Protection & Security

Greg Bilodeau

CS 5204

October 13, 2009

Protection from:

- Malicious users
  - Modifying data
  - Accessing confidential data
  - Misuse of resources
- Collaborative efforts
  - Controlling access to data
  - Allowing the right users the right privileges

How to secure a system

- Authenticate
- Authorize
- Audit changes, prepare for recovery

- Close the system to un-trusted users
  - **Isolation**
  - **Exclusion (firewalls)**
- Restrict un-trusted users to sandbox
- File security system
- Best approach is combination of tactics

Principles of Security

- Assign only the rights needed (least privilege)
- Partition rights by duty or role (separation of duties)
- Enforce rights in simplest manner (economy of mechanism)
- Acceptable to user community (acceptability)
- Universal enforcement of policies (complete mediation)
- "Security through obscurity" not required (open design)

Goals for Security

- Only modified by authorized entities
- Only accessed by authorized entities (confidentiality)
- Cannot disclaim ownership (non-repudiation)
- Source can be verified (authenticity)
- Accessible to authorized entities

Goals for Collaborative Systems

- Applied and enforced at distributed platform level
- Generic and useful for variety of tasks
- Be scalable to large numbers of potential shared operations
- Protect data at various levels of granularity
- Transparent access for authorized, strong exclusion for unauthorized that doesn’t hamper collaboration
- Allow high-level access rights
- Dynamic, specify and change policies at run time
- Not hamper performance

Access Matrix

	O_1	O_2	O_3	O_4	O_5	O_6
S_1	*O	R	W	R		W
S_2	R		W		*R	W
S_3					R	

S_x	= Subject
O_x	= object
O	= Owner
R	= read access
W	= write access
*X	= copy bit set

Abstract reference model to associate subjects with objects (and other subjects), used by security monitor querying reference monitor

Access Matrix

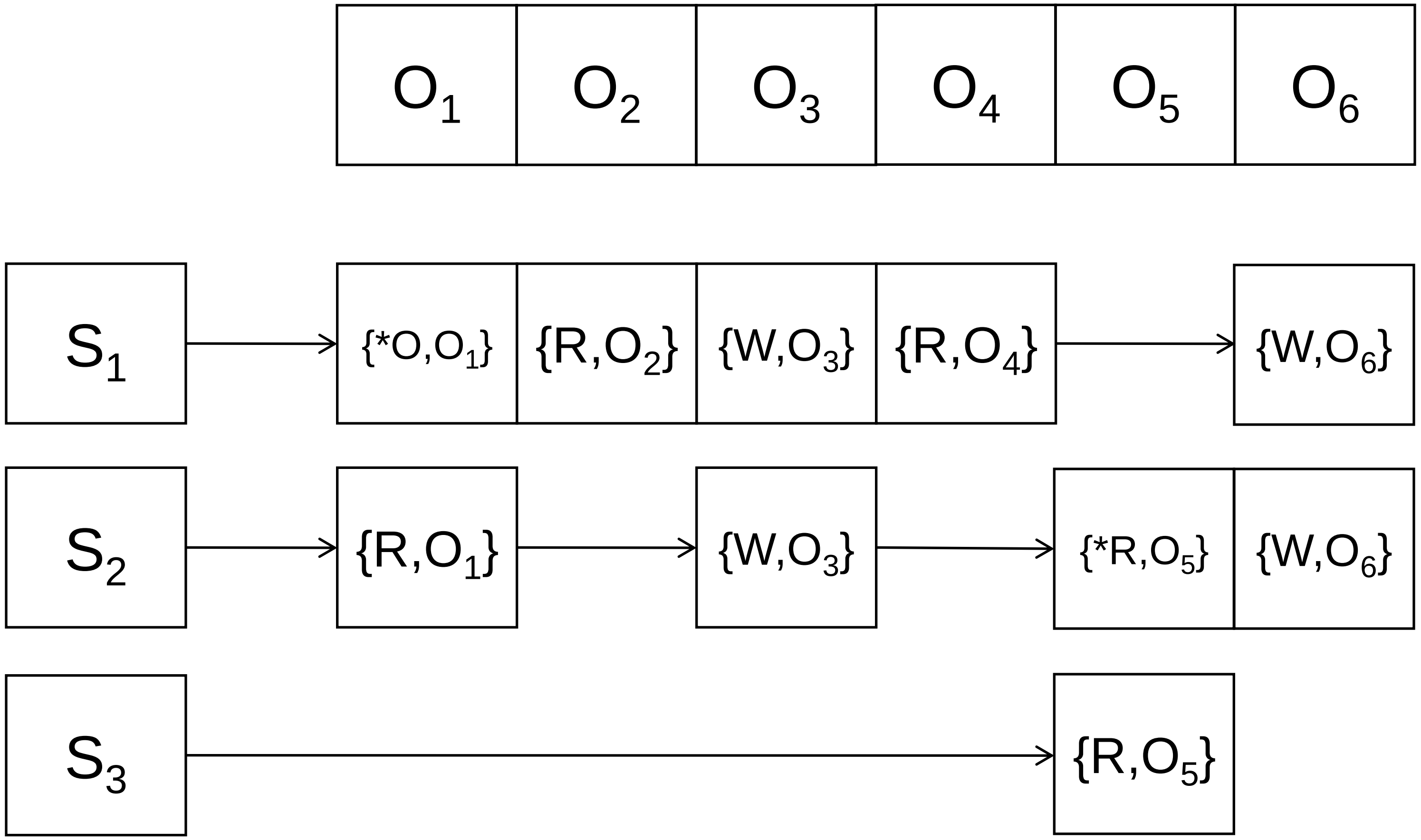
- May change over time
  - **Users added or removed**
  - **Files created or deleted**

- Commands manipulate the matrix
  - **First test if conditions for command are true, ex: check if subject attempting to give write access to another subject for an object has *W access right**
  - **If it does, update the entry for the target subject with the new access right**

Implementing the access matrix

- Matrix can be implemented in various ways
- Subject perspective
  - **Subjects**
  - **Roles**
  - **Teams**
- Object perspective
  - **Objects**
  - **Domains**
- System perspective
  - **Tasks**
  - **Clearance**
  - **Context**
  - **Lock and key**
  - **Memory mapping**

Capability Lists

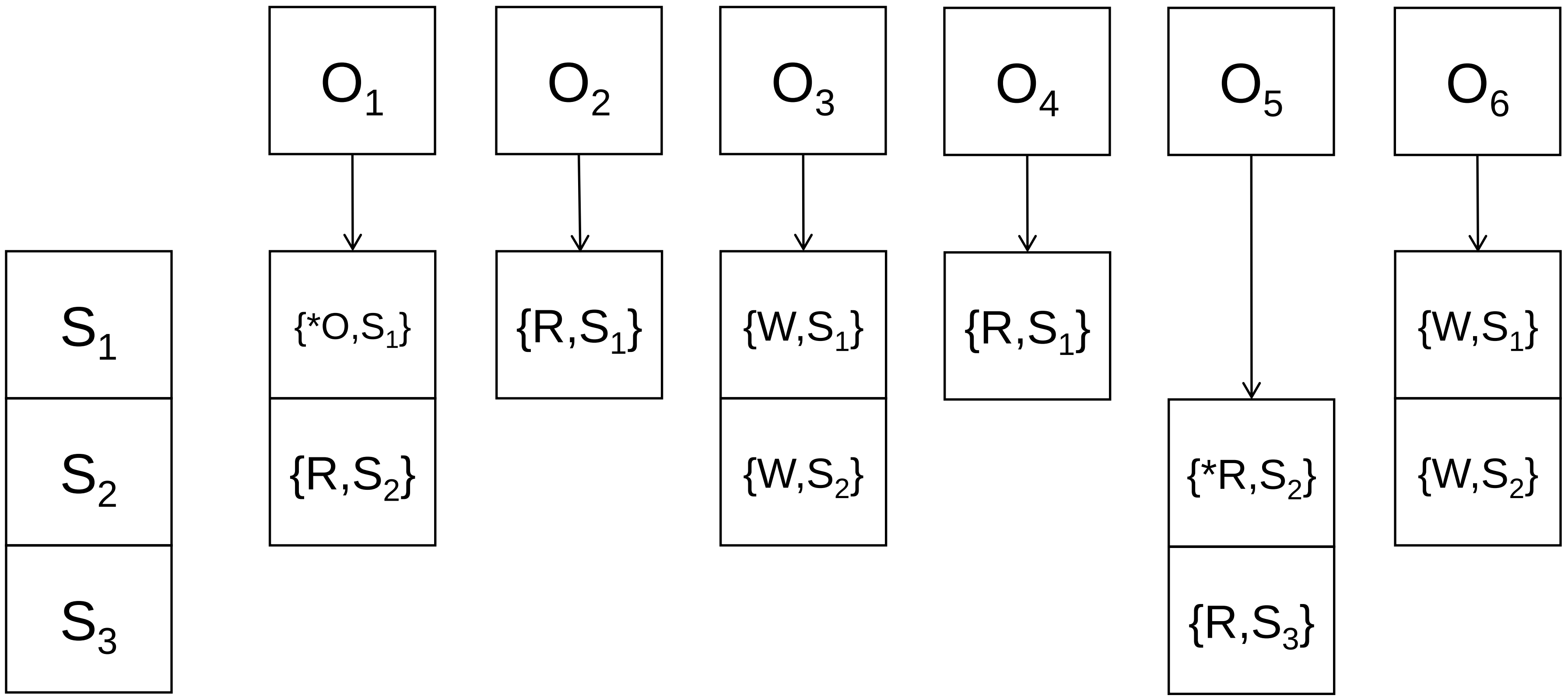


Capability Lists

- Defined as a list of {permission, object} pairs
- Gives entire access rights profile of each subject
- Easier to store in memory than entire matrix

- Subject incapable of starting action without permission

Access-control Lists



Access-control Lists

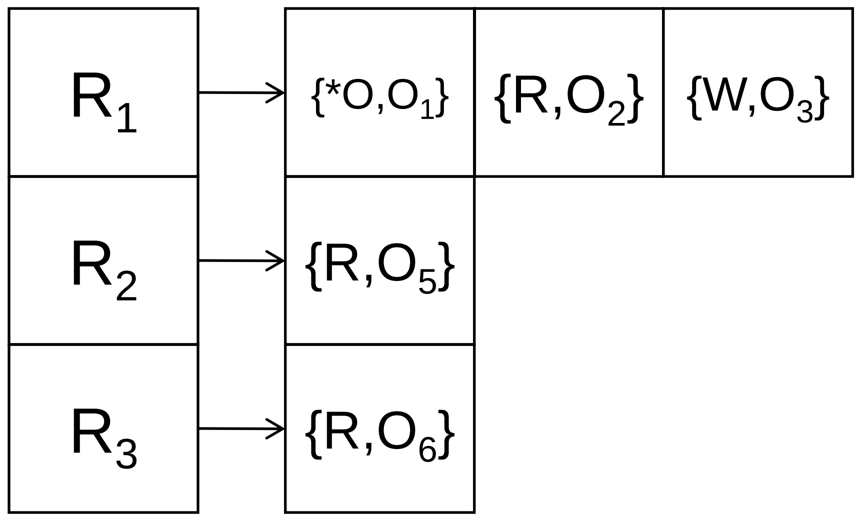
- Defined as a list of {permission, subject} pairs
- Gives entire access rights profile of each object
- Easier to store in memory than entire matrix

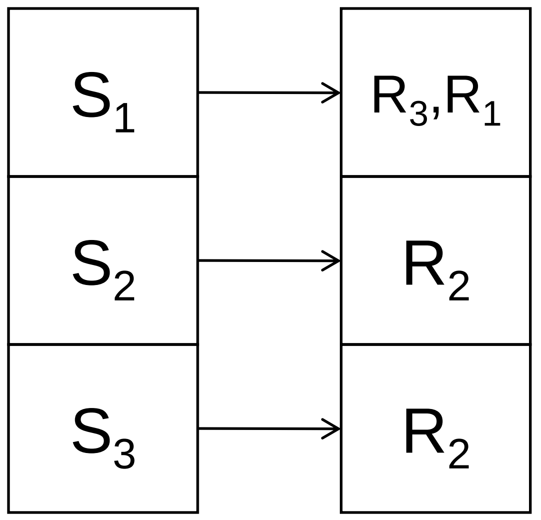
Capability List and Access-control List

- Difficult to dynamically change access rights:
  - Capability list: difficult to determine how many and which subjects have rights for a given object
  - Access-control list: difficult to tell all accesses a subject has across objects
- Do not account for contextual information
  - Context: environmental variables, dynamism and unpredictability
  - Context can define access when in a given role

Role-Based Access Control (RBAC)

Assigning access rights to roles



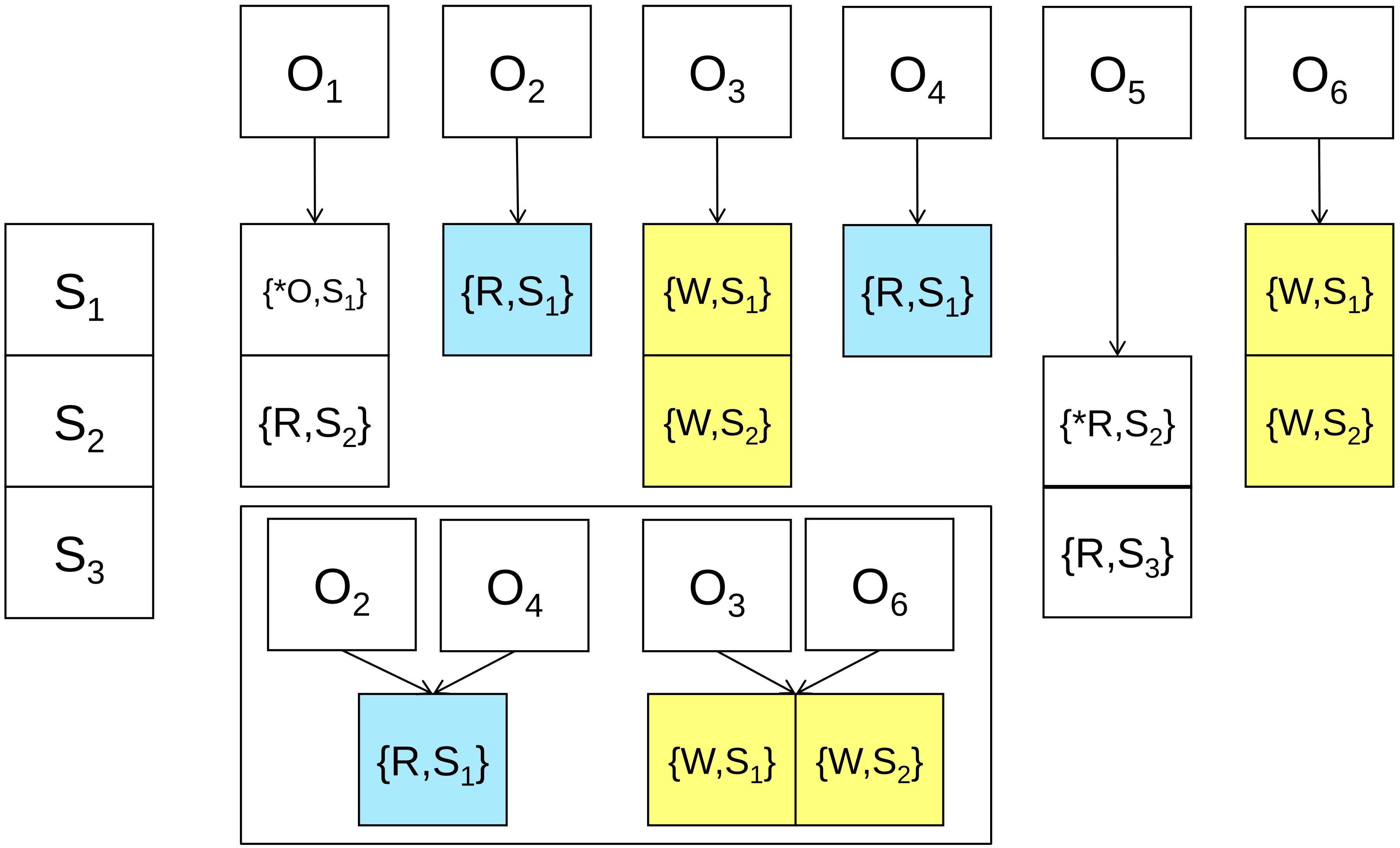


Assigning roles to subjects

Role-Based Access Control (RBAC)

- Roles created for various combinations of {object, rights} pairs, subject assigned one or more roles
- Assignment of rights to roles and roles to subjects can be done at different times
- Allows updating of multiple subject's access rights automatically through inheritance
- Allows cardinality and conflict of interest rules to be enforced
- However, cannot allow specific subject access to specific object

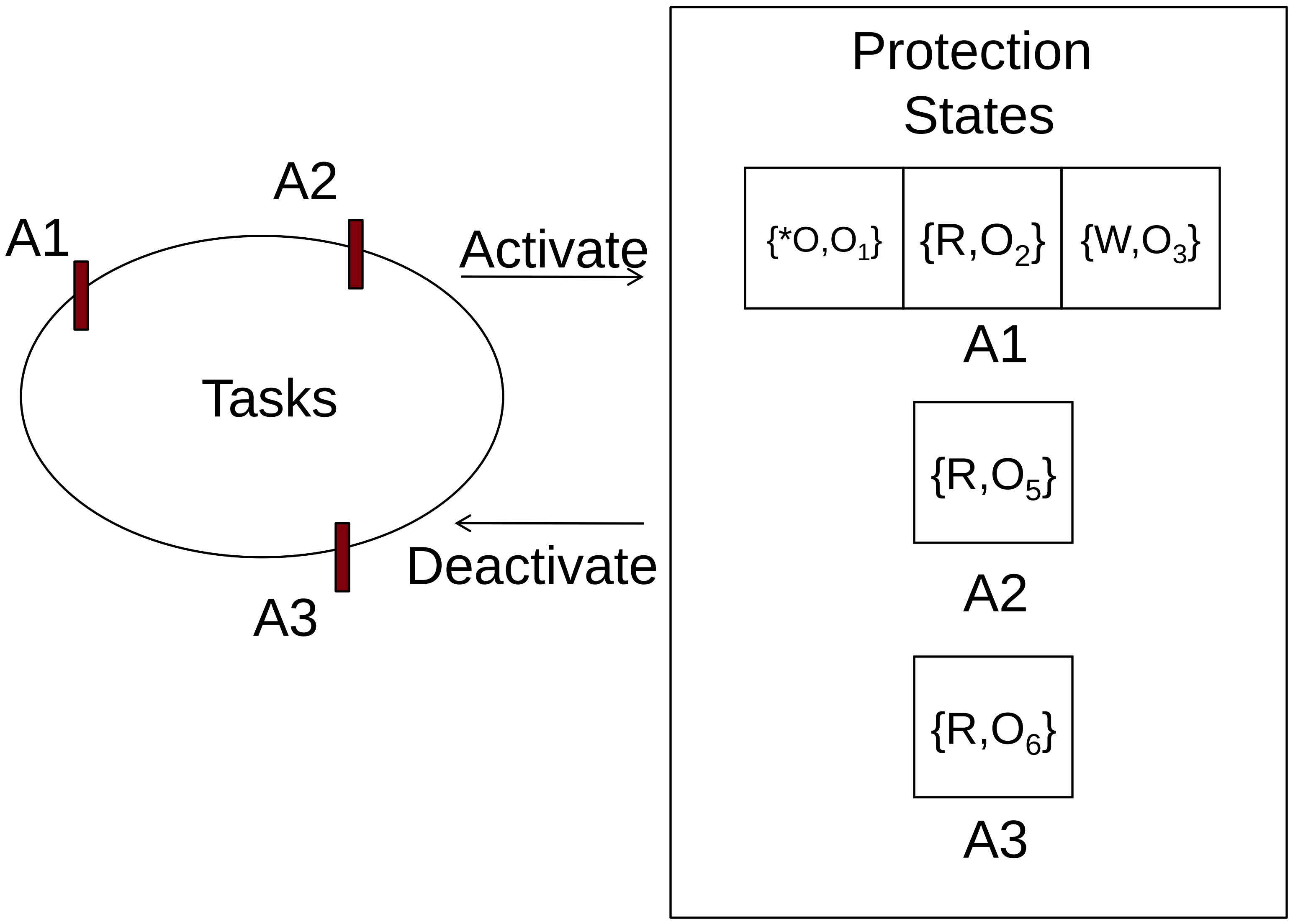
Domain-Based Access Control



Domain-Based Access Control

- Similar to Role-Based
- Authenticates user when he enters the domain

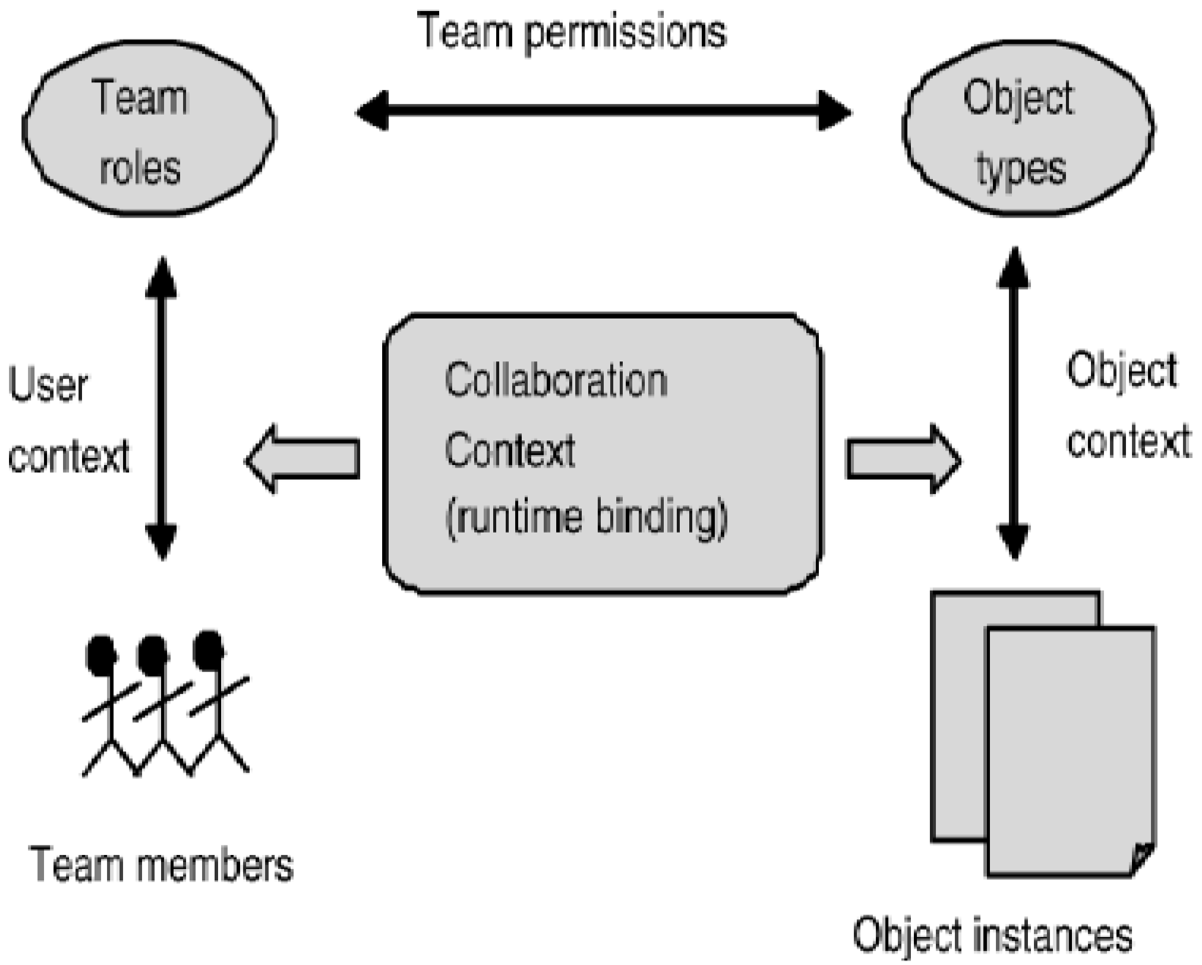
Task-Based Access Control (TBAC)



Task-Based Access Control (TBAC)

- Task-based access control (TBAC) changes subject's rights as it moves through stages of a given task
- Steps associated with protection state, with specific rights
- Each step has disjoint protection state
- Limited in context to activities, tasks and workflow progress

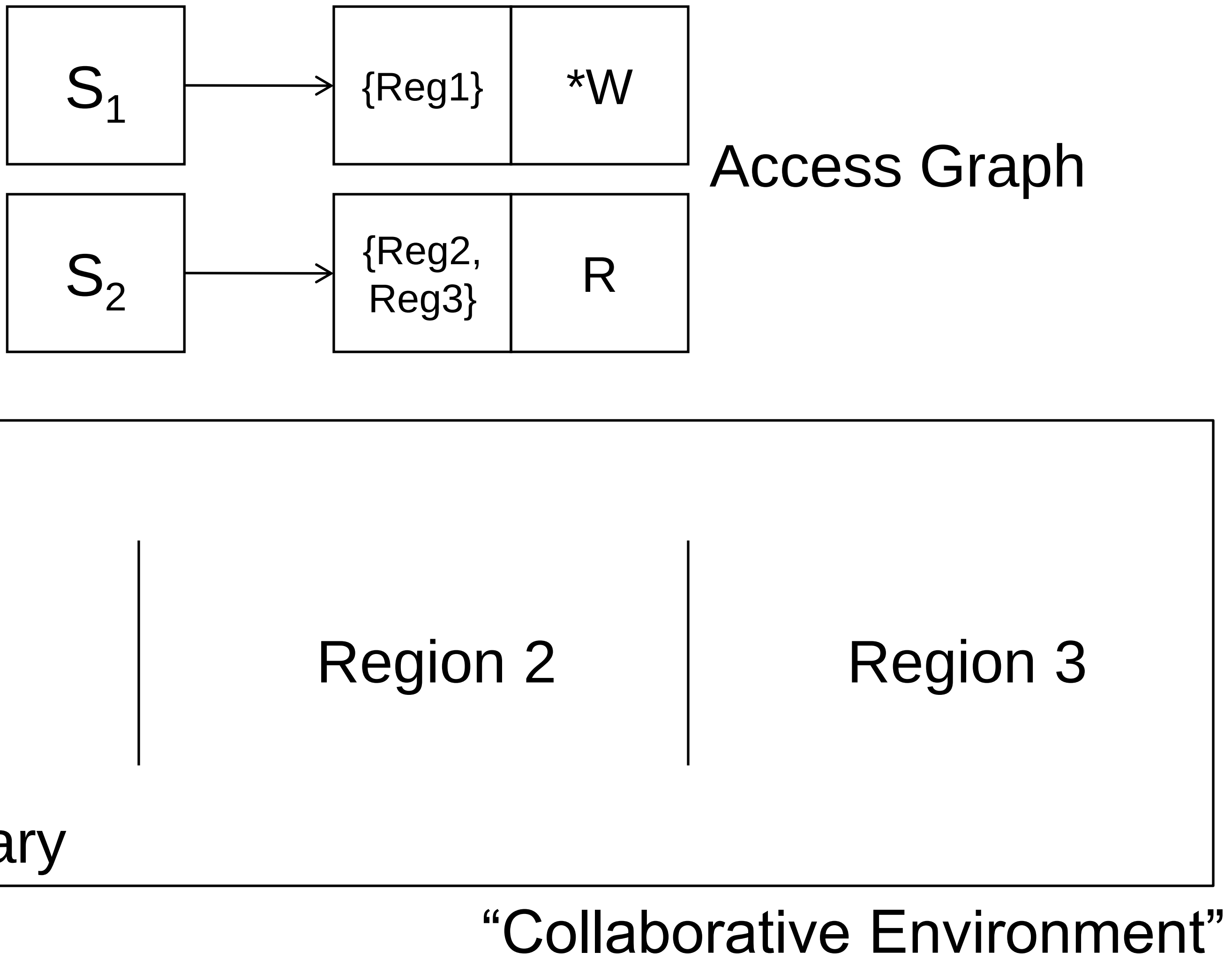
Team-Based Access Control (TMAC)



Team-Based Access Control (TMAC)

- Assigns permissions based on team designations
- Both user and object contexts (similar in concept to a combination of RBAC and Domain)
- Can be modified depending on environment context, i.e. what other subjects/teams are present
- Allows fine-grained control over individual users and objects
- Context-Based TMAC (C-TMAC)
- Incorporates context as an entity in the architecture

Spatial Access Control



Spatial Access Control

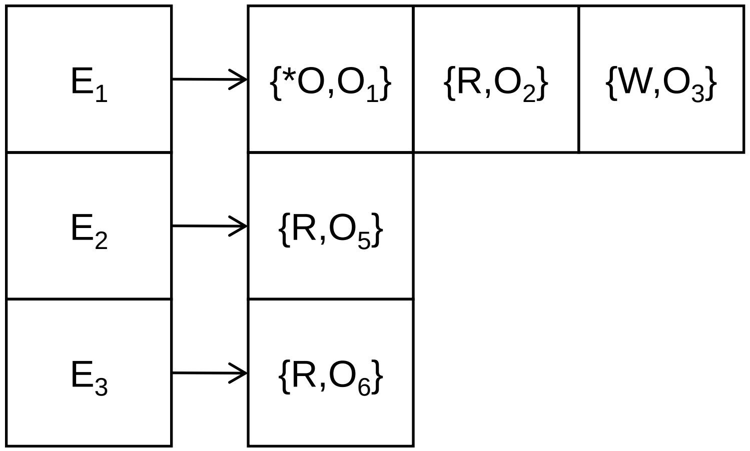
- Boundaries create regions
- Access graph gives credentials
  - Governs movement within and between regions
  - Specifies access rights within each region

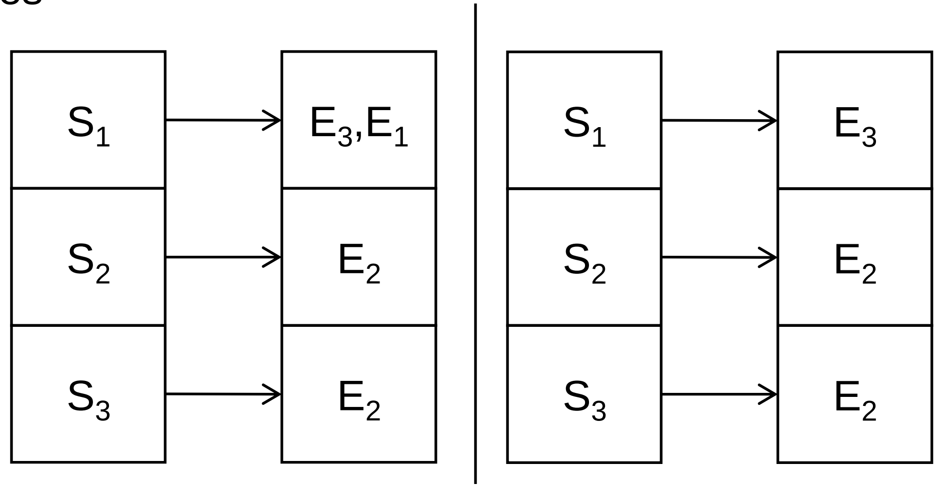
- No support for fine-grained control
- Insecure region creation possible

Context-Aware Access Control (Context-AW)

- Extends RBAC with environment roles
- Capture environment state
- Roles activated based on environment conditions at time of request

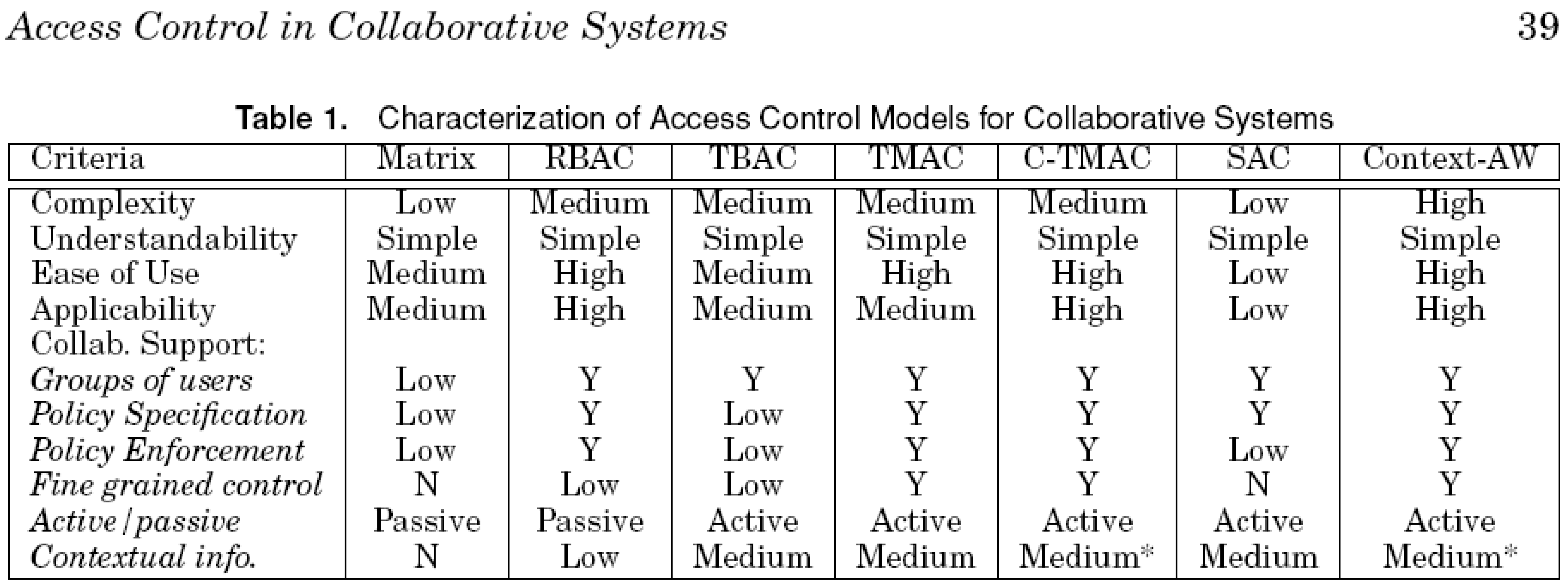
Assigning access rights to environment roles





Assigning roles to subjects based on context

Comparison of access control methods

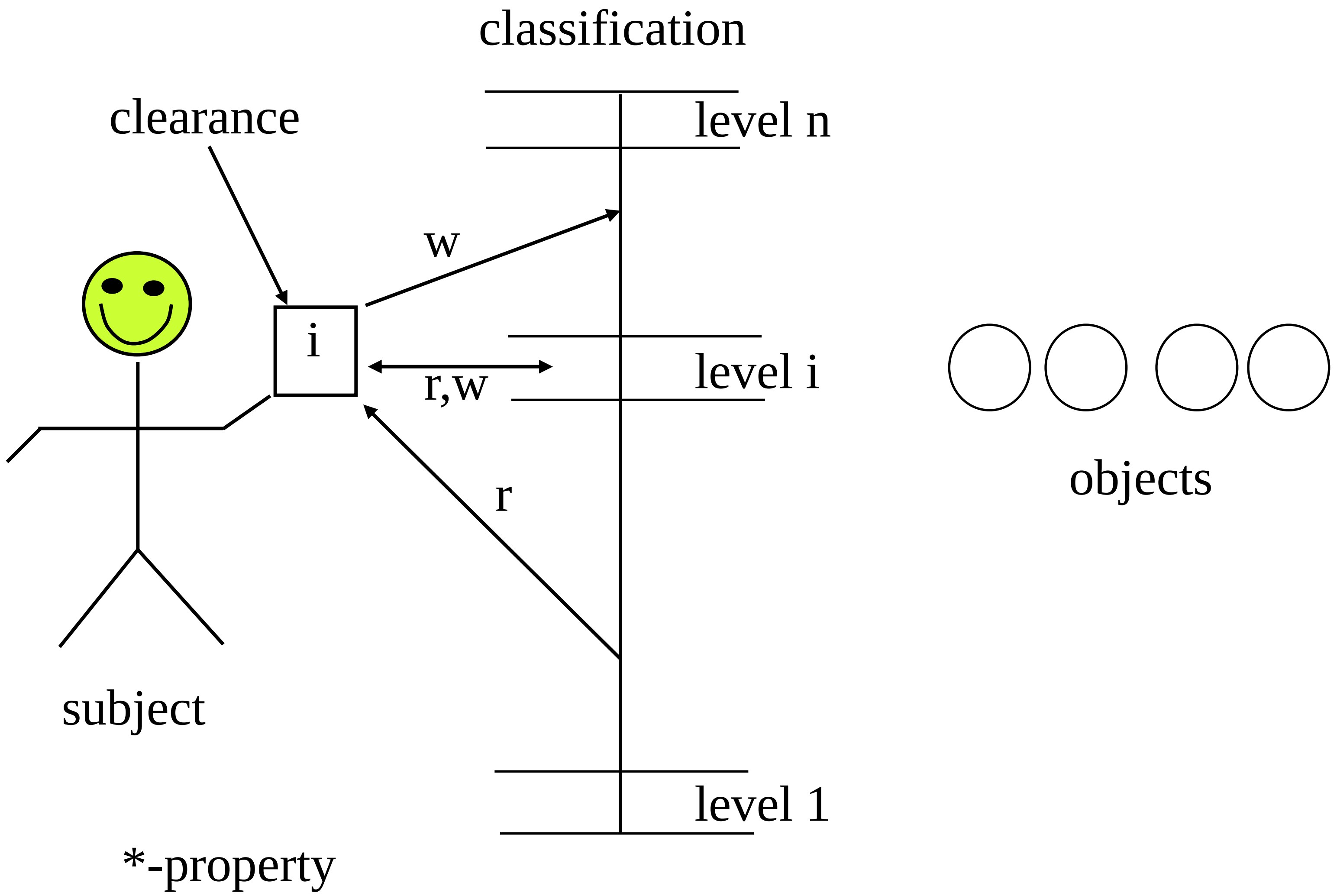


Access Control in Collaborative Systems 39

Table 1. Characterization of Access Control Models for Collaborative Systems

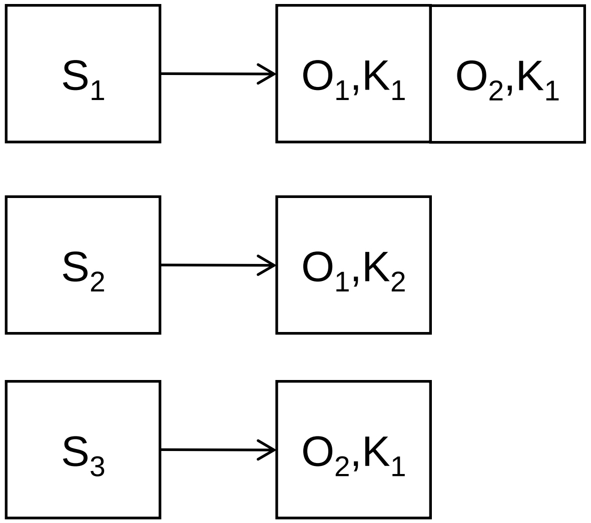
Criteria	Matrix	RBAC	TBAC	TMAC	C-TMAC	SAC	Context-AW
Complexity	Low	Medium	Medium	Medium	Medium	Low	High
Understandability	Simple	Simple	Simple	Simple	Simple	Simple	Simple
Ease of Use	Medium	High	Medium	High	High	Low	High
Applicability	Medium	High	Medium	Medium	High	Low	High
Collab. Support:							
Groups of users	Low	Y	Y	Y	Y	Y	Y
Policy Specification	Low	Y	Low	Y	Y	Y	Y
Policy Enforcement	Low	Y	Low	Y	Y	Low	Y
Fine grained control	N	Low	Low	Y	Y	N	Y
Active/passive	Passive	Passive	Active	Active	Active	Active	Active
Contextual info.	N	Low	Medium	Medium	Medium*	Medium	Medium*

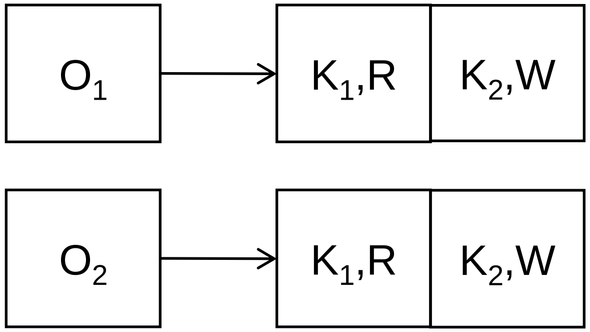
Bell-LaPadula Model



Lock-and-Key Access Control

Subjects associated with set of keys for objects





Each object associated with a {key, rights} pair

Lock-and-Key Access Control

- Like capability list
- Key values are meaningless to the subject
- Key values have no inherent rights

Memory Protection

- Hardware based on mapping
- Memory not in range of map protected
- Each subject/domain has own address space

Conclusion

- Propagation of rights between users
  - Simple in subject and system* perspectives
  - Difficult for data perspective
- Discovery of rights to a resource
  - Simple in object perspective
  - Difficult in subject and system* perspectives
- Revocation
  - Simple in object and system* perspectives
  - Difficult in subject perspective
- Reclamation
  - Simple in object and system* perspective
  - Difficult in subject perspective

Questions

?