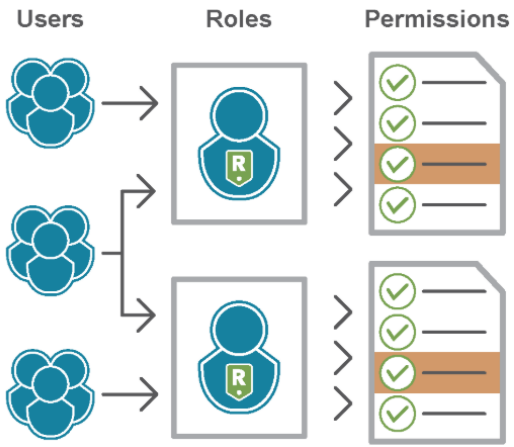
Authorization

Outline

1. Role-based Access Control (RBAC)
2. Attribute-based Access Control (ABAC)

Role-based Access Control (RBAC)



Back

Authorization

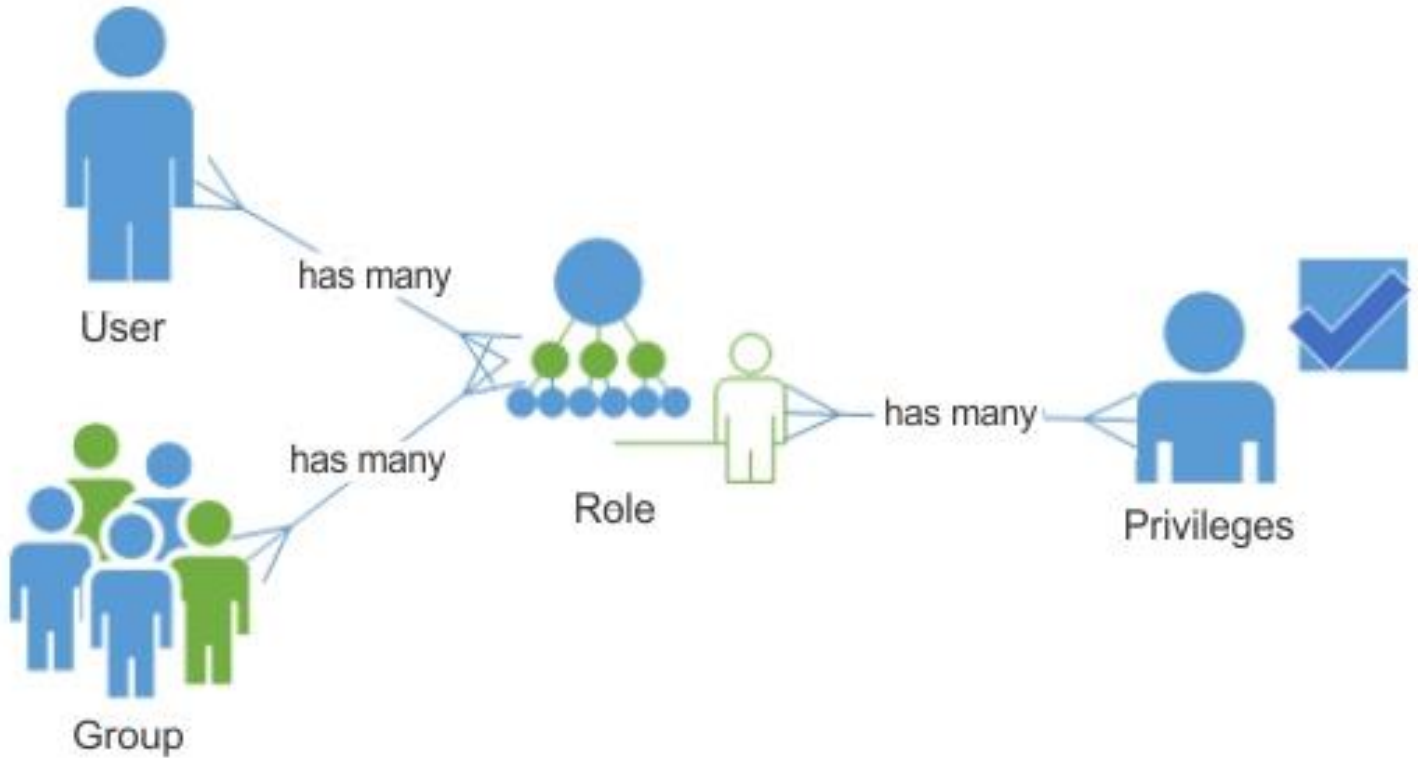
- Authorization follows authentication
  - After establishing who the user is, the system needs to decide what they can do
- Authorization is usually represented as a policy specification of what resources can be accessed by a given subject (i.e., *Who* can access *what*?)
  - Can also include the nature of the access (e.g., read, write)

Subject **or** **Principal**	Users of a system could a person or an automated agent
Resource **or** **Object**	Resource acted upon by subjects

RBAC vs. MAC/DAC

- Mandatory Access Control (MAC) and Discretionary Access Control (DAC)
  - Users, Groups **have** Permissions on objects (e.g., files)
  - DAC allows the owner of the object to determine who has access. In MAC however the access rights are determined by the administrator/policy
- Role-based Access Control (RBAC)
  - Roles **have** Permissions
  - Users **have** Roles
  - Many-to-many relations
  - Simplifies maintainability in systems with lots of users
- Difference between groups and roles?
  - Groups: collection of users
  - Roles:
    - collection of permissions and possibly other roles
    - job function within an organization

Basic RBAC Illustrated



User Assignment
(UA)

Permission Assignment
(PA)

Access Matrix Representation

Role has permissions on resources

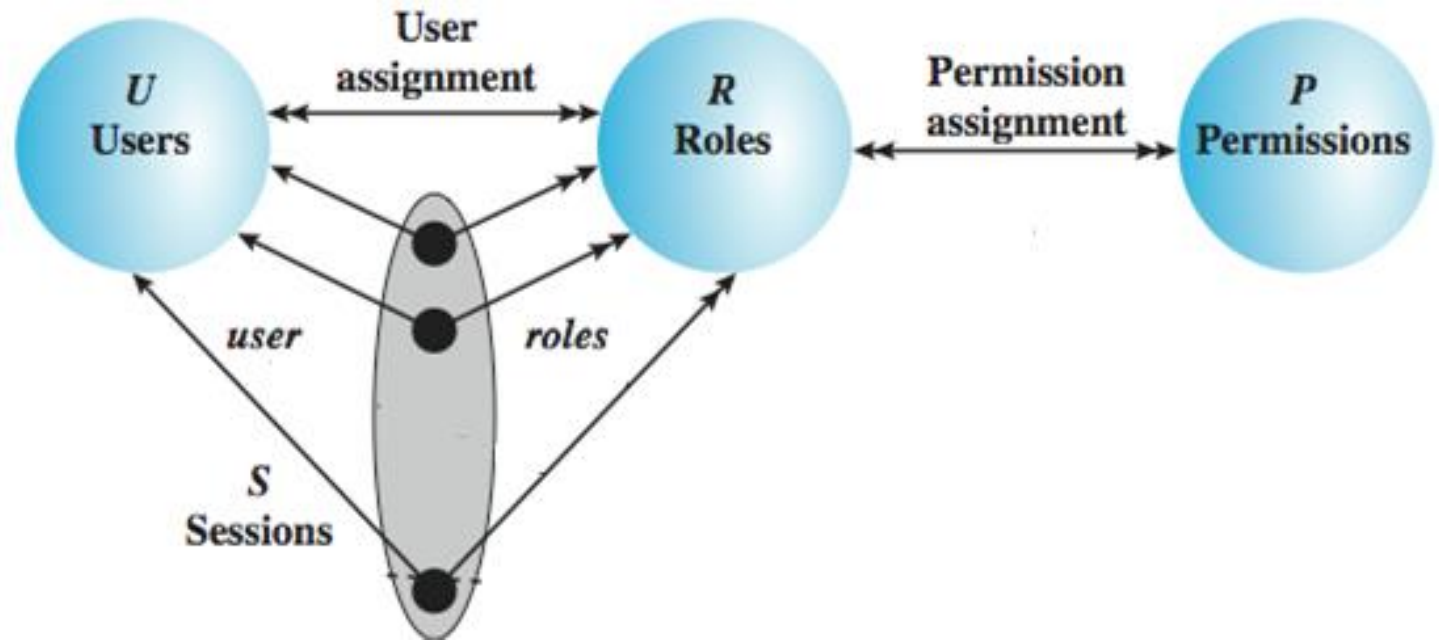
e.g., permission on blog posts

Role	Read	Write	Publish
manager	1	1	1
writer	1	1	0
guest	1	0	0

Users are assigned to roles

User	Role
Ali	Writer
Fatma	Manager
Samira	Guest

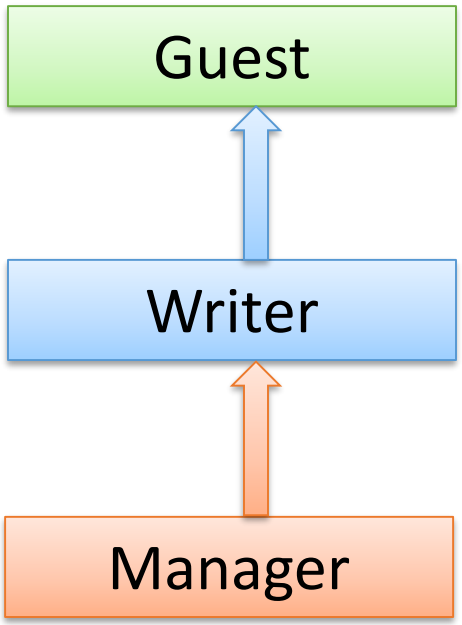
RBAC – Base



- **Users:** individuals with access to the system
- **Role:** named job function within the org
- **Permission:** approval of a particular mode of access to objects
- **Session:** mapping between a user and a subset of roles

RBAC – Role Hierarchies

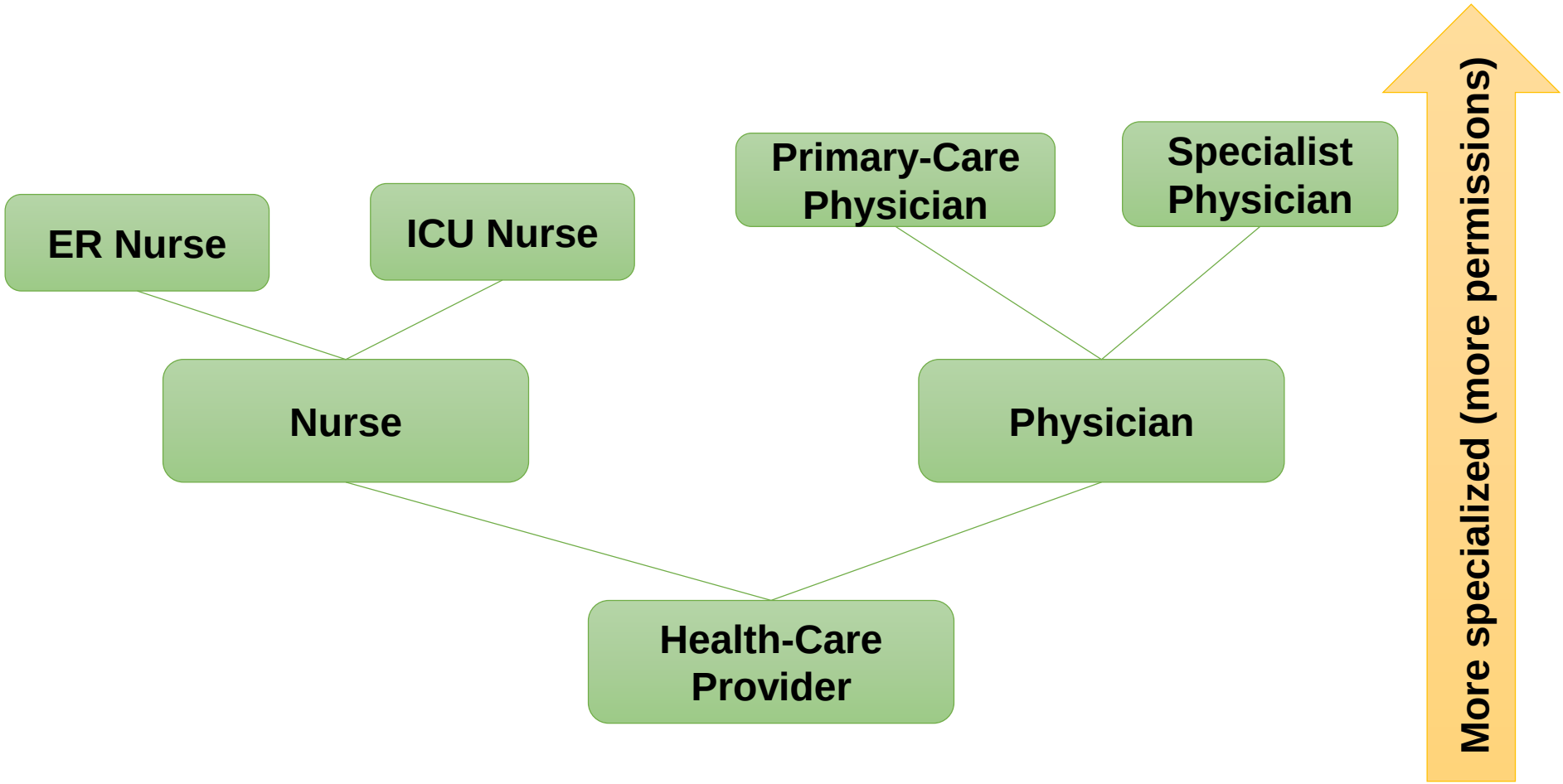
- Reflect hierarchical structure of roles in org




```
let roles = {
    manager: {
        can: ['publish'],
        inherits: ['writer']
    },
    writer: {
        can: ['write'],
        inherits: ['guest']
    },
    guest: {
        can: ['read']
    }
}
```

- Manager role has all permissions of Writer role
- Writer roles has all permission of Guest role

Role Hierarchy Example



ICU - Intensive Care Unit

ER - Emergency Room

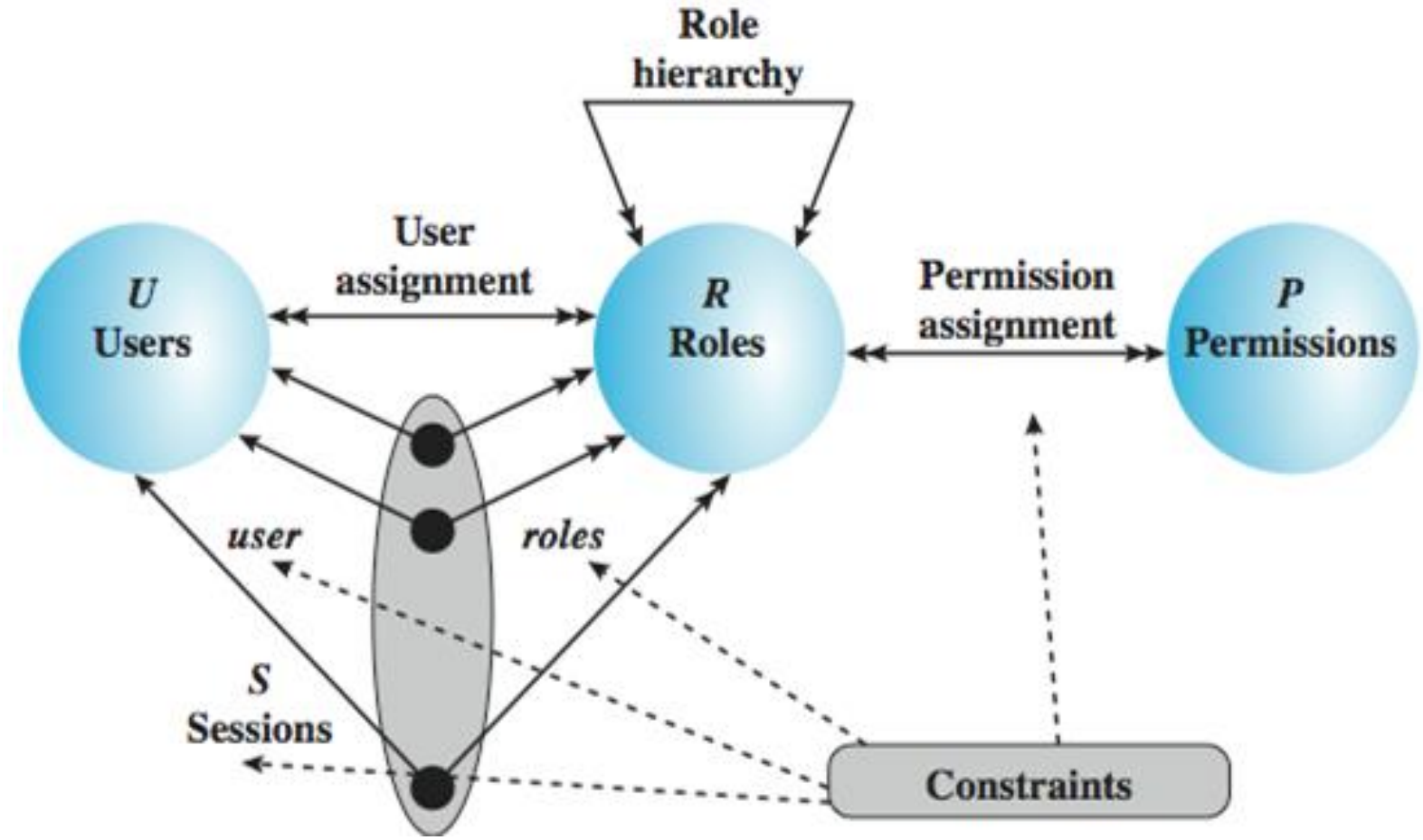
RBAC – Constraints

- Reflect organizational policies
- **Mutually exclusive roles:** no user is allowed to be a member of both roles (e.g., Expenses & Expense Approvals, Hire & Approve Hire)

→ Separation of Duty to prevent conflict of interest

- **Cardinality** – maximum number with respect to role
- **Prerequisite** – can assign role only if already assigned prerequisite role

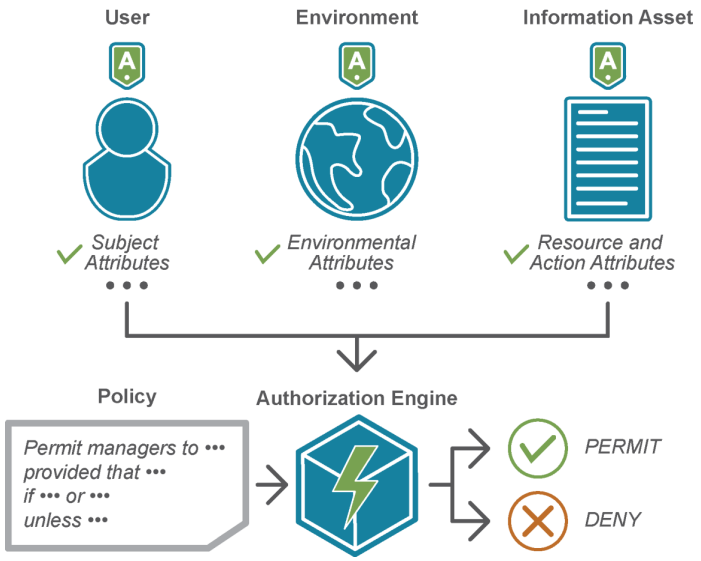
RBAC – Consolidated Model



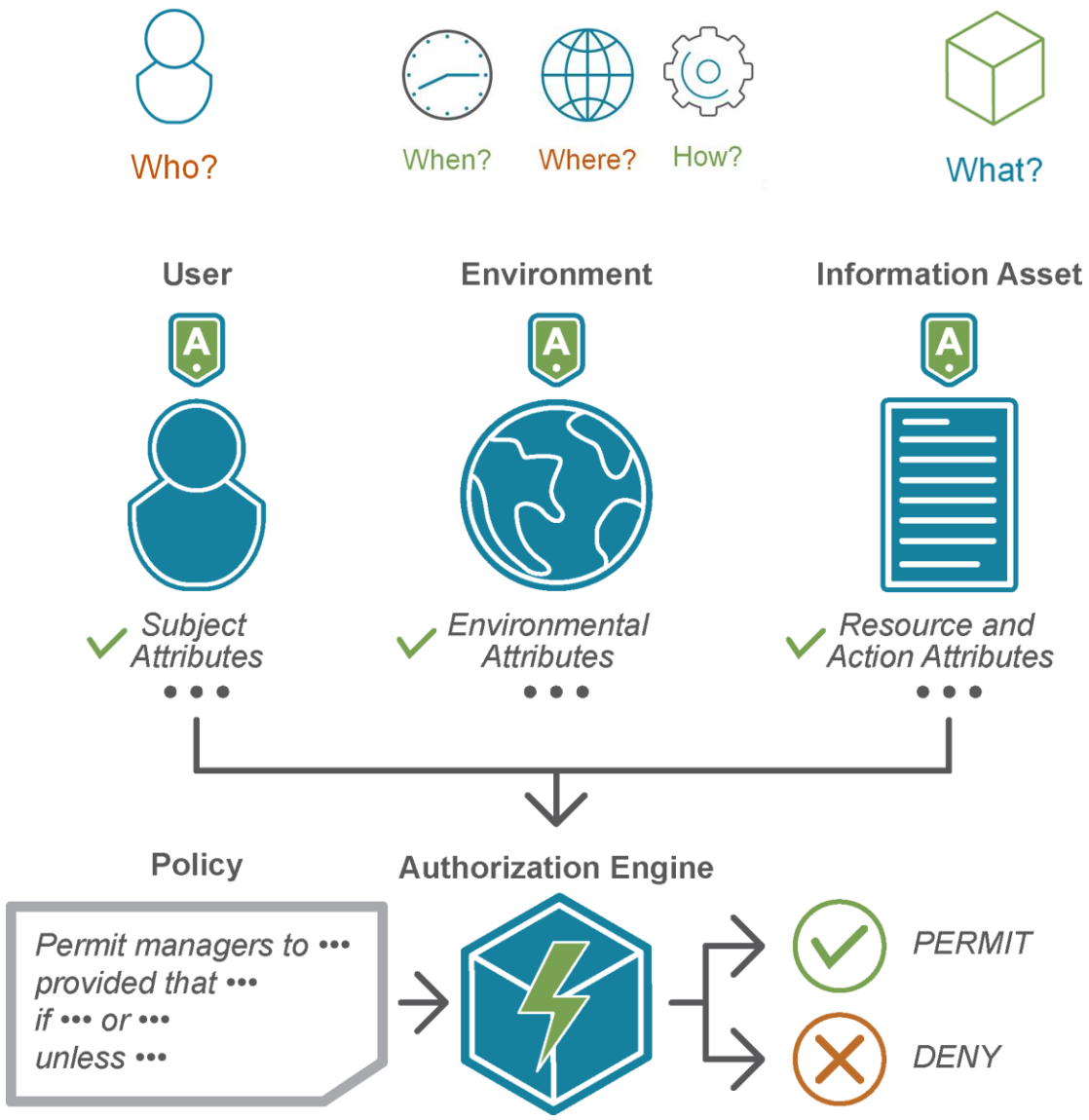
Key Points

- Roles are collections of permissions, users, and possibly other roles (many-to-many)
- Role hierarchies simplify RBAC management and can be derived from org structure
- Constraints prevent conflict of interest
- RBAC implementations simplify access control but may require role engineering

Attribute-based Access Control (ABAC)



Attribute-Based Access Control (ABAC)



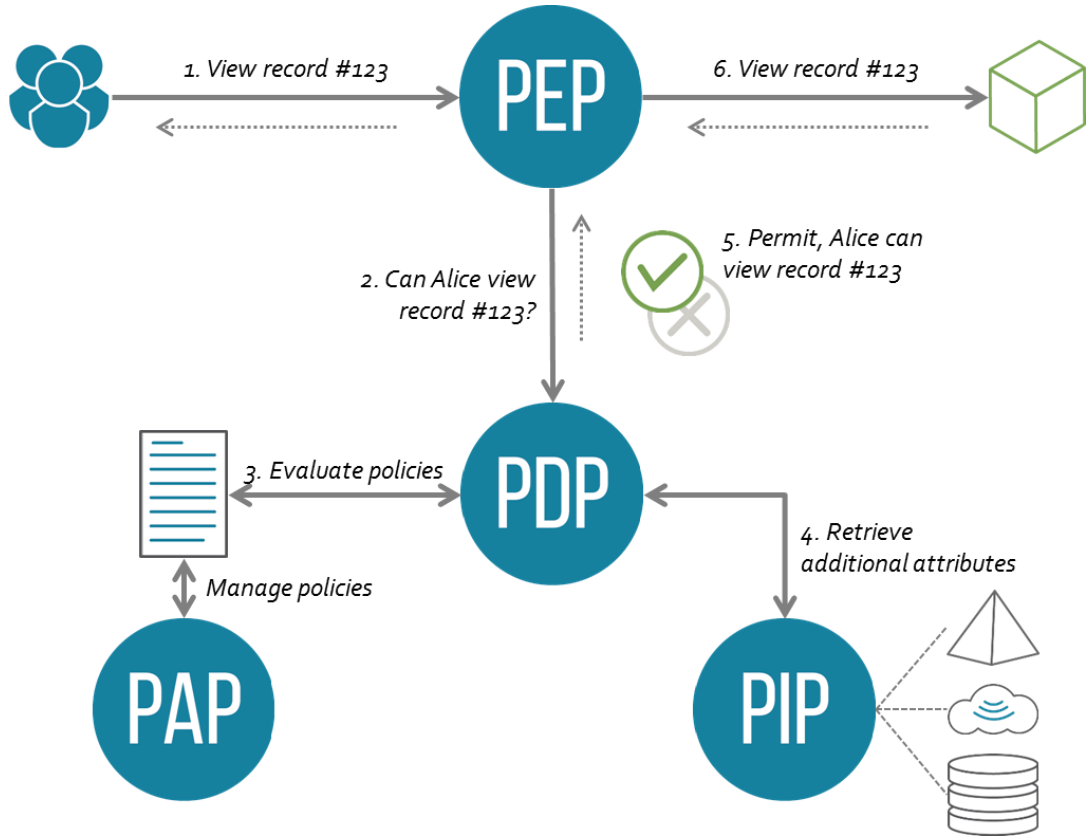
Subject attributes	Resource attributes	Environment attributes
• A subject is an active entity that access resources or change the system state • Attributes define the identity and characteristics of the subject	• A resource is the information asset to protect • Resource have attributes that can be used to make access control decisions • (e.g. student can only get their transcript and not other student transcripts)	• Describe the operational, technical, and environment or context in which the information access occurs • e.g., Time of access, place of access and device used

XACML "eXtensible Access Control Markup Language" is a standard that defines:

- A declarative attribute-based access control policy language
- An architecture to enforce and manage the policies
- A processing model describing how to evaluate access requests according to the rules defined in policies.

Architecture

PAP	Policy Administration Point	Point which manages access authorization policies
PDP	Policy Decision Point	Point which evaluates access requests against authorization policies before issuing access decisions
PEP	Policy Enforcement Point	Point which intercepts user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision
PIP	Policy Information Point	The system entity that acts as a source of attribute values (i.e. a resource, subject, environment)



ABAC model

bool **IsActionAllowed**(subject, resource, action) {

permit = true

for each rule in rules {

Examines the rule condition using the attributes of *subject* and *resource* as well as the *environment*

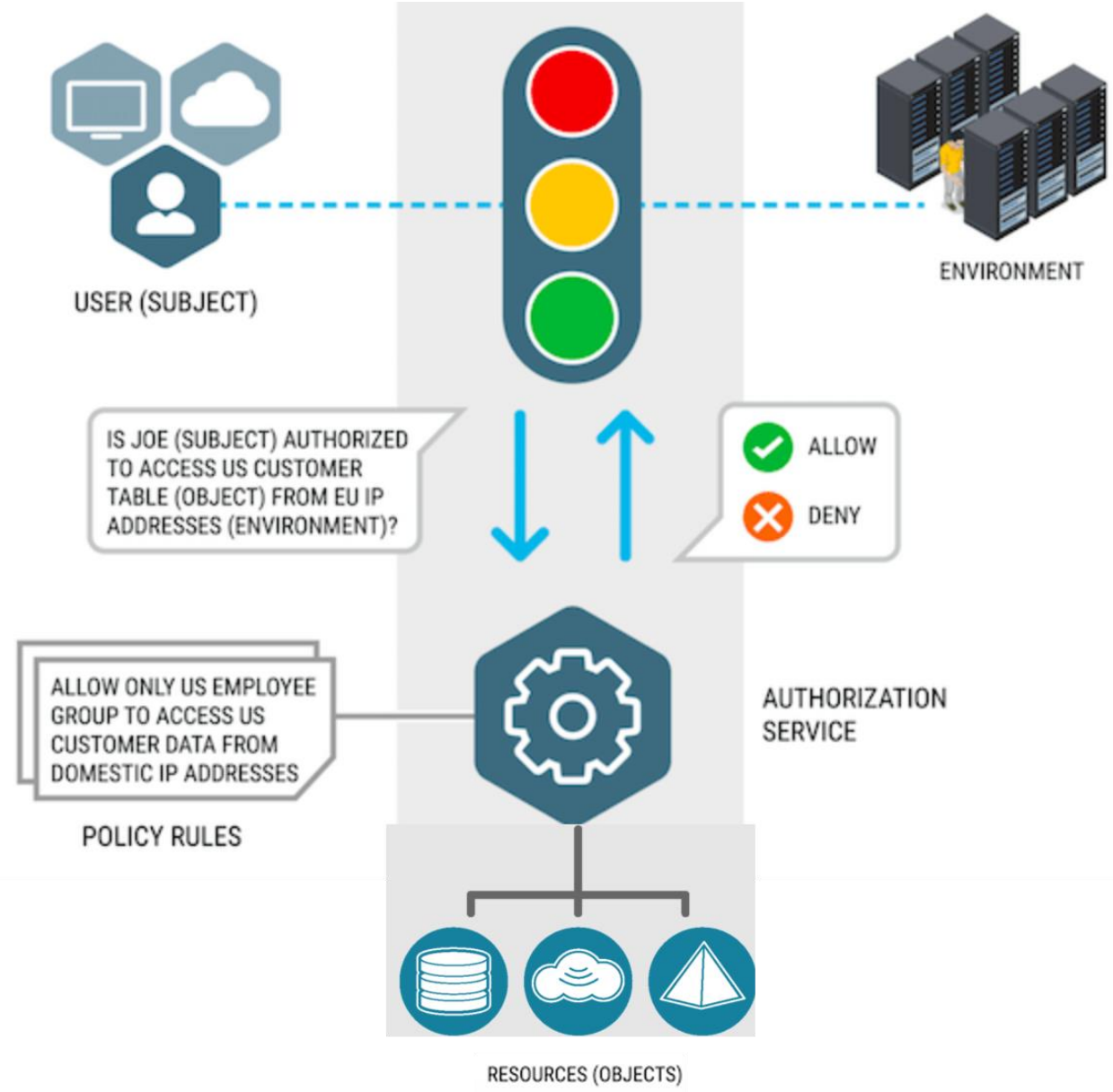
If rule not meet then permit = false

}

return permit

}

Scenario 1

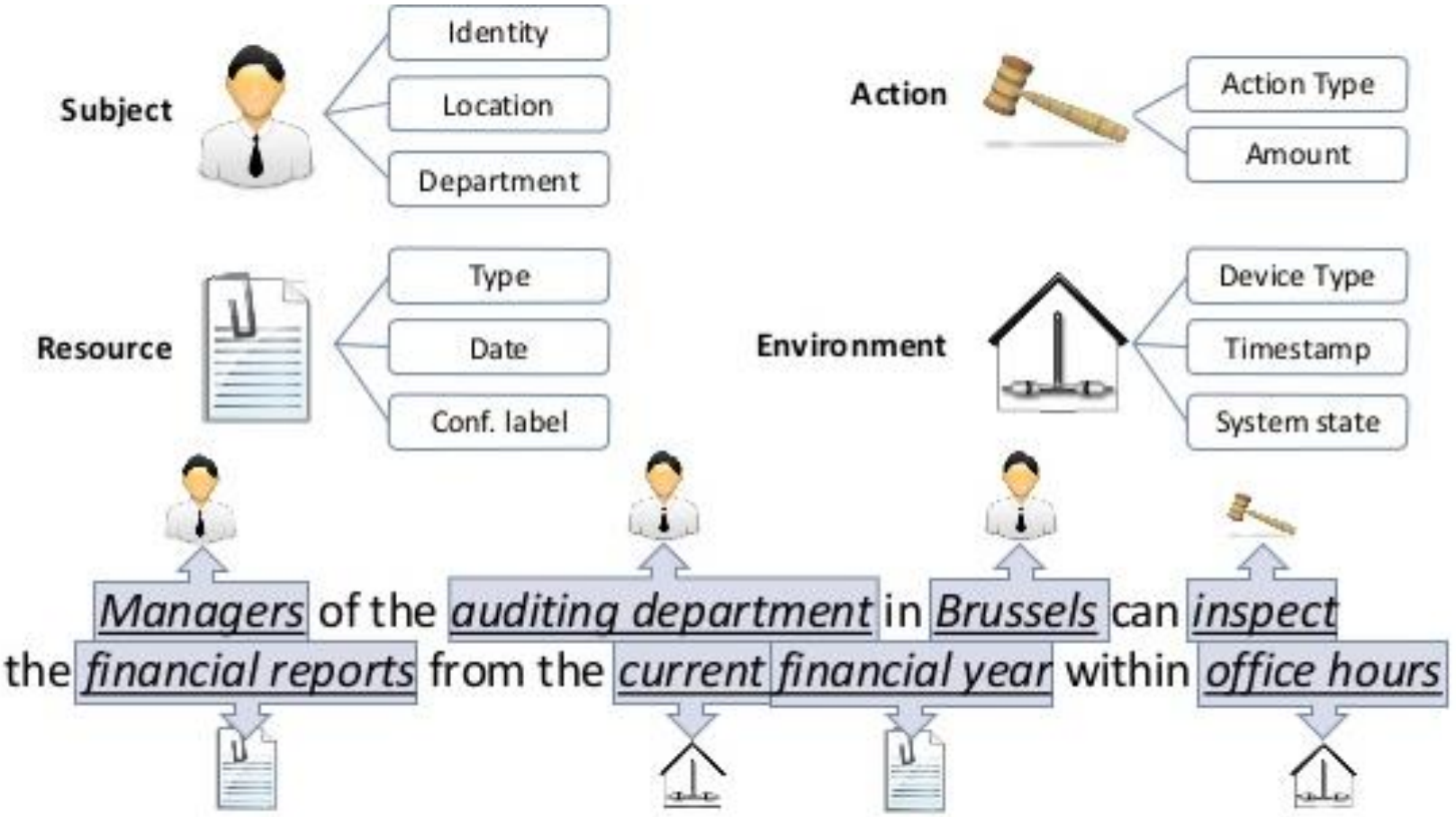

USER (SUBJECT)
ENVIRONMENT
IS JOE (SUBJECT) AUTHORIZED TO ACCESS US CUSTOMER TABLE (OBJECT) FROM EU IP ADDRESSES (ENVIRONMENT)?
ALLOW
DENY
ALLOW ONLY US EMPLOYEE GROUP TO ACCESS US CUSTOMER DATA FROM DOMESTIC IP ADDRESSES
POLICY RULES
AUTHORIZATION SERVICE
RESOURCES (OBJECTS)

Scenario 2

- A user with the role = “banking consultant” can do the action = “view” on documents of type = “loan application” if the loan status = “approved” and the loan customer’s branch = the consultant’s branch.

```
Allow access to resource MedicalJournal with attribute patientID=x
      if Subject match DesignatedDoctorOfPatient
      and action is read
```

Scenario 3


Subject
Identity
Location
Department
Action
Action Type
Amount
Resource
Type
Date
Conf. label
Environment
Device Type
Timestamp
System state
Managers of the auditing department in Brussels can inspect
the financial reports from the current financial year within office hours

Resources

- NIST Digital Identity Guidelines

https://pages.nist.gov/800-63-3/

- XAML

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