Dr. Ammar Haider
Assistant Professor
School of Computing

CS3002 Information Security

Access Control

Reference: Stallings SPP chap 4

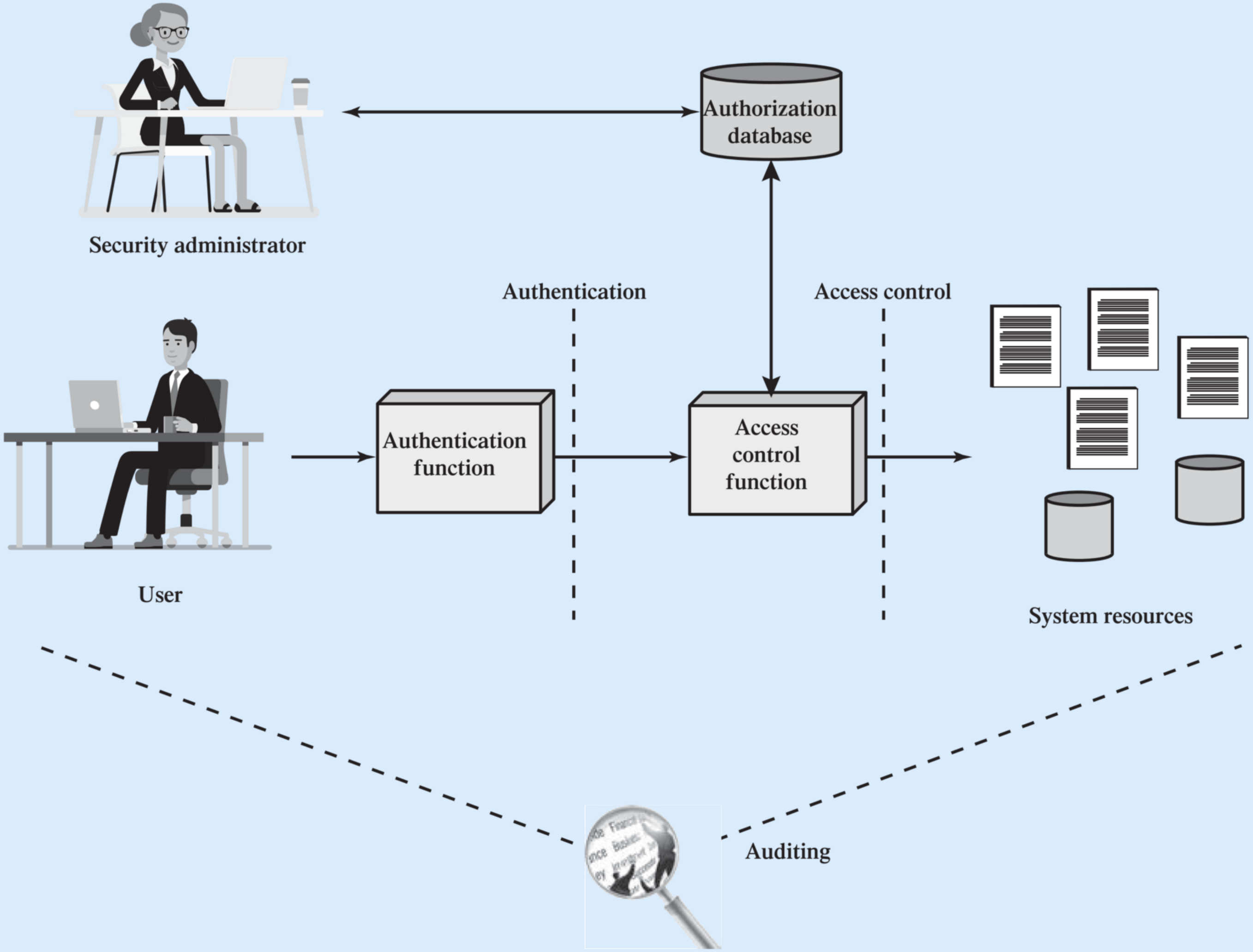
Access Control

Access control implements a security policy that specifies who may have access to each specific system resource, and the type of access that is permitted in each instance.

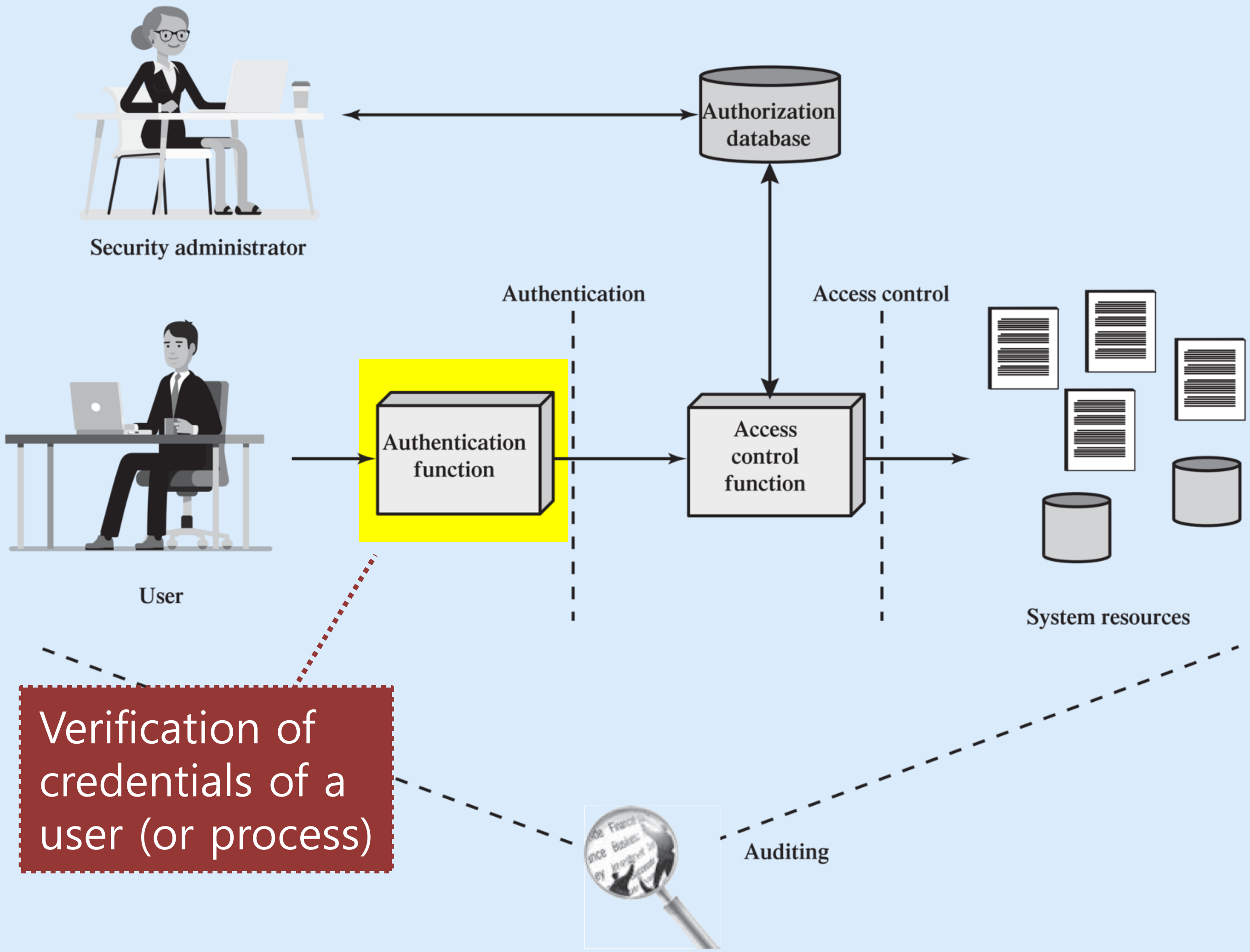
It is the process of ensuring authenticated users have access to the resources they are authorized to use and don’t have access to any other resources.

In a broad sense, all of computer security is concerned with access control.

Context of Access Control

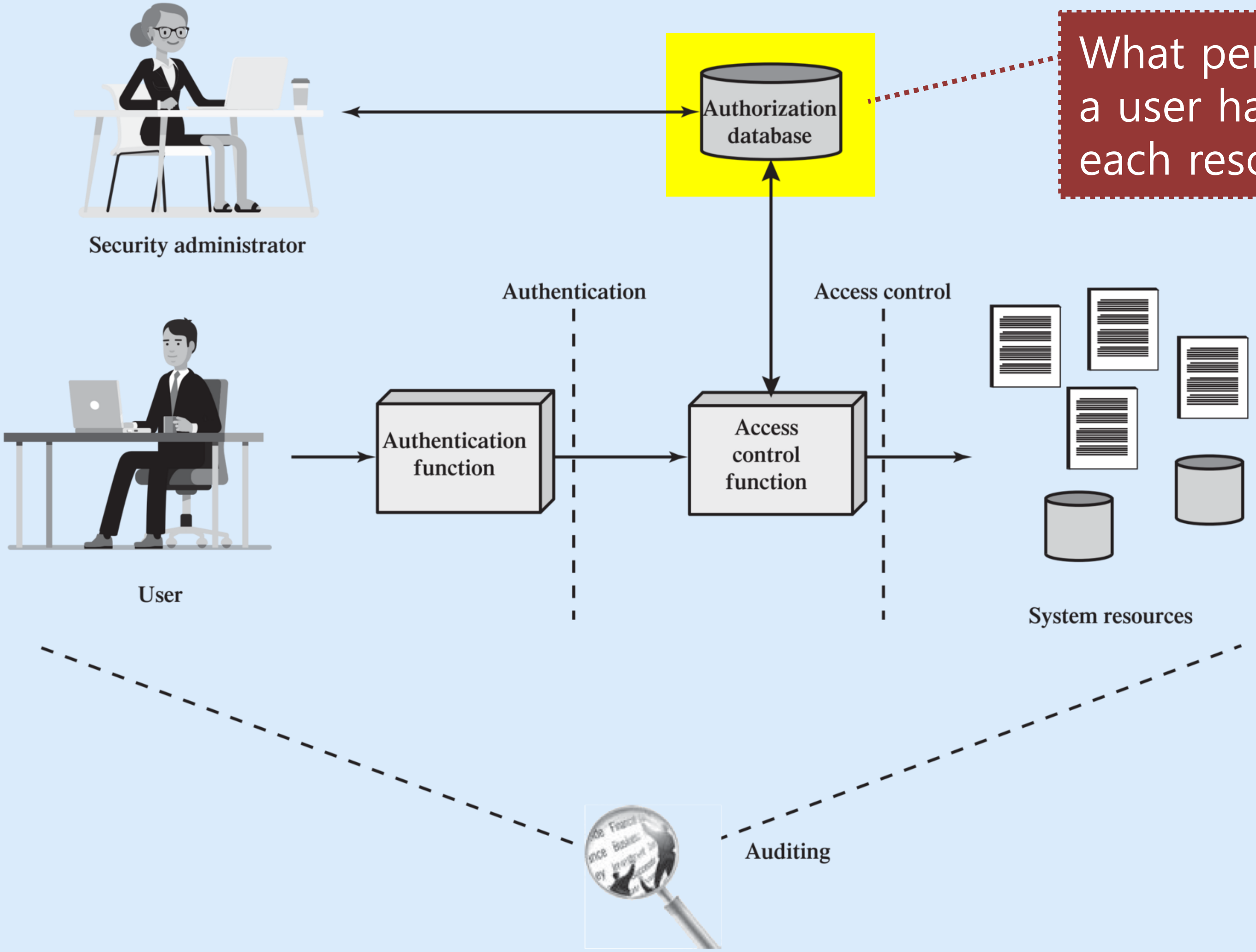


Context of Access Control



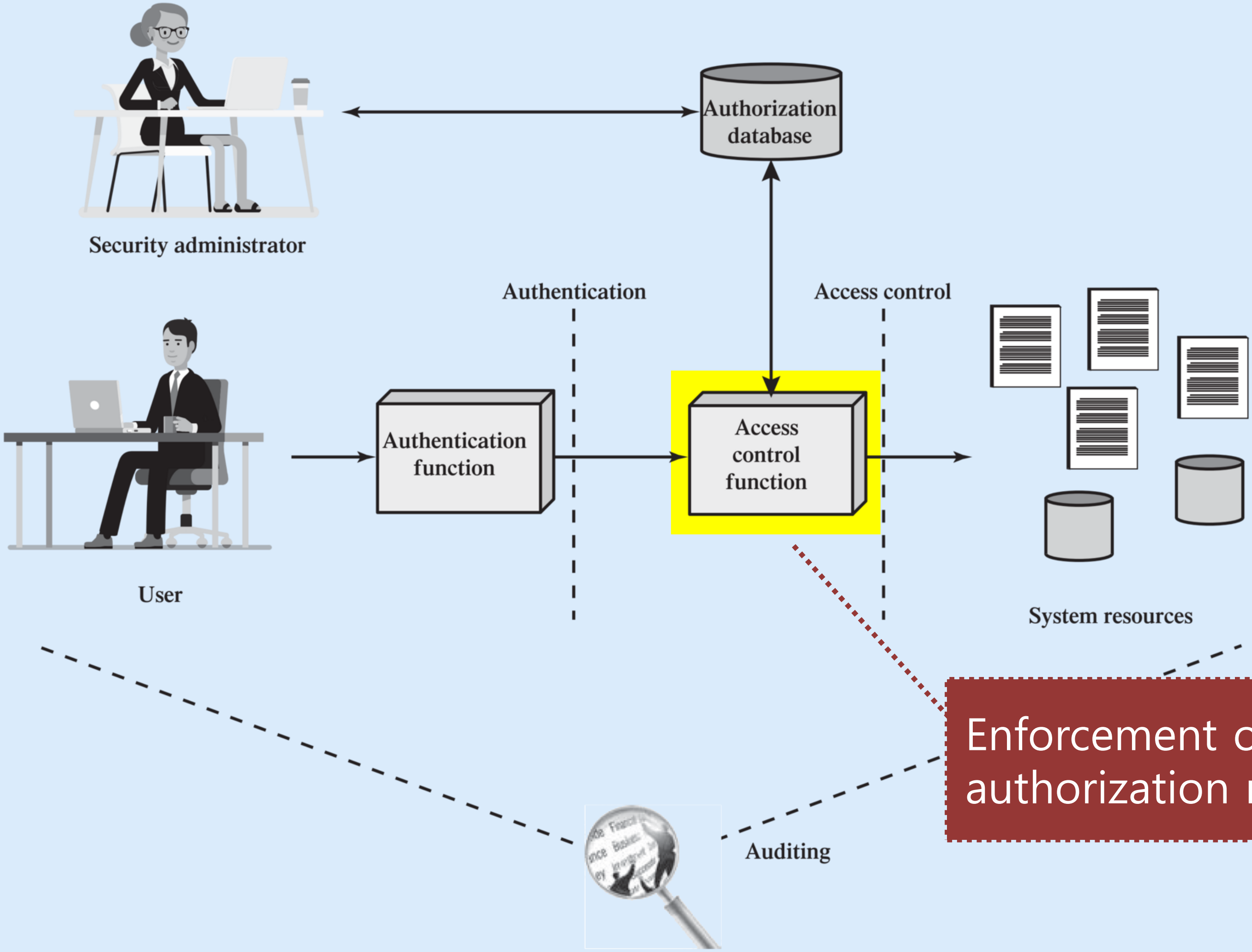
Verification of credentials of a user (or process)

Context of Access Control



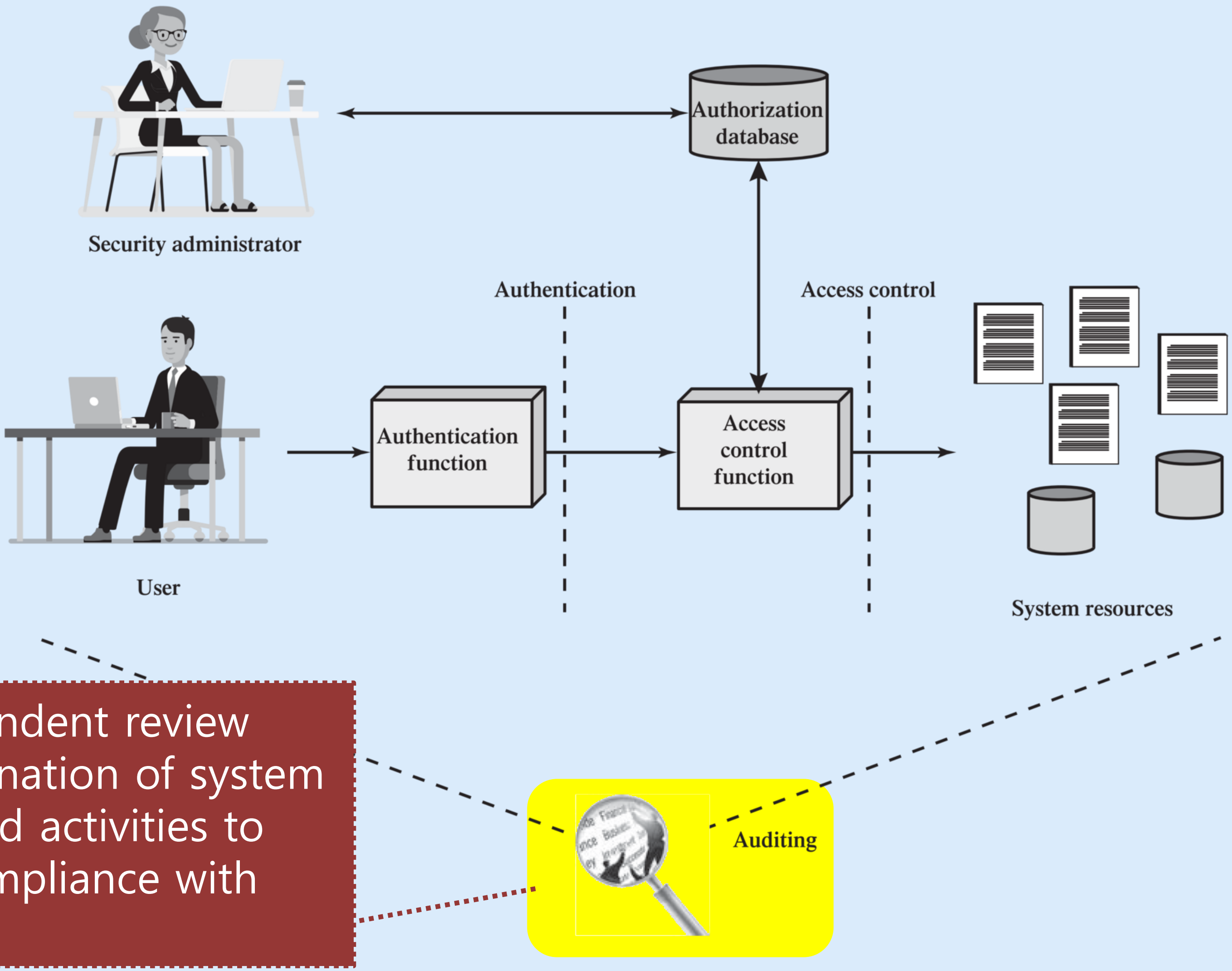
What permissions a user has for each resource

Context of Access Control



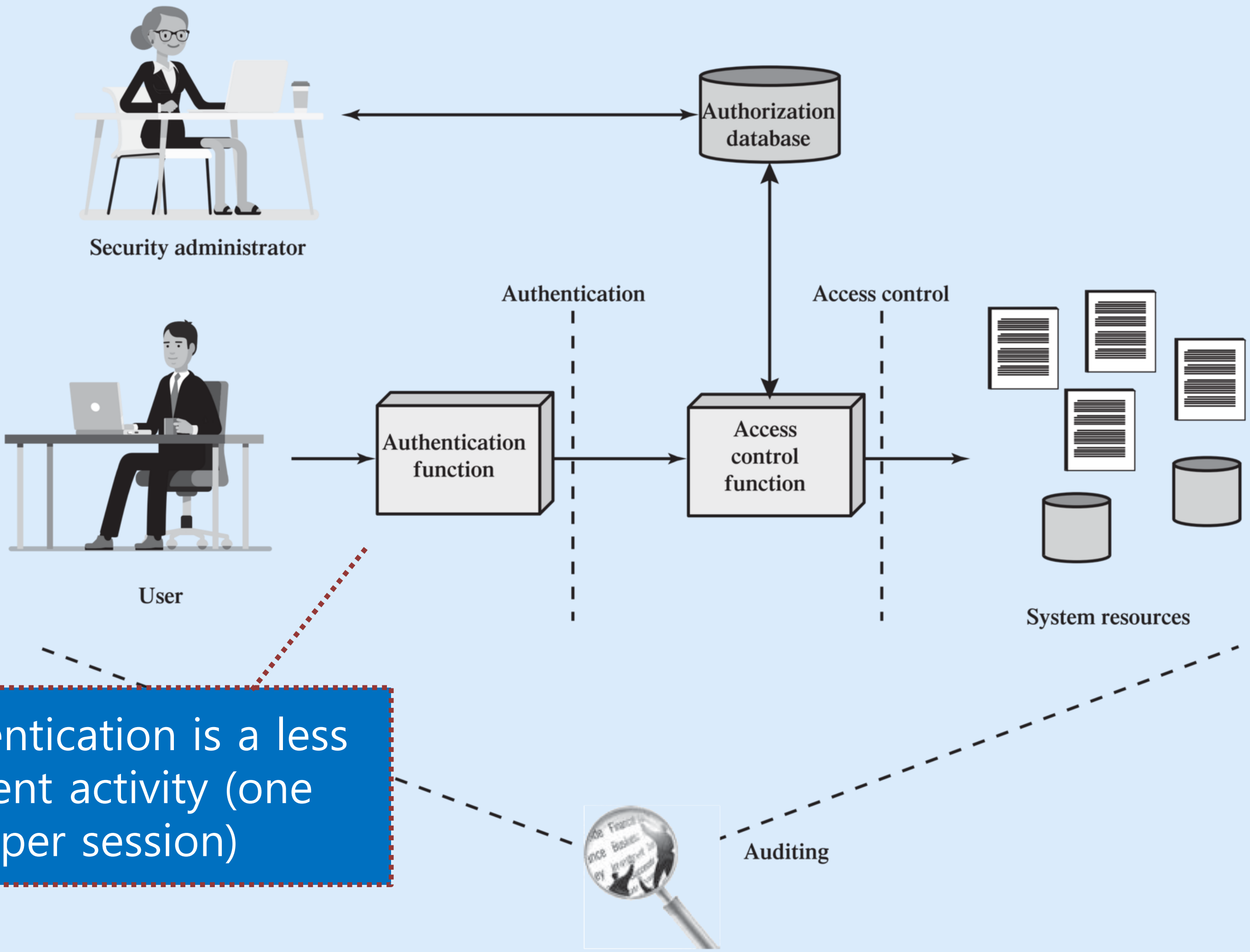
Enforcement of all authorization rules

Context of Access Control



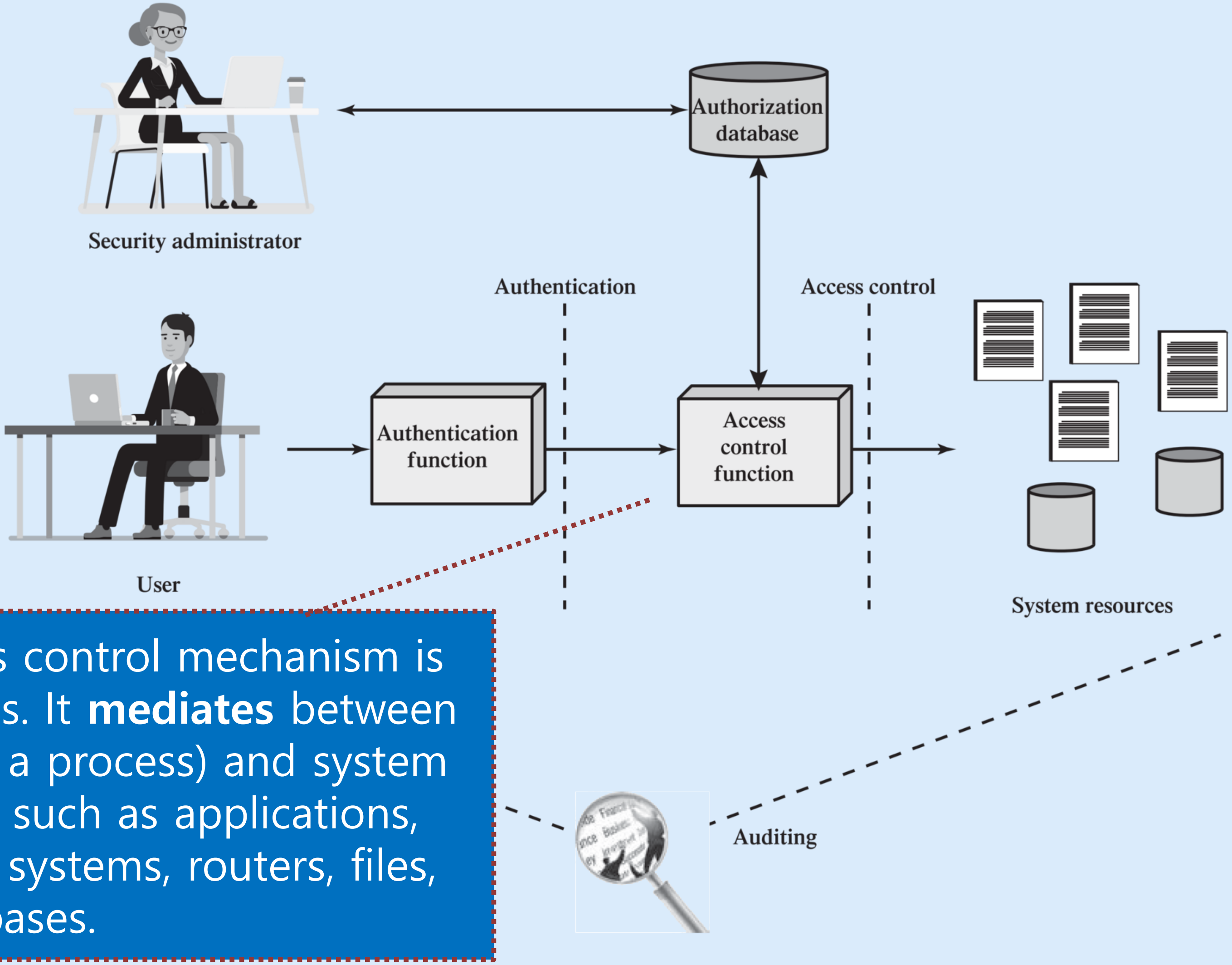
An independent review and examination of system records and activities to ensure compliance with the policy.

Context of Access Control



Authentication is a less frequent activity (one login per session)

Context of Access Control



But access control mechanism is continuous. It **mediates** between a user (or a process) and system resources, such as applications, operating systems, routers, files, and databases.

Terminology: Subject

Subject is an entity capable of accessing objects.

- Three classes
  - Owner
  - Group
  - World
- A user might be **owner** of a resource, or might be included in the **group** who has higher access to resource. If neither is true, that user is said to be included in (rest of the) **world**.

Terminology: Object

An object is a resource to which access is controlled. Examples include:

- Records
- Files, portion of files
- Directories, directory-trees
- Messages
- Programs
- Processors
- Communication ports
- Network nodes
- Memory segments
- Bits, bytes, words

The number and types of objects to be protected depends on the system environment and the desired tradeoff between security on one hand, and complexity, processing burden, and ease of use on the other hand.

Terminology: Access Right

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

- Could include:
  - Read: view, copy, print
  - Write: add, modify, delete. Read access is included!
  - Execute
  - Delete
  - Create
  - Search

Access Control Models

An access control model (i.e. methodology) dictates what types of access are permitted, under what circumstances, and by whom.

Some well-known ones are:

- **Discretionary** Access Control (DAC)
- **Non-discretionary** (centralized administration)
  - Mandatory Access Control (MAC)
  - Role-Based Access Control (RBAC)
  - Rule-based access control
  - Attribute-Based Access Control (ABAC)
  - Risk-based access control

Discretionary Access Control (DAC)

- Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do
- An entity might have access rights that permit it to **grant another entity** access to some resource. That's why this policy is called *discretionary*.
- In most implementations, granting access rights is at the discretion of the **owner**.

DAC: Access Control Matrix

In general, all authorization rules of a system can be put together in a matrix format

Objects

Subjects

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

DAC: Access Control Matrix

In general, all authorization rules of a system can be put together in a matrix format

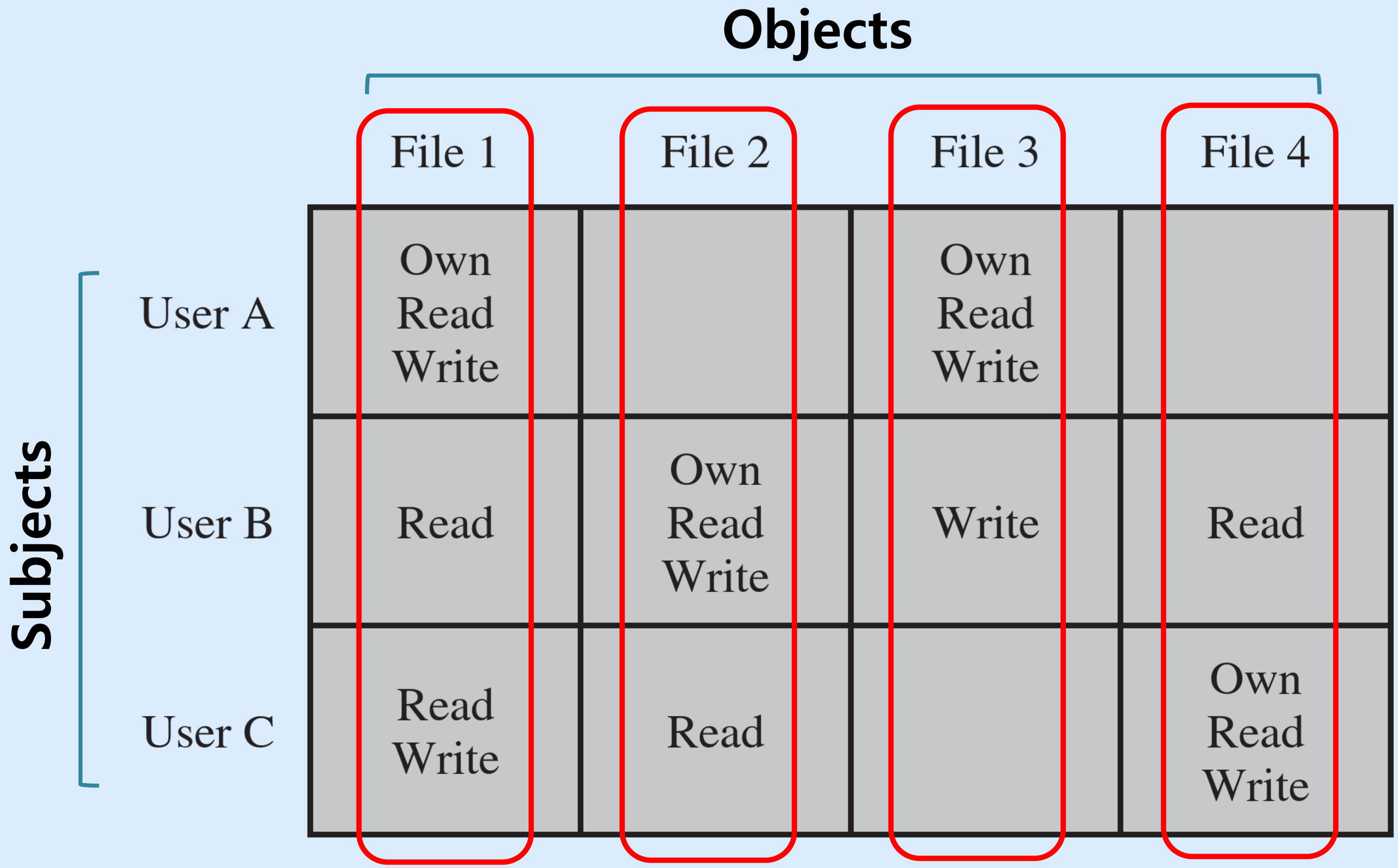
Here, subjects could be individual users or groups of users

Objects

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

Access Control Lists (ACL)

Columns in the access matrix represent ACL for each object: a list of users with their permitted access rights.



Objects

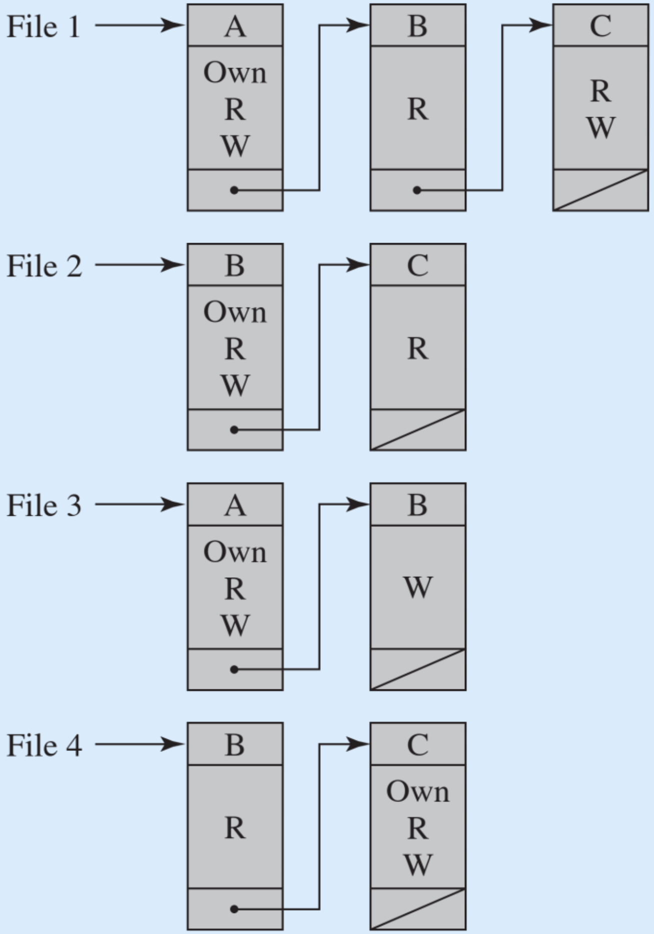
Subjects

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

Access Control Lists (ACL)

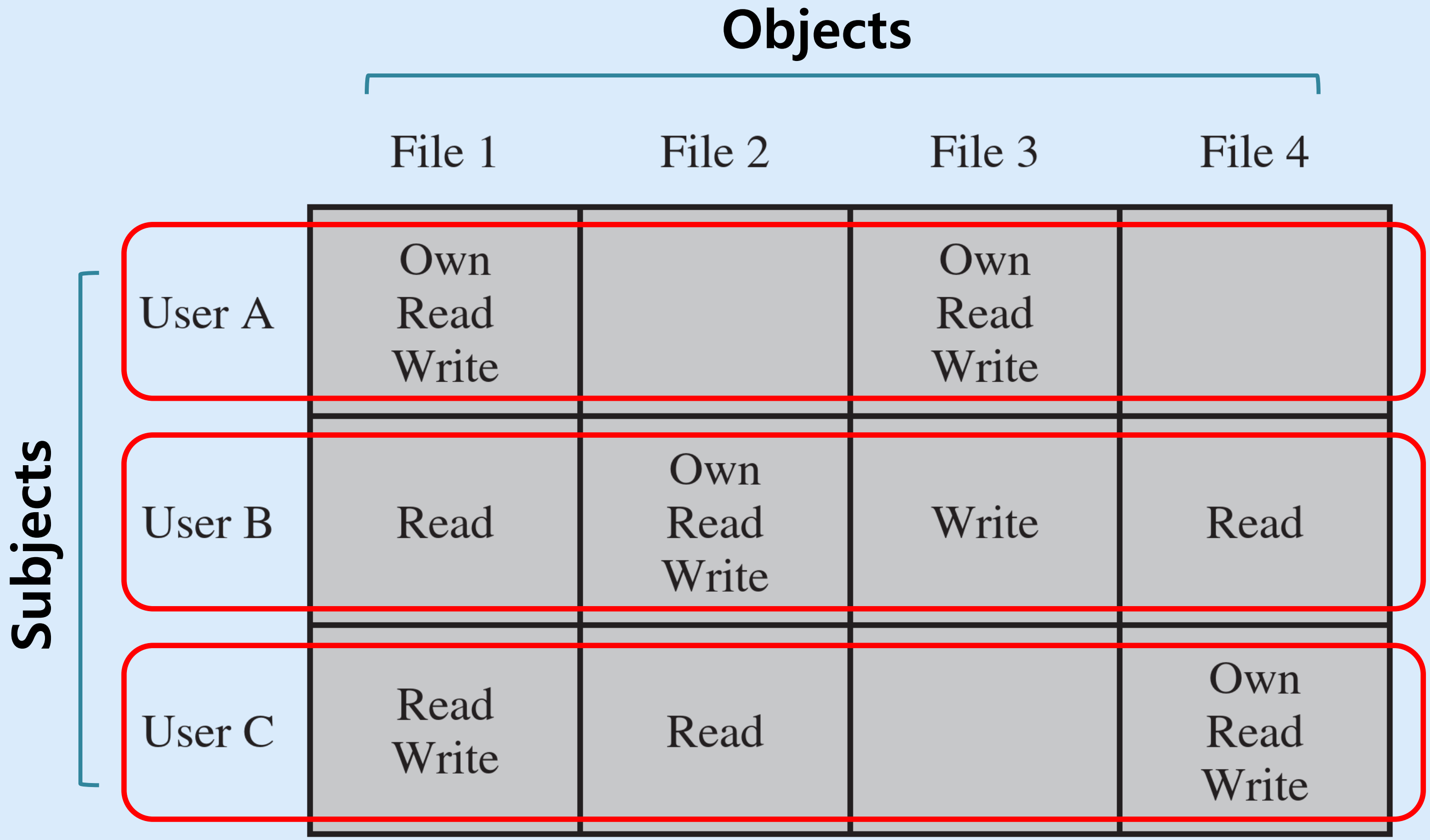
ACLs can be stored as a linked list (arrays would be very sparse).

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



Capability Tickets

Rows in the access matrix represent the capability tickets of users. A user’s ticket specifies their authorized objects and operations.



Objects

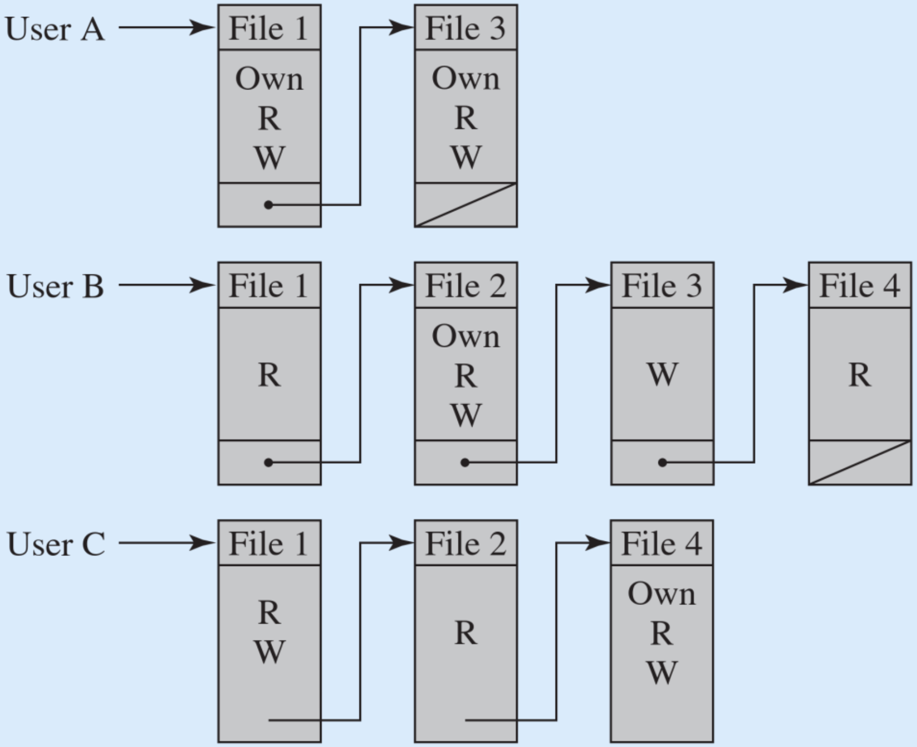
Subjects

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

Capability Tickets

Tickets can also be maintained as linked lists.

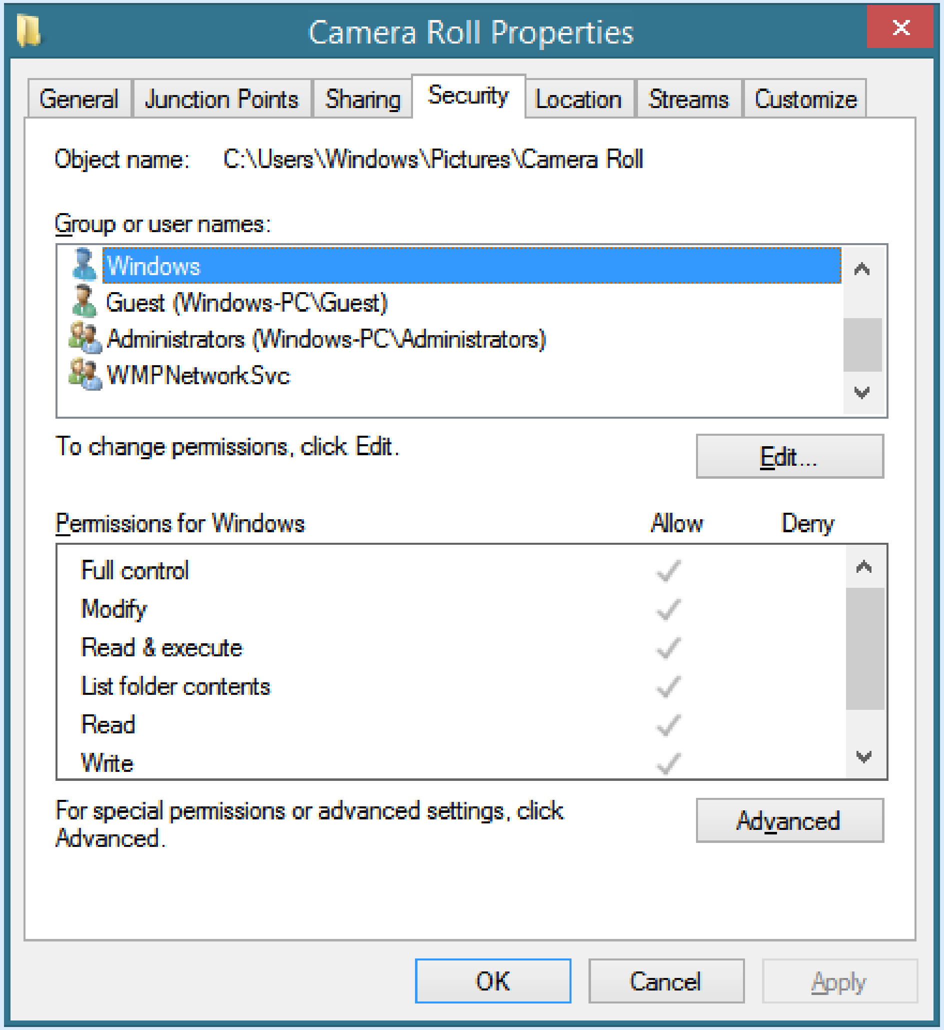
	File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write



Capability Tickets

- The integrity of the ticket must be protected and guaranteed.
  - In particular, the ticket must be unforgeable.
- One way to accomplish this is to have the OS hold all tickets on behalf of users, storing them in a protected region of memory.
- Alternatively, OS can include an unforgeable token in the ticket, such as a message authentication code (MAC).
  - Such tickets can be handed over to users – they present it whenever accessing a resource
  - MAC value is verified by the resource when access is requested.

Example: Windows File/Folder ACLs



Object name: C:\Users\Windows\Pictures\Camera Roll

Group or user names:

- Windows
- Guest (Windows-PC\Guest)
- Administrators (Windows-PC\Administrators)
- WMPNetworkSvc

To change permissions, click Edit.

Permissions for Windows	Allow	Deny
Full control	✓	
Modify	✓	
Read & execute	✓	
List folder contents	✓	
Read	✓	
Write	✓	

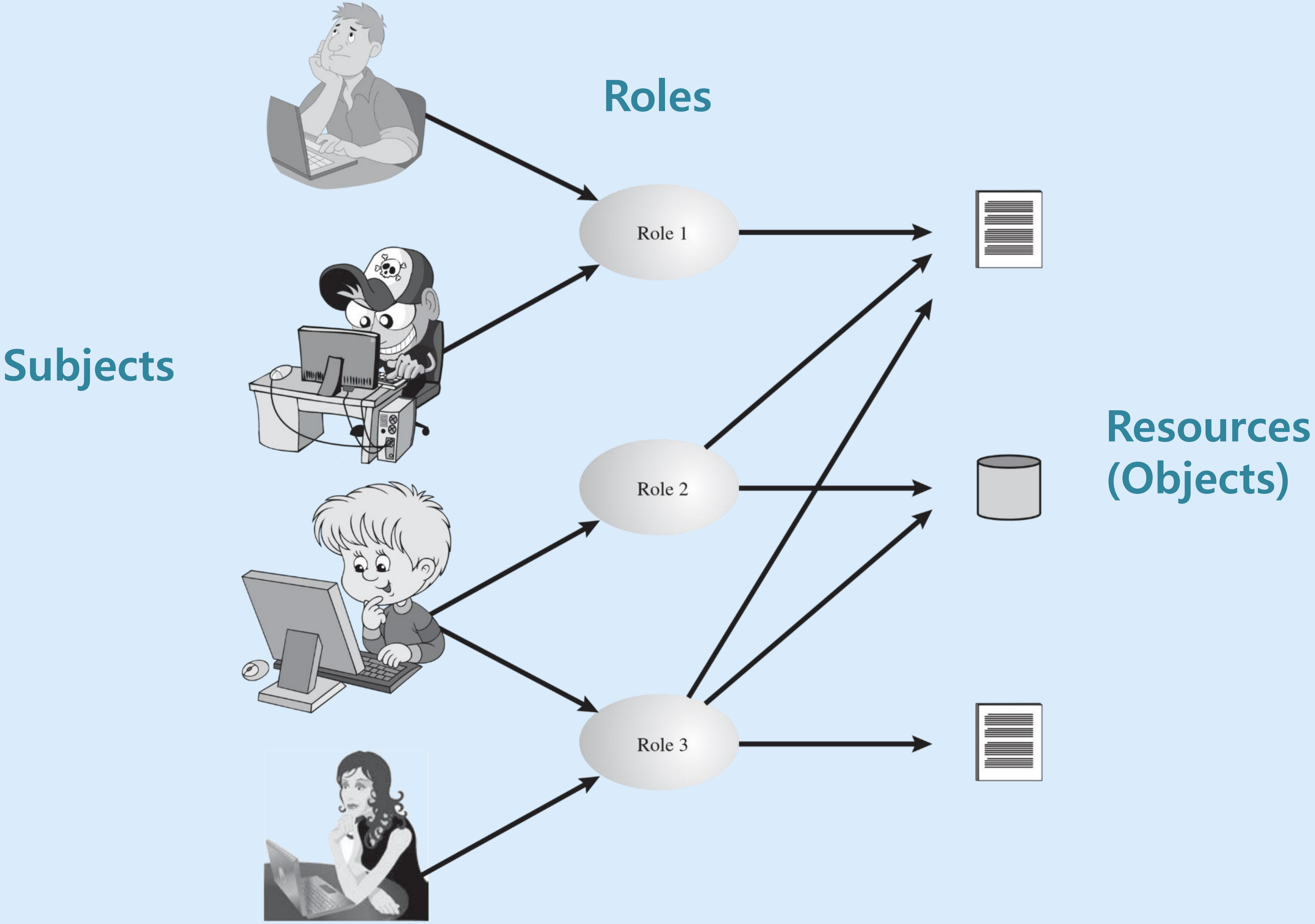
For special permissions or advanced settings, click Advanced.

https://winaero.com/easier-way-to-set-copy-and-manage-ntfs-permissions-acls-on-files-and-folders-in-windows/

Role Based Access Control (RBAC)

- RBAC is based on the roles that users assume in a system rather than the user's identity.
  - e.g. project manager, software engineer, team leader, surveyor, IT technician, cleaner
- It assigns access rights to roles instead of individual users.
- In turn, users are assigned to different roles, either statically or dynamically, according to their responsibilities.

Role Based Access Control



Roles

Subjects

Role 1

Role 2

Role 3

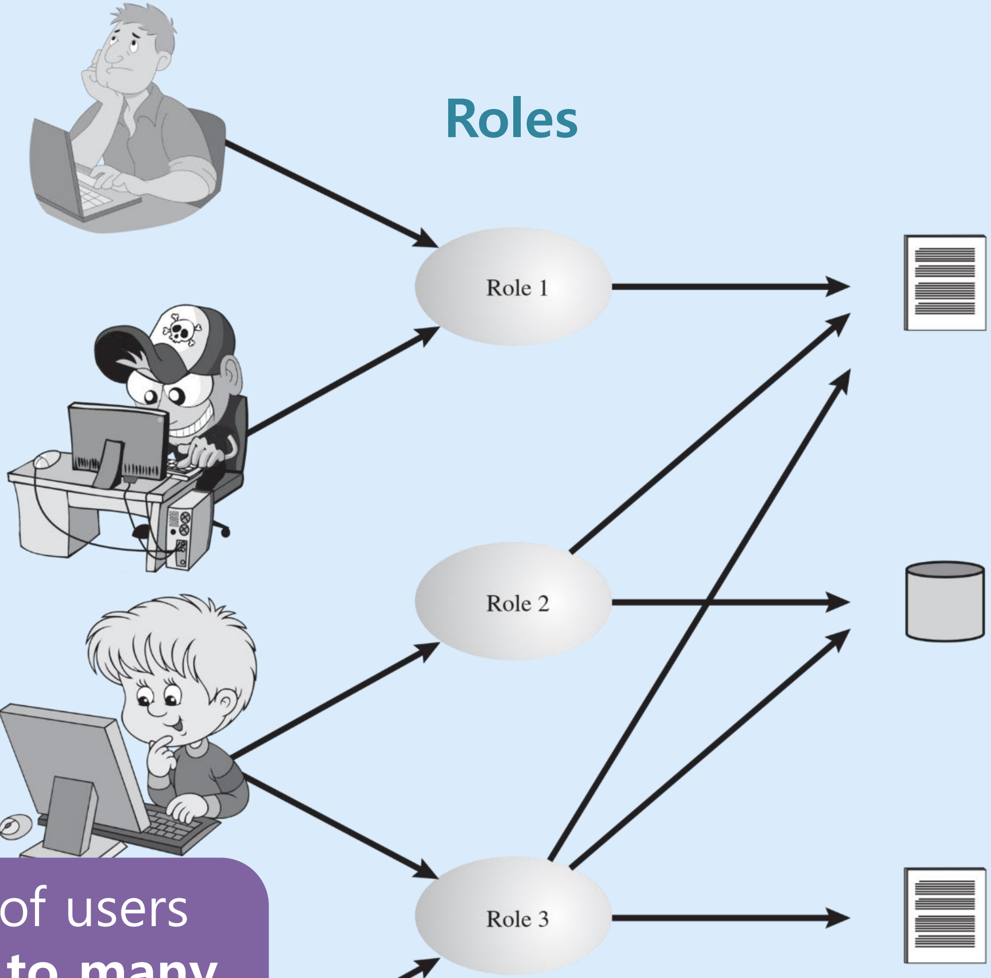
Resources
(Objects)

Role Based Access Control

Roles

Subjects

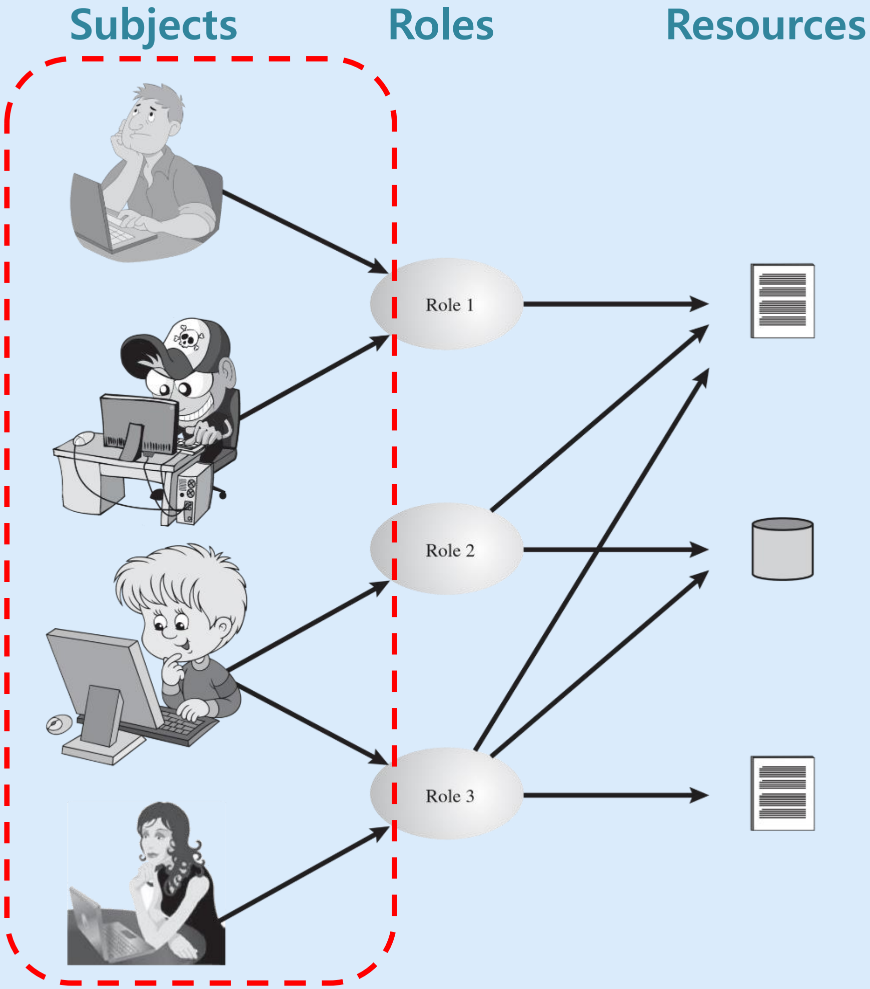
Resources (Objects)



The relationship of users to roles is **many to many**, as is the relationship of roles to objects.

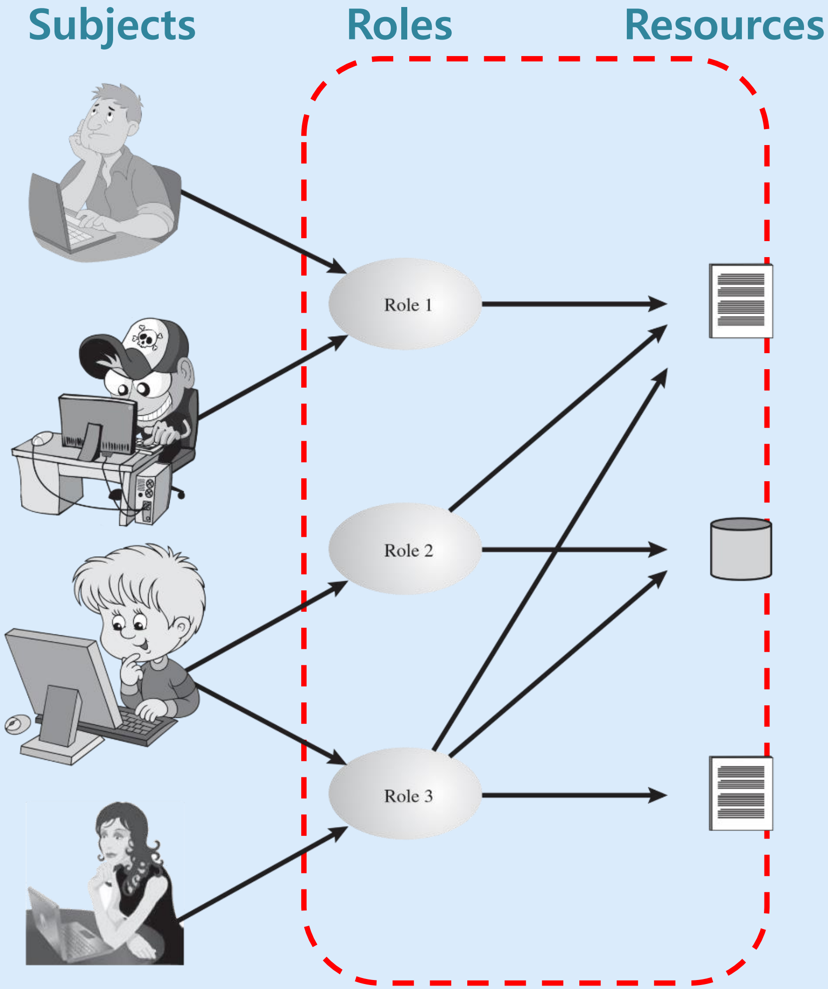
Role Based Access Control

- The set of users could change frequently.
- Assignment of a user to one or more roles is also dynamic.

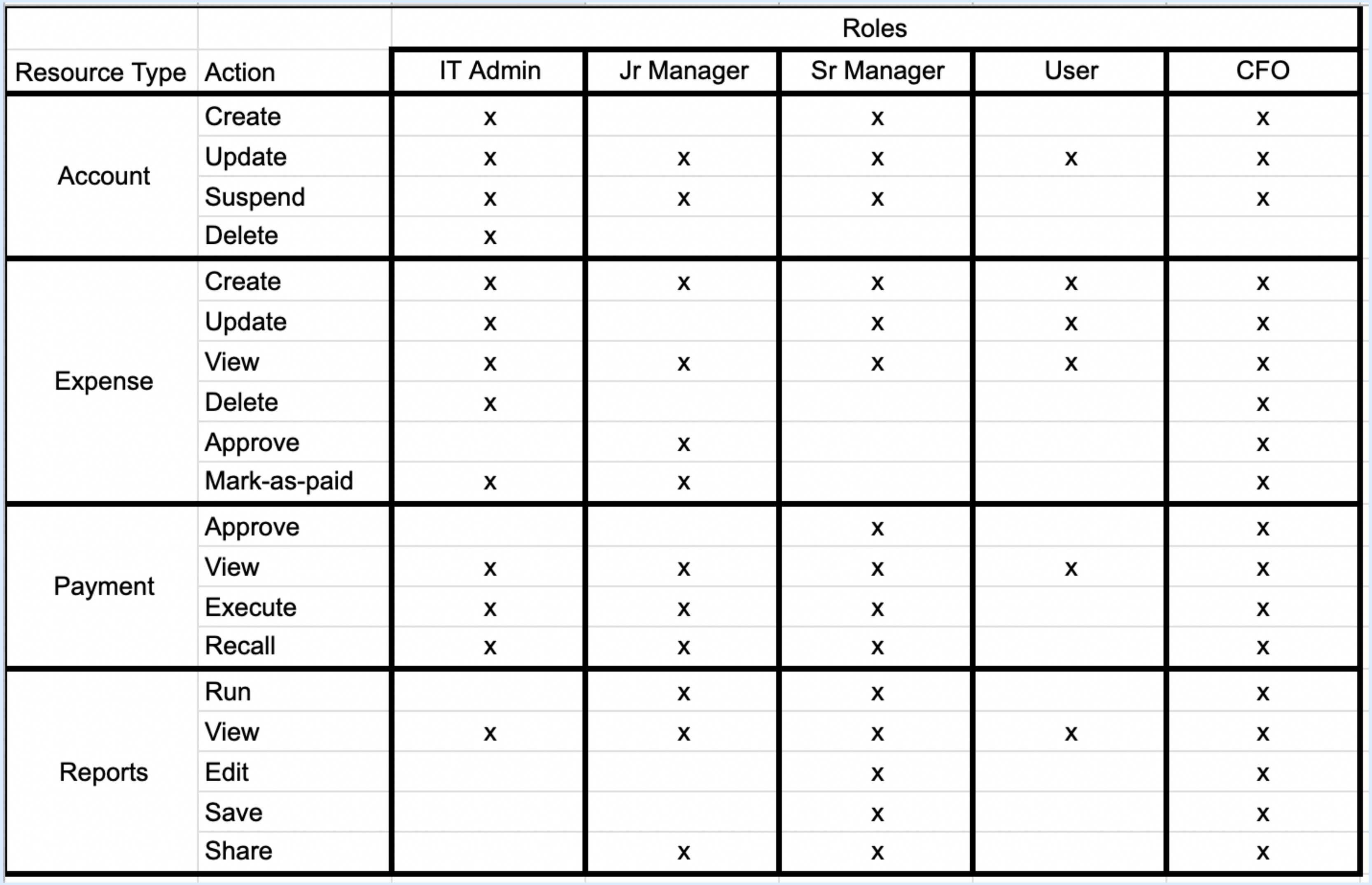


Role Based Access Control

- The set of roles in the system in mostly static, with occasional updates.
- The set of resources and the specific access rights associated with a particular role also change infrequently.



Example: Role privilege matrix



		Roles				
Resource Type	Action	IT Admin	Jr Manager	Sr Manager	User	CFO
Account	Create	x		x		x
	Update	x	x	x	x	x
	Suspend	x	x	x		x
	Delete	x				
Expense	Create	x	x	x	x	x
	Update	x		x	x	x
	View	x	x	x	x	x
	Delete	x				x
	Approve		x			x
	Mark-as-paid	x	x			x
Payment	Approve			x		x
	View	x	x	x	x	x
	Execute	x	x	x		x
	Recall	x	x	x		x
Reports	Run		x	x		x
	View	x	x	x	x	x
	Edit			x		x
	Save			x		x
	Share		x	x		x

https://www.cerbos.dev/blog/mapping-business-requirements-to-authorization-policy

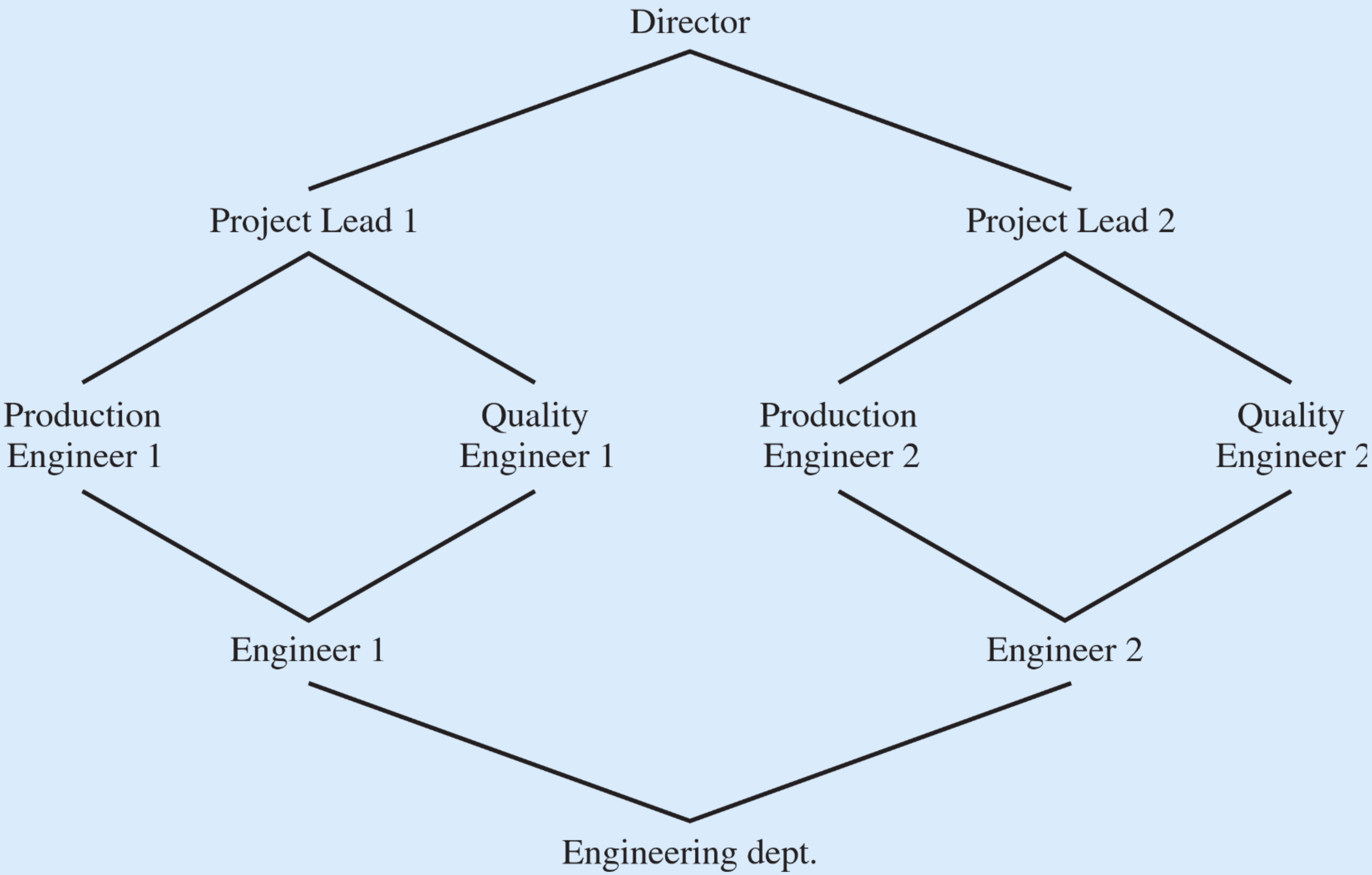
Example: Role assignments

	IT Admin	Junior Manager	User	Accounts officer	CFO
Ali	X				
Hania		X			
Faiqa	X		X		
Zaheer					X
Usman		X		X	

RBAC Hierarchies

- Its possible to design RBAC in a hierarchical fashion, reflecting the hierarchical structure of organization.
- Typically, job functions with greater responsibility have greater authority to access resources. A subordinate job function may have a subset of the access rights of the superior job function.
- Role hierarchies make use of the inheritance to enable one role to implicitly include access rights associated with a subordinate role.

RBAC Hierarchies



DAC vs RBAC

"It's easy to confuse DAC and RBAC because they can both use groups to organize users into manageable units, but they differ in their deployment and use. In the DAC model, objects have owners, and the owner determines who has access. [...] In a strict RBAC model, administrators do not assign privileges to users directly but only grant privileges by adding user accounts to roles or groups."

CISSP Study Guide

Attribute-Based Access Control

- An ABAC model can define authorizations that express conditions on properties of object, subject or environment.
- Attributes are characteristics that define specific aspects of the subject, object, environment conditions, and/or requested operations that are predefined and preassigned by an authority.

ABAC Model: Terms

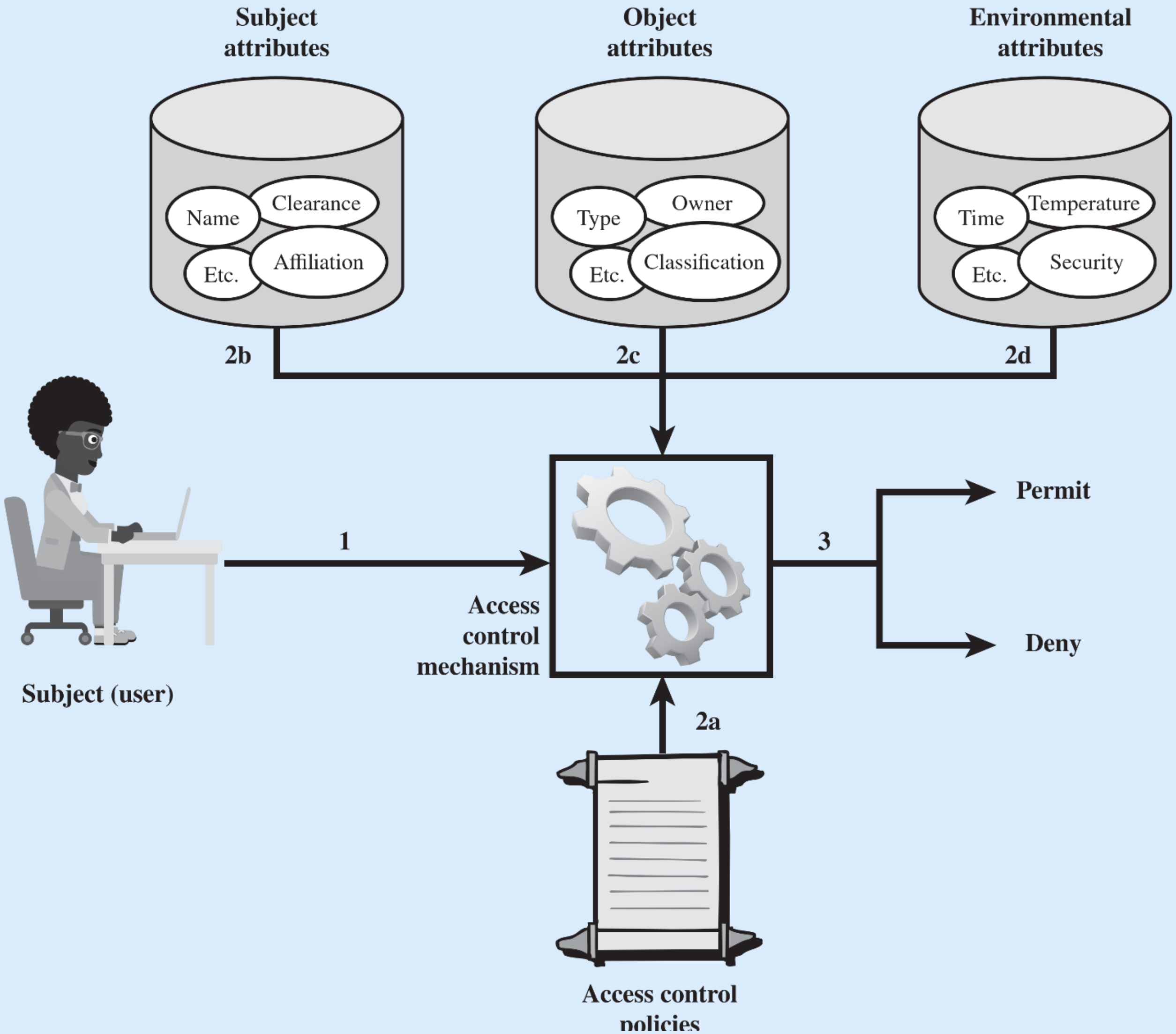
subject	an active entity (user, process, device) that causes information flow or change in system state
object	a passive entity containing or receiving information
environment	the operational, technical, or situational context in which the information access occurs

ABAC Model: Attributes

Attribute examples

Subject	Object	Environment
• id • name • date of birth • organization • job title • training record • location • experience duration • role	• type • date created • date modified • title • author • size • state (e.g. on, off, asleep) • ownership	• date & time • temperature • network security level (Internet or intranet) • current attack situation

ABAC Model


Subject attributes
Object attributes
Environmental attributes
Name
Clearance
Etc.
Affiliation
Type
Owner
Etc.
Classification
Time
Temperature
Etc.
Security
2b
2c
2d
1
3
Permit
Deny
Access control mechanism
Subject (user)
2a
Access control policies

ABAC Policy Rules

- ABAC controls access to objects by evaluating rules against the attributes of entities, operations, and the environment relevant to a request
- Access to resource will be allowed as per the access control rule defining the allowable operations for subject-object attribute combinations in a given environment
- A policy is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions
- Privileges represent the authorized behavior of a subject and are defined by an authority and embodied in a policy

ABAC Policy Rules

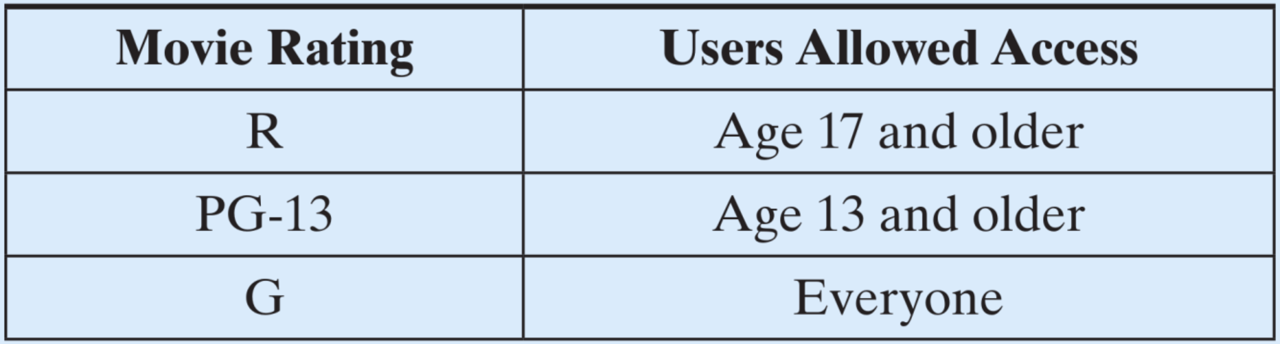
- In general form, a Policy Rule, which decides on whether a subject s can access an object o in a particular environment e, is a Boolean function of the attributes

$$\texttt{Rule: can\_access }(s,\ o,\ e) \leftarrow f(\texttt{ATTR}(s),\ \texttt{ATTR}(o),\ \texttt{ATTR}(e))$$

- If the function f evaluates to true, access to the resource is granted; otherwise denied.

Case Study: Movie Streaming

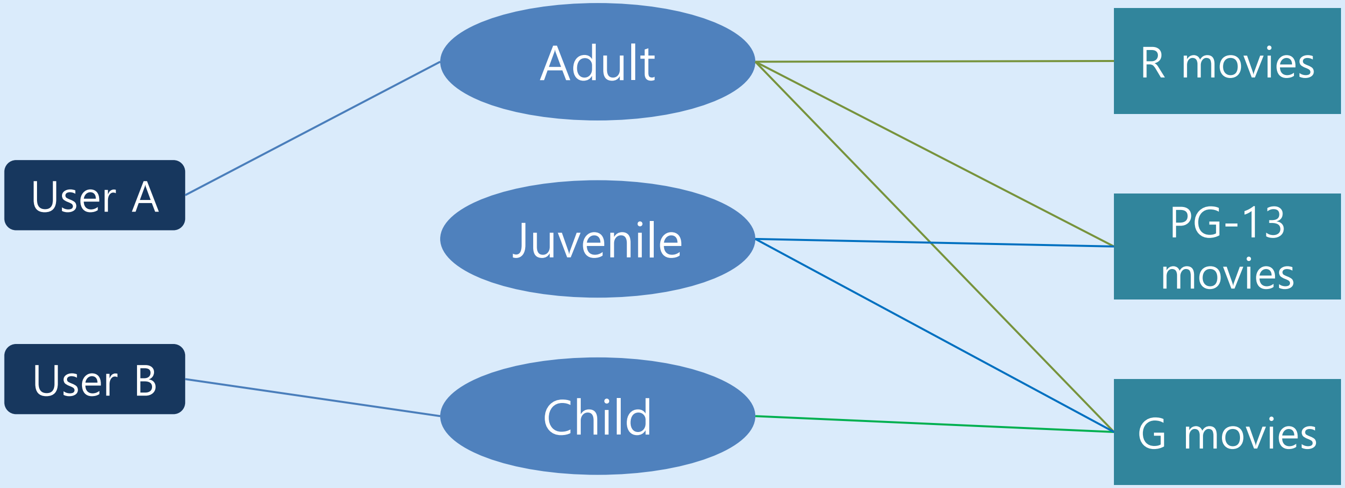
- An online entertainment store streams movies to users for a flat monthly fee.
- The store enforces the following access control policy



Movie Rating	**Users Allowed Access**
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

Case Study: Movie Streaming

- In an RBAC model, every user would be assigned one of three roles: Adult, Juvenile, or Child, possibly during registration.
- Both the user-to-role and permission-to-role assignments are manual administrative tasks.



Case Study: Movie Streaming

- The ABAC approach to this application does not need to explicitly define roles
- Instead, whether a user u can access or view a movie m (in a security environment e which is not used yet) would be resolved by evaluating a policy rule $R1$:

```
R1:can_access(u, m, e) ←
    (Age(u) ≥ 17 ∧ Rating(m) ∈ {R, PG-13, G}) ∨
    (Age(u) ≥ 13 ∧ Age(u) < 17 ∧ Rating(m) ∈ {PG-13, G}) ∨
    (Age(u) < 13 ∧ Rating(m) ∈ {G})
```

- Here Age and Rating are the subject and the object attributes respectively.

∧ = logic AND, ∨ = logic OR

Case Study: Movie Streaming

- Further, now suppose that movies are classified as either New Release or Old Release, based on release date.
- Users are classified as Premium User or Regular User, based on the fee they pay.
- The company would like to enforce a policy: *only premium users can view new movies*.

Case Study: Movie Streaming

- To incorporate these new requirements in the RBAC model, we would have to:
  - double the number of roles, to distinguish each user by age and fee (Adult_Premium, Adult_Regular, Child_Premium, ...)
  - double the categories of movies (G_New, G_Old, PG13_New, PG13_Old, ...)
- And so the number of role-to-object permissions will increase four times.

Case Study: Movie Streaming

- In contrast, the ABAC model deals with additional attributes efficiently. The rule $R1$ defined previously still applies. We need two new rules:

R2:can_access(u, m, e) ←
  (MembershipType(u) = Premium) ∨
  (MembershipType(u) = Regular ∧ MovieType(m) = OldRelease)
R3:can_access(u, m, e) ← R1 ∧ R2

- It is also easy to add environmental attributes, e.g.
  - A new policy states: Regular users are allowed to view new releases in promotional periods.
  - We only need to add one more OR condition in $R2$ that checks to see the environmental attribute *today's date* falls in a promotional period.

ABAC strengths

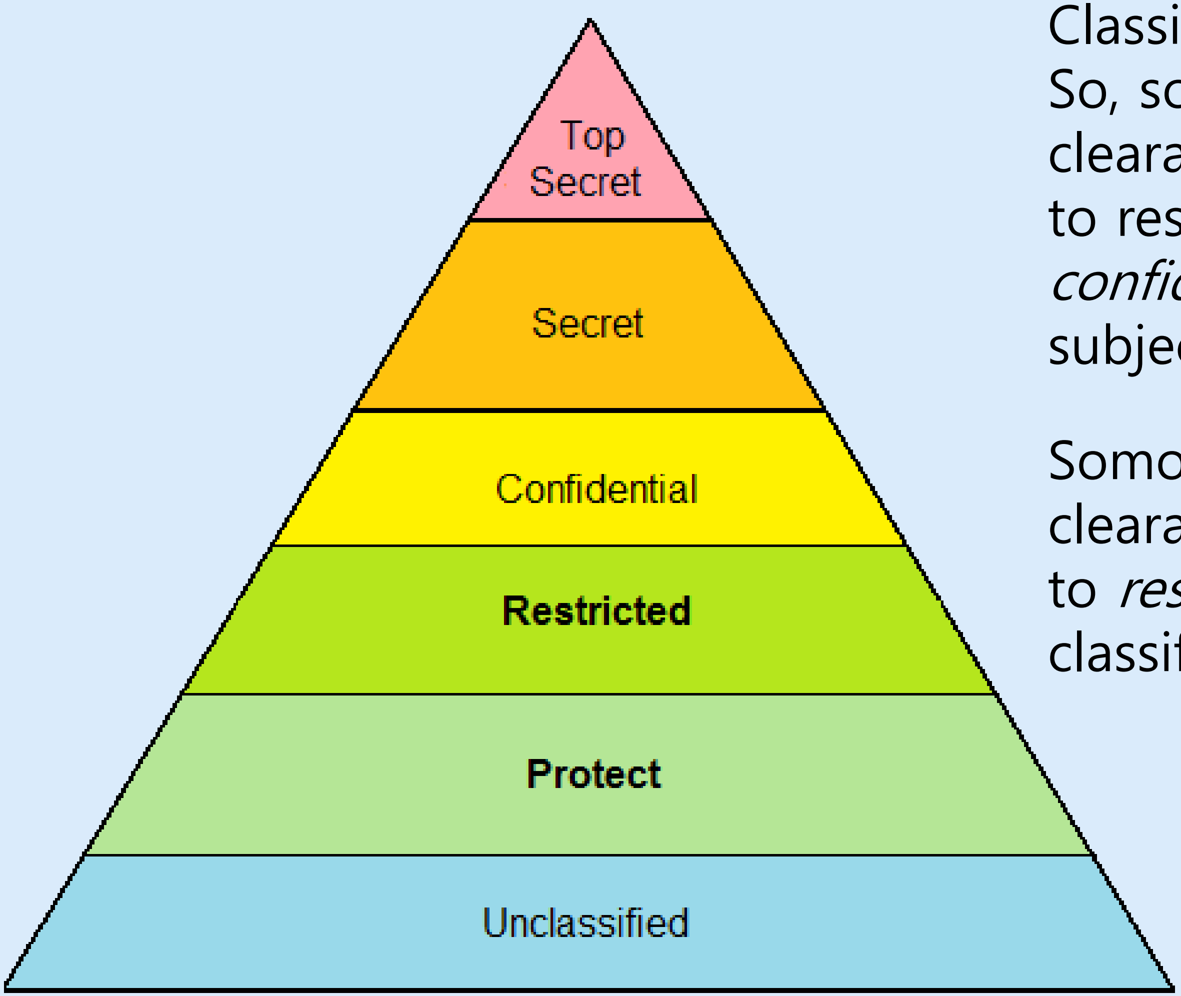
- Strength of ABAC model is its flexibility and expressive power
- ABAC is actually a more generalized model. It is capable of enforcing DAC, RBAC, and MAC concepts.
- It allows an unlimited number of attributes to be combined to satisfy any access control rule.
- ABAC model eliminates the definition and management of static roles, hence eliminating the need for the administrative tasks for user-to-role assignment and permission-to-role assignment.

Extra Material

Mandatory Access Control (MAC)

- MAC model evolved out of requirements of government and military applications
- It applies **labels** (or **classification**) to objects
  - e.g. confidential, top secret, secret, public, etc.
- Similarly, subjects are given a security **clearance**
- MAC controls access based on comparing security classifications with security clearances
  - e.g. someone with secret clearance can't access top secret data
- The system owner/administrator manage access rights themselves, individual users have no control

Classifications: UK govt example



Classifications are hierarchical. So, someone with *confidential* clearance *could* have access to resources that are classified *confidential* or below (but subject to other constraints).

Somone with *protected* clearance can NOT have access to *restricted* (or higher classification) resources.

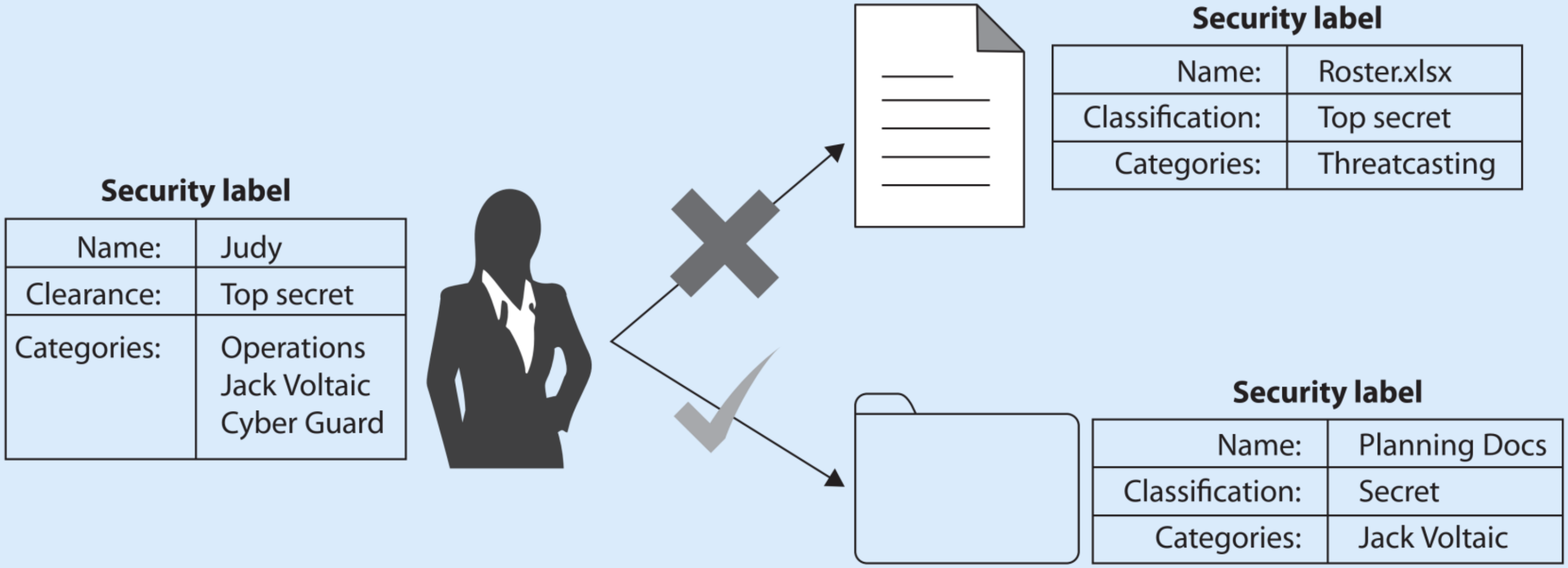
https://en.wikipedia.org/wiki/Government_Security_Classifications_Policy

MAC Rules

- When the system makes a decision about fulfilling an access request, it is based on the clearance of the subject, the classification of the object, **and the security policy of the system**.
- This means that even if a user has the right clearance to read a file, specific policies (e.g., requiring “need to know”) could still prevent access to it.
- Each object is assigned a security label containing
  a) classification: sensitivity level
  b) category: the sub-compartment of information

MAC example

- Judy has Top secret clearance. She can access objects classified as Top secret or lower but ONLY if the object's category belongs to set of categories in Judy's security label.



Security label

Name:	Judy
Clearance:	Top secret
Categories:	Operations Jack Voltaic Cyber Guard

Security label

Name:	Roster.xlsx
Classification:	Top secret
Categories:	Threatcasting

Security label

Name:	Planning Docs
Classification:	Secret
Categories:	Jack Voltaic

CISSP Exam Guide, Harris Maymi