles in the set
    - An individual permission type can only be granted to one of the roles in the set
  - Cardinality
    - Set a maximum number of users within a role
      - e.g., the role of 'department president'
  - Prerequisites
    - A user can be assigned to a role only if that user already has been assigned to some other specific role

Formalizing RBAC as an MLS

- **RBAC can be implemented as an MLS, as described in**
  - **Osborne, Sandhu, Munawer, Configuring RBAC to Enforce MAC and DAC policies, ACM, 2000**

Constraint on Users (security clearance)

$\forall u \in U[\mathrm{L}(u) \text{ is given}]$

Constraint on Permissions (security classification)

$P = \{(o,r),(o,w) \mid o \text{ is an object in the system}\};$

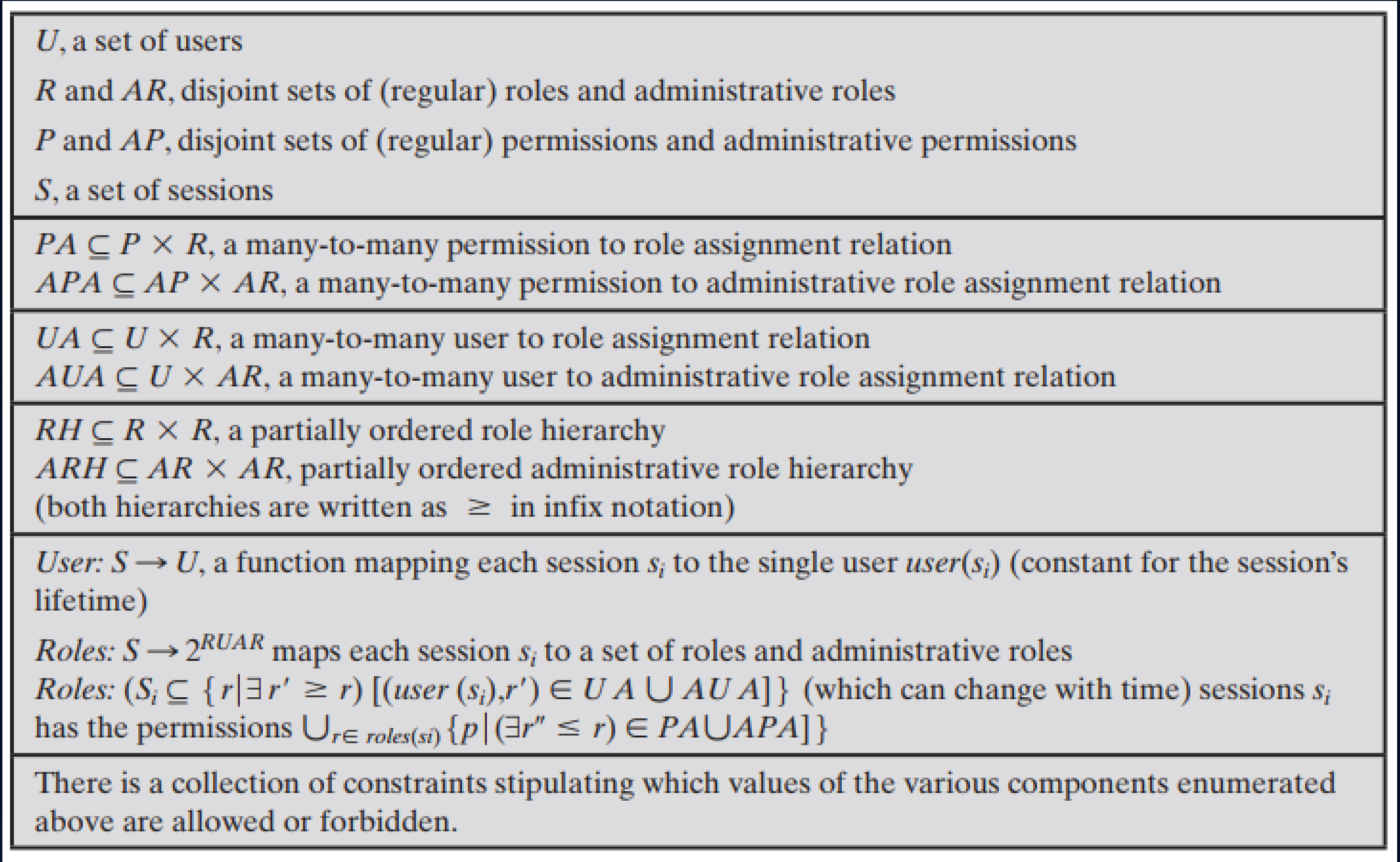
$\forall o \in P[\mathrm{L}(o) \text{ is given}]$

Constraint on Role Assignment (UA)

$\forall r \in UA\ [\text{w-level}(r) \text{ is defined}];$

$\forall (u,r) \in UA\ [\mathrm{L}(u) \geq \text{r-level}(r)];$

$\forall (u,r) \in UA\ [\mathrm{L}(u) \leq \text{w-level}(r)]$



U, a set of users

R and AR, disjoint sets of (regular) roles and administrative roles

P and AP, disjoint sets of (regular) permissions and administrative permissions

S, a set of sessions

$PA \subseteq P \times R$, a many-to-many permission to role assignment relation
$APA \subseteq AP \times AR$, a many-to-many permission to administrative role assignment relation

$UA \subseteq U \times R$, a many-to-many user to role assignment relation
$AUA \subseteq U \times AR$, a many-to-many user to administrative role assignment relation

$RH \subseteq R \times R$, a partially ordered role hierarchy
$ARH \subseteq AR \times AR$, partially ordered administrative role hierarchy
(both hierarchies are written as $\geq$ in infix notation)

User: $S \rightarrow U$, a function mapping each session s_i to the single user $user(s_i)$ (constant for the session's lifetime)

Roles: $S \rightarrow 2^{RUAR}$ maps each session s_i to a set of roles and administrative roles
Roles: $(S_i \subseteq \{r \mid \exists r' \geq r)\ [(user\ (s_i), r') \in UA \cup AUA]\}$ (which can change with time) sessions s_i has the permissions $\bigcup_{r \in roles(si)} \{p \mid (\exists r'' \leq r) \in PA \cup APA]\}$

There is a collection of constraints stipulating which values of the various components enumerated above are allowed or forbidden.

There are other works, demonstrating how MLS can be employed in many Access Control situations, like: Bertino, Jajodia, Samarati, Database Security: Res. and Practice, *in Inf. Systems*, 1995

Attribute based access control (ABAC)

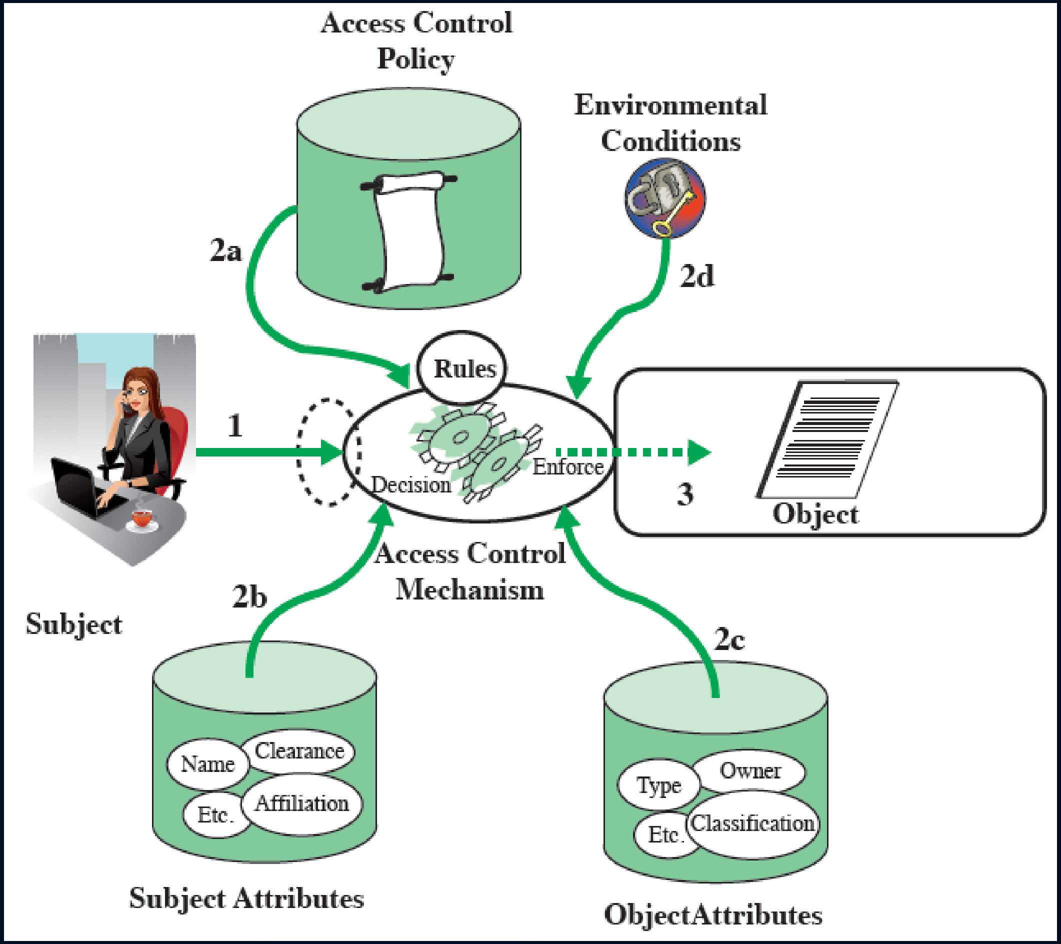
- Recent model for authorization
- Authorizations are conditions on properties of the subjects and the resources (objects)
  - Each resource has the attribute specifying the subject that created it
  - A rule can state ownership privileges for the creators
- It has as strength, flexibility and expressive power
- It is being considered for application in cloud services
- Types of attributes
  - Subject attributes
  - Object attributes
  - Environment attributes

Attributes

- **Subjects**
  - Active entities
  - Change system state or cause information to flow
  - Attributes define the identity and characteristics of each subject
    - Ex: name, organization, job, role, affiliations, clearance, etc.
- **Objects**
  - Passive entities
  - Contain or receive information
  - Attributes contain characteristics that leverage access control decisions
    - Ex: title, author, owner, date, type, classification, etc.
- **Environment**
  - Context in which the attempt to access objects occurs
    - Ex: current date, current time, network traffic, virus activity, etc.
    - Attributes not associated with a resource or subject

Example of an ABAC system

- 1. Access request (with operation)
- 2. Access control is governed by rules taking into consideration the attributes of subject, object and environment
- 3. Access control grants (or not) authorization to perform the operation


Access Control Policy
Environmental Conditions
2a
2d
Rules
1
Decision
Enforce
3
Object
Access Control Mechanism
2b
Subject
2c
Name
Clearance
Etc.
Affiliation
Type
Owner
Etc.
Classification
Subject Attributes
ObjectAttributes