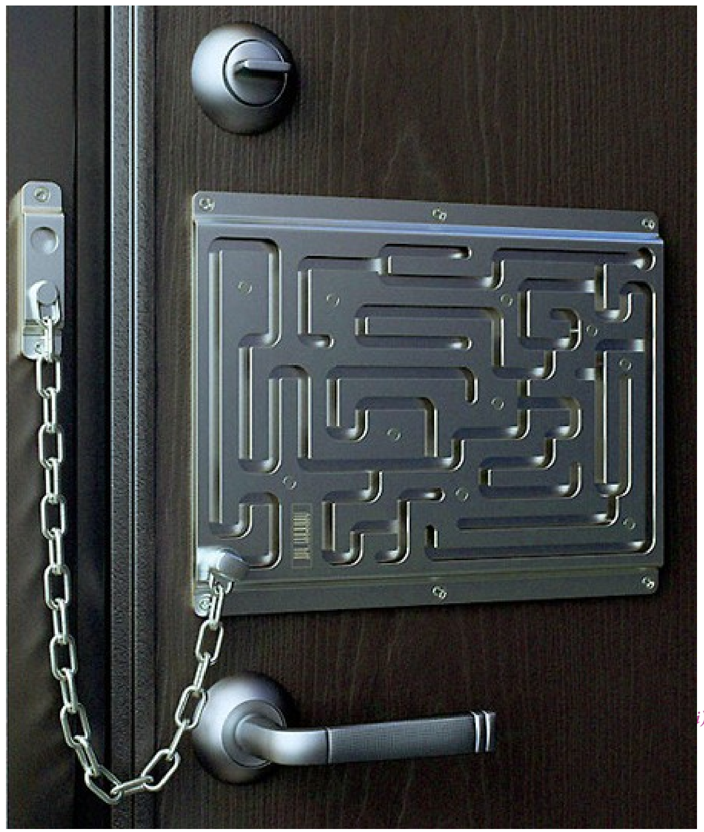
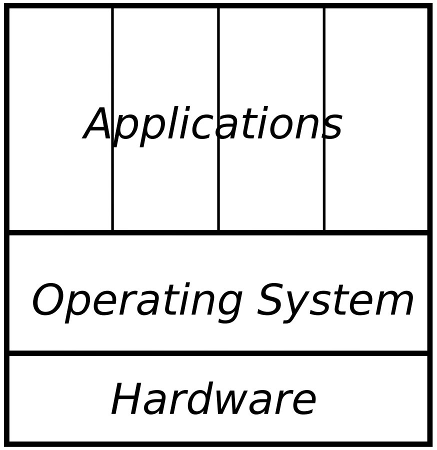
Protection Mechanisms



DTU Compute
Department of Applied Mathematics and Computer Science

Protection in Operating Systems

- OS implements the fundamental security mechanisms



- What needs to be protected
  - Memory
  - Sharable I/O devices (disks, network interfaces, …)
  - Serially reusable I/O devices (printers, tape drives, …)
  - Shared programs and sub-procedures (services)
  - Shared data (files, databases, ...)

Separation of Subjects' Access to Objects

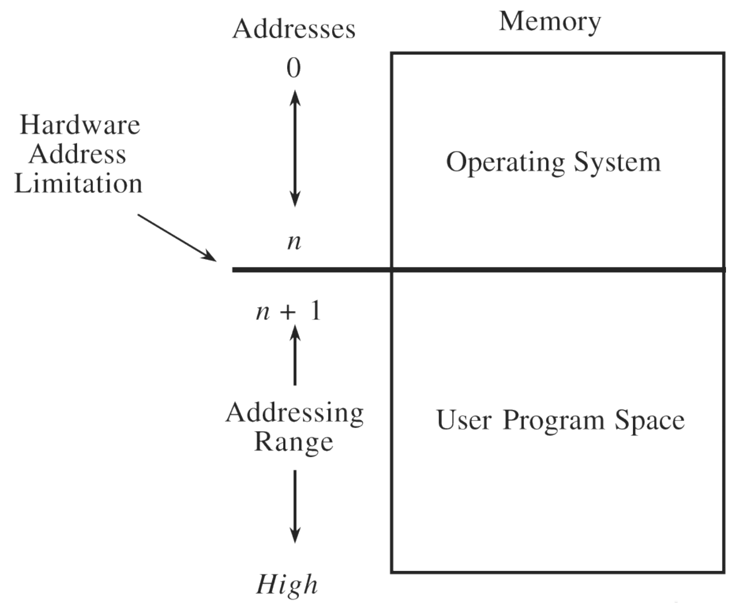
- Separation forms basis for most protection mechanisms
- Processes may have different security requirements
- Physical separation
  - Different processes use different physical objects (separate hardware)
- Temporal separation
  - Different processes are executed at different times
- Logical separation
  - OS creates illusion of physical separation
- Cryptographic separation
  - Processes conceal data and computations in a way that makes them unintelligible to outside processes
    - *Encrypt data*
    - *Some algorithms for computation on encrypted data exist*
      - Homomorphic encryption

Principles of Protection

- Do not protect
  - Appropriate when physical/temporal separation is used
- Isolate
  - Processes are completely unaware of other processes (virtual machines)
- Share all or share nothing
  - Public or private data
- Selective sharing (*share via access limitations/share by capabilities*)
  - OS enforces a policy that defines how objects can be shared by users
    - *Mandatory- /Discretionary policies*
    - *Generally implemented in a reference monitor*
- Usage control (*limit use of an object*)
  - Restricts use of objects after access has been granted
  - Typical goal for DRM systems
    - *Applications require support from hardware and OS*

Fence
Memory and Address Space Protection I

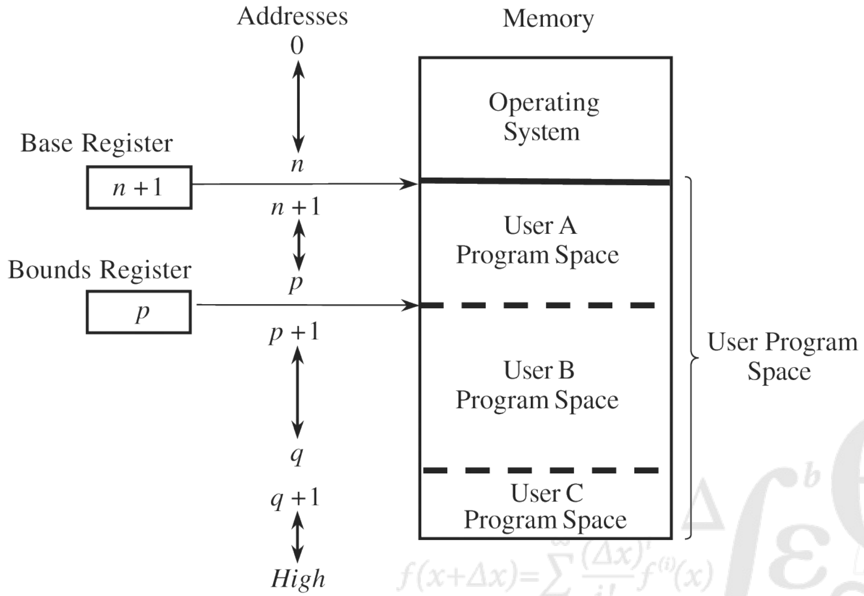
- Separation between OS and user programs



- Predefined memory address
  - Operating system resides below this address
  - Programs are loaded from this address and cannot access OS memory
  - Special *Fence Register* allows re-allocation of memory

Base/Bounds Registers
Memory and Address Space Protection II

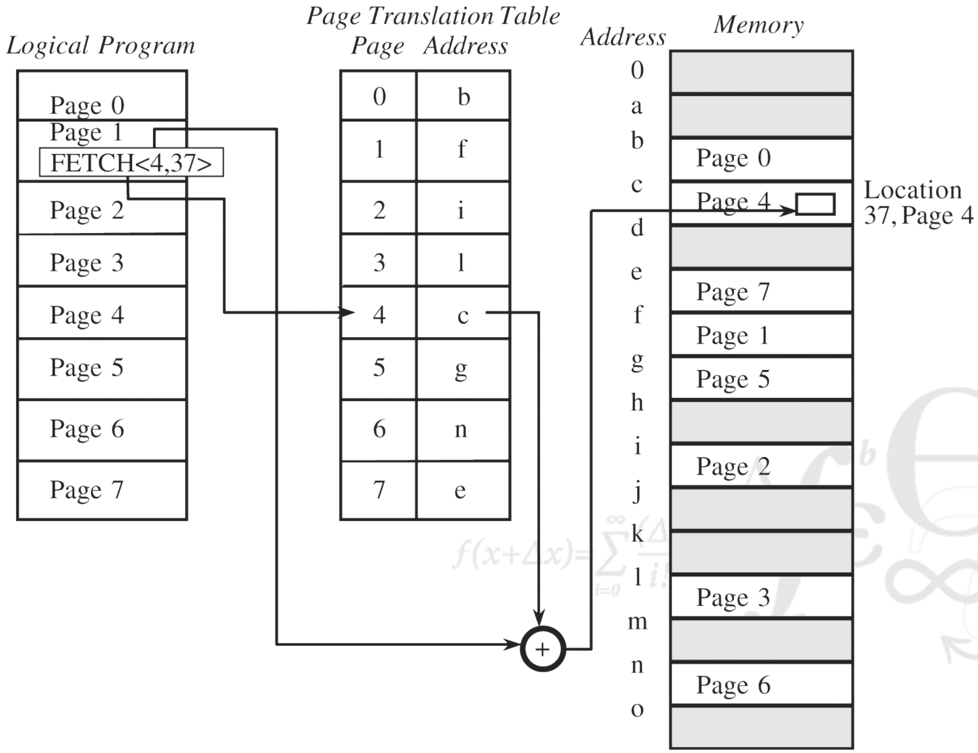
- Fence only protects in one direction (underflow)
- Base/Bounds registers protect in both directions
  - Base register corresponds to fence



- Each process has its own pair of base/bounds registers
  - Protects processes from each other
    - *One man's bounds is another man's base*

Paging
Memory and Address Space Protection III

- Variable size segments are difficult/expensive to manage
- Paging introduces fixed sized segments (page frames)
  - Typically powers of 2 between 512 and 4096 bytes



Paging II
Memory and Address Space Protection IV

- Page translation tables define the addressable memory of a process
  - Managed by the OS
    - *Prevent user processes from "mapping" OS memory into its address space*
- There is no logical structure to memory pages
  - Data with different security requirements may reside on the same page
    - *Similar to problem of false sharing*
- Security benefits of paging include:
  - Address references can be checked for protection
    - *When the relevant page is "mapped" (inserted in page translation table)*
  - Users can share data by sharing physical memory pages
    - *Access rights do not have to be the same for all users*
  - Users cannot access main memory directly
- Most current systems implement a paging architecture

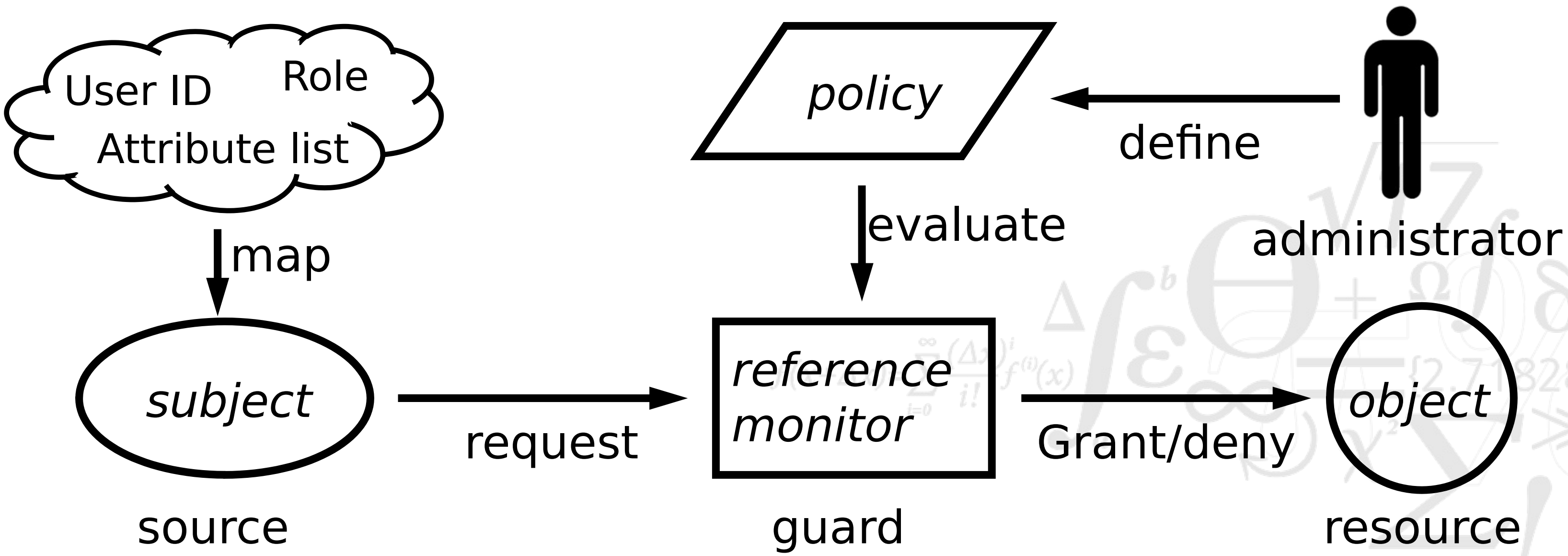
Classic view of security

Identification
→ Authentication
→ Authorization
→ Accountability

- Authentication
  - Verifies the claimed identity of subjects
- Authorization
  - Enforces access control policy
    - *Decides whether a subject has the right to perform an operation on an object*
- Accountability
  - Records security relevant events
    - *What happened? and who did what?*

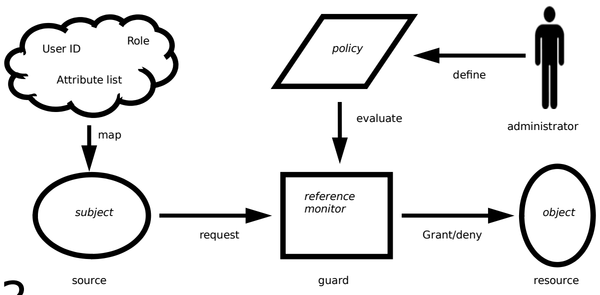
Access Control Model

- Security policy is evaluated every time an object is accessed
  - Reference Monitor mediates all access by subjects to objects
    - *Guards access to object*
    - *Interprets access control policy*
  - Subjects are active entities (users, processes)
  - Objects are passive entities (resources, e.g. files, devices, …)



Reference Monitors in Distributed Systems

- Concept developed for centralised Operating Systems
  - Policy enforced by components in the OS
  - Policy defined by local system administrators
  - Policy based on local information

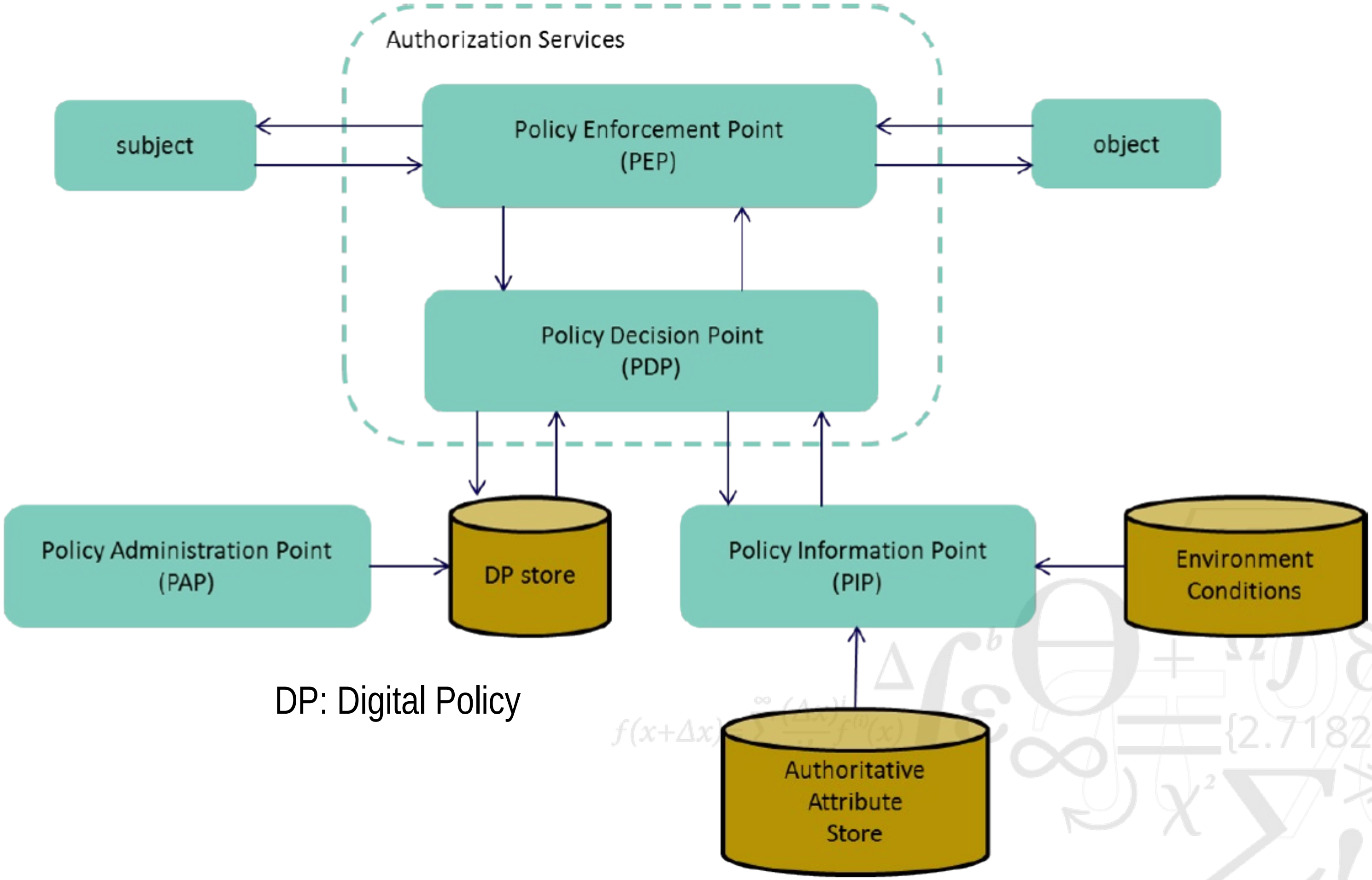


- How does this extend to distributed systems?
  - Resources hosted on different machines
    - *Possibly managed by different local administrators*
    - *Possibly belonging to different administrative domains*
  - Access Control decisions may be federation of local policies
    - *Federated identity management*
    - *Federated access control policies*
  - Distributed enforcement of policies

Access Control Architectural Elements

- PEP: Policy Enforcement Point:
  - Grants or denies access
- PDP: Policy Decision Point:
  - Decides whether access should be granted or denied
  - Uses the policies recorded in the Policy Store
- PAP: Policy Administration Point
  - Manages the Policy Store: adds, removes and modifies policies
- PIP: Policy Information Point
  - Provides the information that the PDP needs to make decisions
    - *Model parameters, roles, attributes, hierarchies, constraints*
    - *State of the environment:*
      - Examples: Time of Day, Normal Working Hours, …
      - Location of users and/or resources
      - Etc.

Access Control Architecture



DP: Digital Policy

Source: NIST Special Publication 800-162

Mapping Subjects

- Identity Based Access Control
  - Permissions are granted directly to users
  - Unique system identifier (UID) for every user
  - User identity must be verified before use (authentication)
- Role Based Access Control
  - Permissions are granted to roles
  - Users assigned one or more roles
  - User identity must be verified before role is assumed (authentication)
- Attribute Based Access Control
  - Permissions depend on user's attributes
  - Users must prove possession of attributes
    - *Attributes are often encoded in certificates*
  - Use of certificates often require user's public-key
    - *Use of public-key certificate implies authentication*
- Ultimately, users must prove identity to exercise access rights

Access Control Matrix Model

- Access Control Matrix defined by
  - Set of subjects S (active entities in the system)
  - Set of objects O (passive entities in the system)
  - Set of rights R (defines operations that subjects can do on objects)
- *A* denotes the entire access control matrix
  - Encodes the access rights of subjects to objects
  - *A* is often a sparse matrix
- *a[s,o]* denotes the element at row *s*, column *o*; *a[s,o]* $\in R$

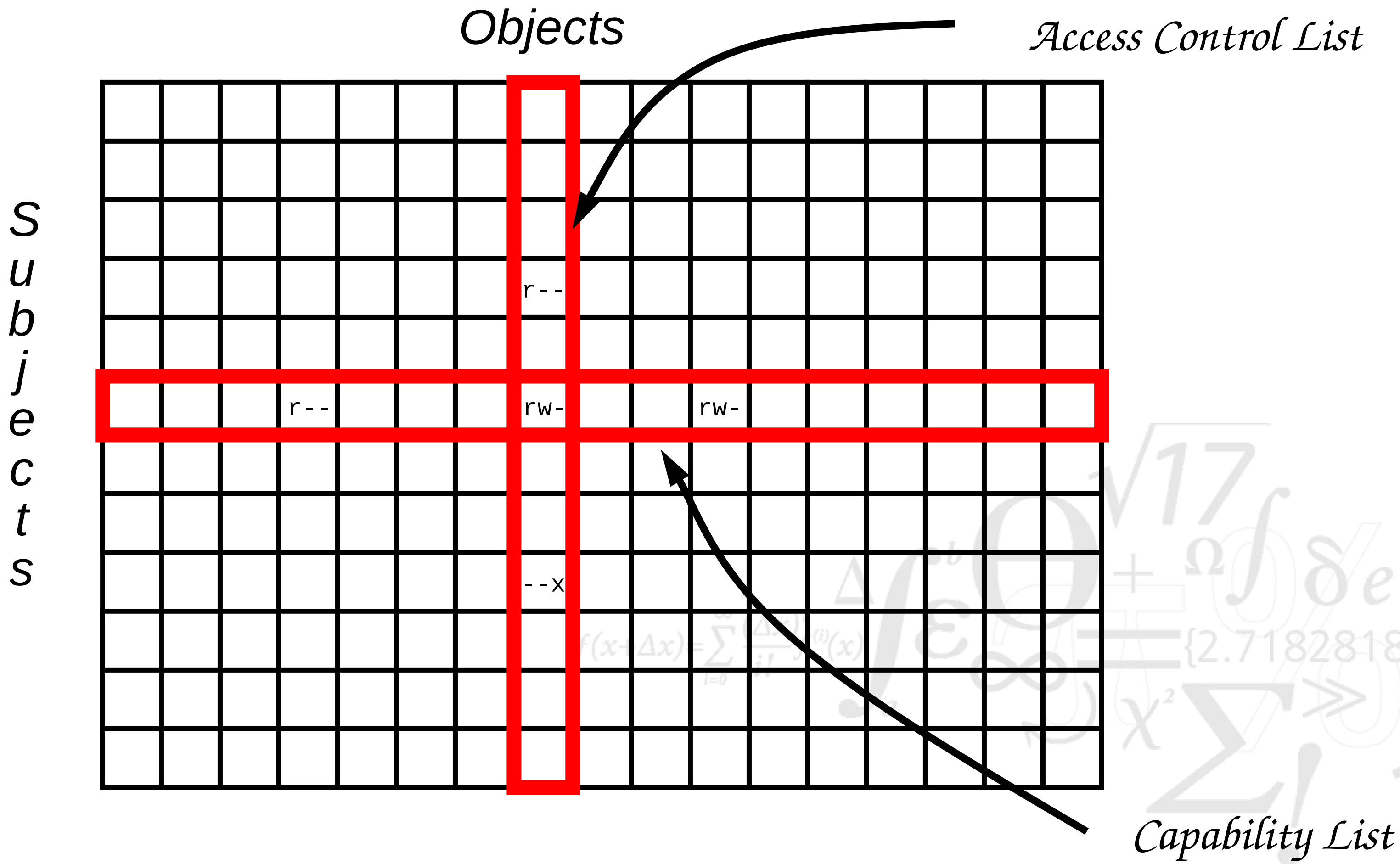
objects

subjects

	file 1	file 2	process 1	process 2
process 1	read, write, own	read	read, write, execute, own	write
process 2	append	read, own	read	read, write, execute, own

Representing the Access Control Matrix

- The Access Control Matrix is often sparse



Access Control Lists

- Associated with every object in the system
  - List of pairs: <*subject name*, *access rights*>
- Access is granted if
  - Subject name is in the list
  - Access rights include requested operation
  - Otherwise access is denied
- Some ACL systems allow special default actions (**`grant`** or **`deny`**)
  - Useful with negative access rights
    - *ACL becomes a list of people to exclude*
- Delegation is difficult
  - Requires the right to modify the ACL
- Questions about access rights
  - Easy to know who may access an object
  - Difficult to know what objects a subject may access

Capabilities

- List of capabilities is associated with every subject in the system
  - List of pairs: <*unique object identifier*, *access rights*>
- Capabilities are used to reference the object
  - Without a capability, object cannot be addressed
  - Access is granted if rights in the capability includes requested operation
- Three types of capabilities
  - Hardware capabilities
  - Segregated capabilities
  - Encrypted capabilities
- Capabilities are easy to delegate
- Questions about access rights
  - Difficult to know who may access an object (who has a capability)
  - Easy to know what objects a subject may access (and how)

Break

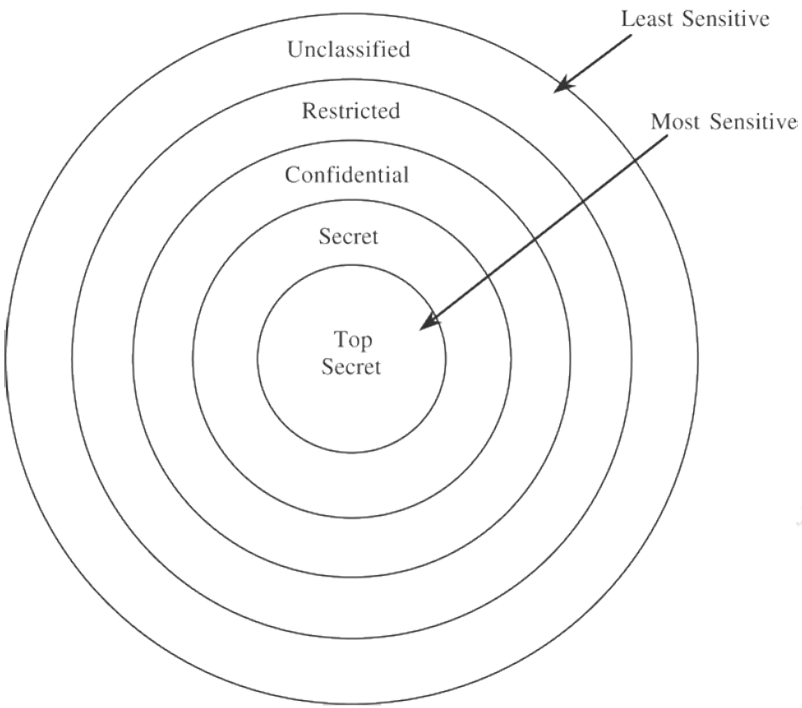

KEEP
CALM
AND
GIVE
YOURSELF
A BREAK
© 2013 KeepCalmStudio.com

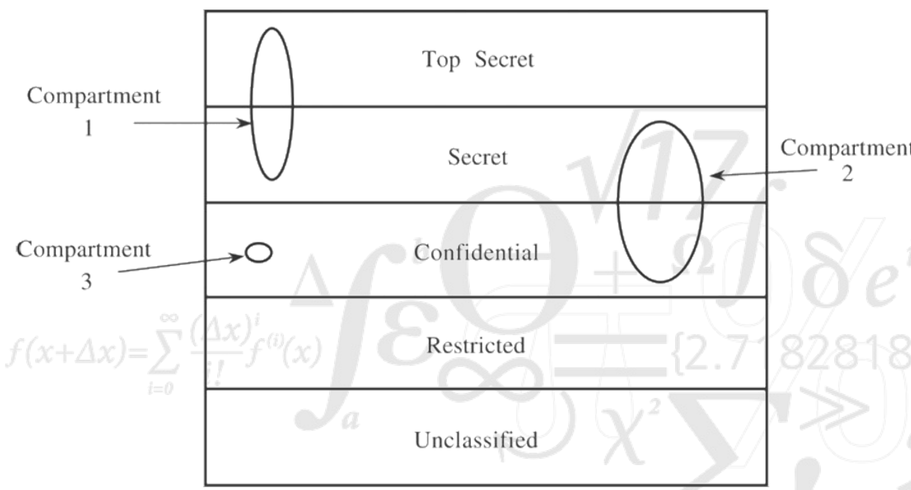
Security Policies

- Prevent disclosure or corruption of sensitive data
  - Controlled access to protected resources
  - Isolation (confinement)
  - Separation of functions (place order and sign check)
  - Well formed transactions
- Mandatory Access Control
  - System defines policies (users have little direct influence)
    - *System "owns" resources*
- Discretionary Access Control
  - Users define policies (system has little direct influence)
    - *User "owns" resources*

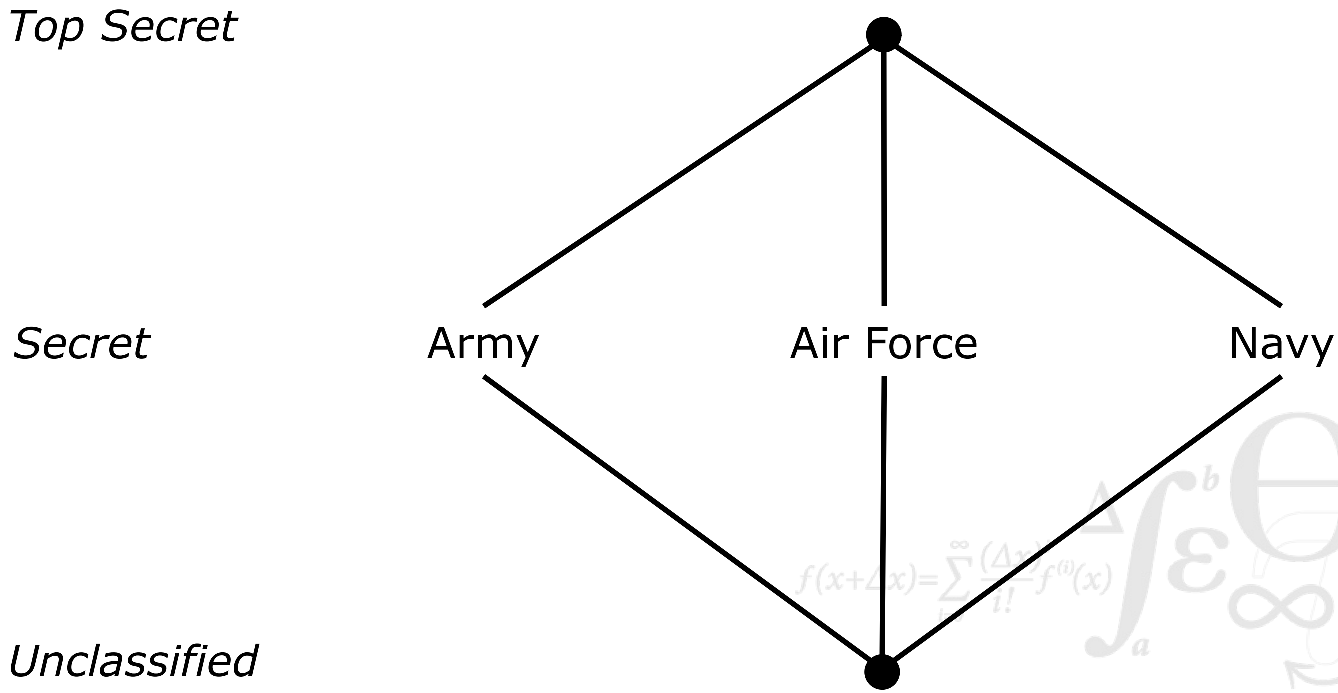
Military Access Control Policies

- Keeping military plans secret
  - Confidentiality is primary concern
    - *Need-to-know principle*
  - Traditional model based on safes and marked binders





Access Control Lattice



Bell & LaPadula
Multilevel Security

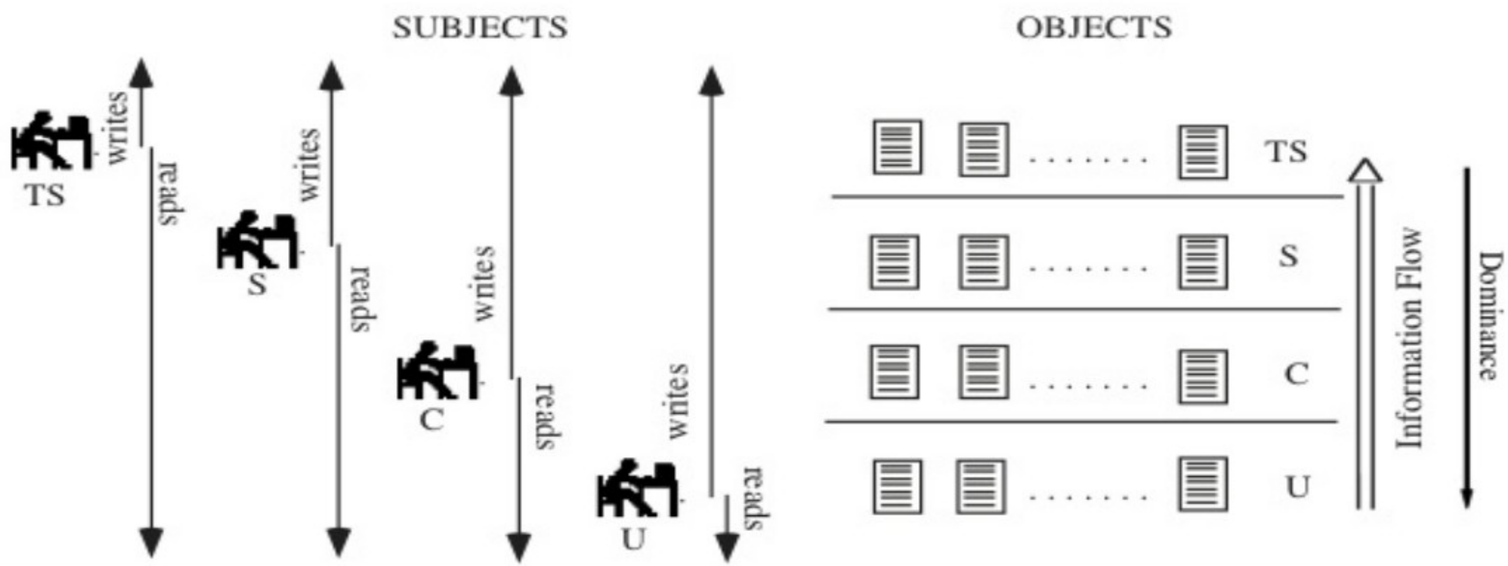
- Mandatory access control model
  - Separate users with multiple levels of privileges on the same system
  - Military system
    - *Security labels: unclassified ≤ restricted ≤ confidential ≤ secret ≤ top secret*
- Basic definitions:
  - **object:** passive entity, stores information
  - **subject:** active entity, manipulates information
  - **label:** identifies the *secrecy* classification of the object
  - **clearance:** specifies the most secret class of information available to the subject
  - **permission:** specifies the operations that the subject is allowed to invoke on the object, the model defines: *read, write, append*, and *execute* permissions

Bell & LaPadula II

- **Domination:** (relation)
  - Label (or clearance) A is said to *dominate* a label B, if a flow of information from B to A is authorized
  - A dominates B is written $A \geq B$
- Security Rules:
  - Simple security condition
    - *Subject s may only access an object o, if the clearance of s dominates the label of o*
  - The *-property
    - *Subject s may only use the content of an object o_1 to modify an object o_2, if the label of o_2 dominates the label of o_1*

*NB! A consequence of the *-property is that objects tend to rise slowly towards the highest classification*

Bell & LaPadula III



Bell & LaPadula IV

- Implementation issues:
  - Unavailability of passive objects
    - *Objects must be activated before they are accessed*
  - Tranquillity principle
    - *The label of an active object cannot be changed*
  - Initialization of objects
    - *The initial state of an object does not depend on any previously allocated resource*

- The system call open() is an example of activation

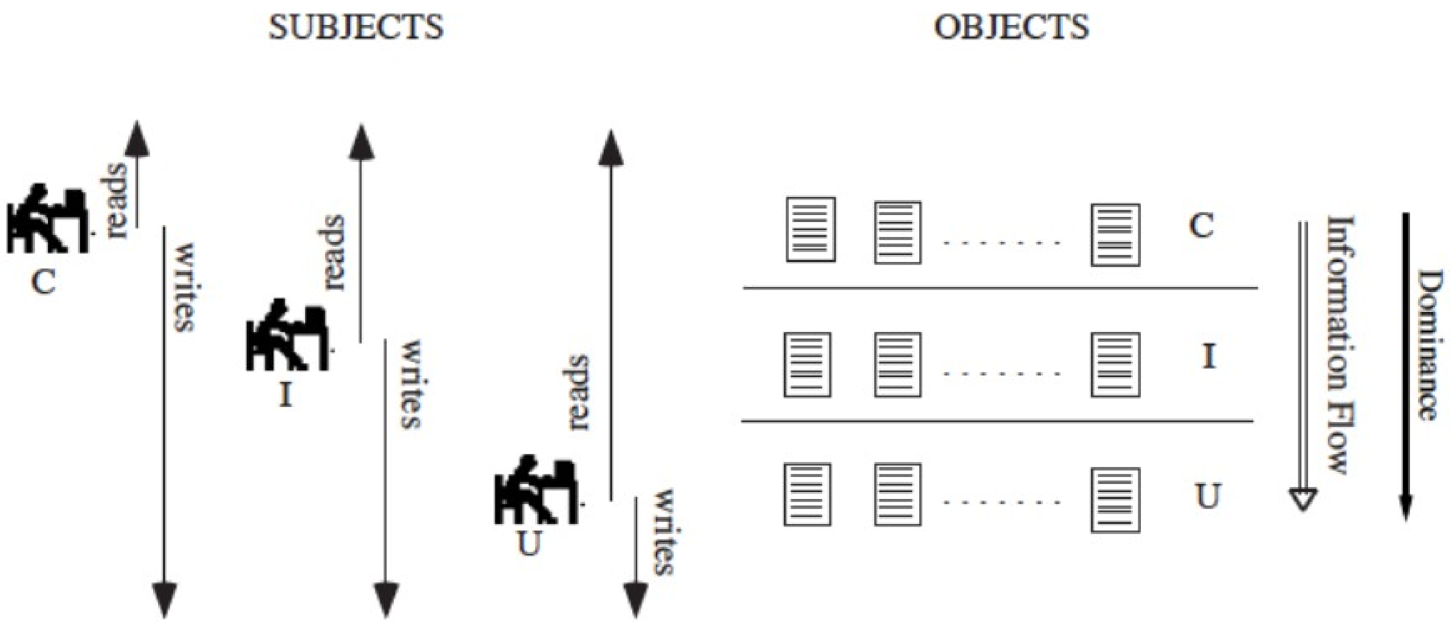
Biba Integrity Model

- In civilian systems, integrity is more important than secrecy
- Biba defines an integrity model similar to the Bell & LaPadula model
  - Introduces integrity classes
  - Prevents information from objects with low integrity to contaminate objects with a higher integrity

Integrity Rules:

1. Simple integrity: Subject *s* can only modify an object *o* if the integrity class of *s* dominates the integrity class of *o*
2. Confined integrity: Subject *s* can only read the content of an object *o* if the integrity class of *o* dominates the integrity class of *s*

Biba Integrity Model II



Role Based Access Control (RBAC)

- In many cases, authorization should be based on the function (role) of the subject in the manipulation of the object
  - Consider the following example:
    - *Anne, accountant for DTU Compute, has access to financial records*
    - *She leaves*
    - *Eva is hired as the new accountant, so she now has access to those records*
  - How are all the necessary permissions transferred from Anne to Eva?

- Examples of Functional Roles:
  - Function in a bank
    - *Teller, Clerk, Financial advisor, Branch manager, Regional manager, Bank director*
  - Function in a hospital
    - *Doctors (GP, consultant, treating doctor, …), Nurses (ward nurse, nurse, …), Hospital administrators*
  - Functions at a university
    - *Academics (teachers, research fellows, …), Non-academic staff (secretaries, system administrators, …), Students*

Common RBAC Concepts

Definitions:

- **Active role:**
  $\mathrm{AR}(s : subject)$ = (the active role for subject s)
- **Authorized roles:**
  $\mathrm{RA}(s : subject)$ = {authorized roles for subject s}
- **Authorized transactions:**
  $\mathrm{TA}(r : role)$ = {authorized transactions for role r}
- **Predicate exec:**
  $\mathrm{exec}(s : subject, t : transaction)$ = true iff s can execute t
- **Session:**
  Binds a user to a set of currently activated roles

General RBAC Rules

Rules:

1. **Role assignment:**

   $\forall s : subject, t : transaction \ (\mathrm{exec}(s,t) \Rightarrow \mathrm{AR}(s) \neq \emptyset)$

   A subject can only execute a transaction if it has selected a role

2. **Role authorization:**

   $\forall s : subject \ (\mathrm{AR}(s) \subseteq \mathrm{RA}(s))$

   A subject's active role must be authorized for the subject

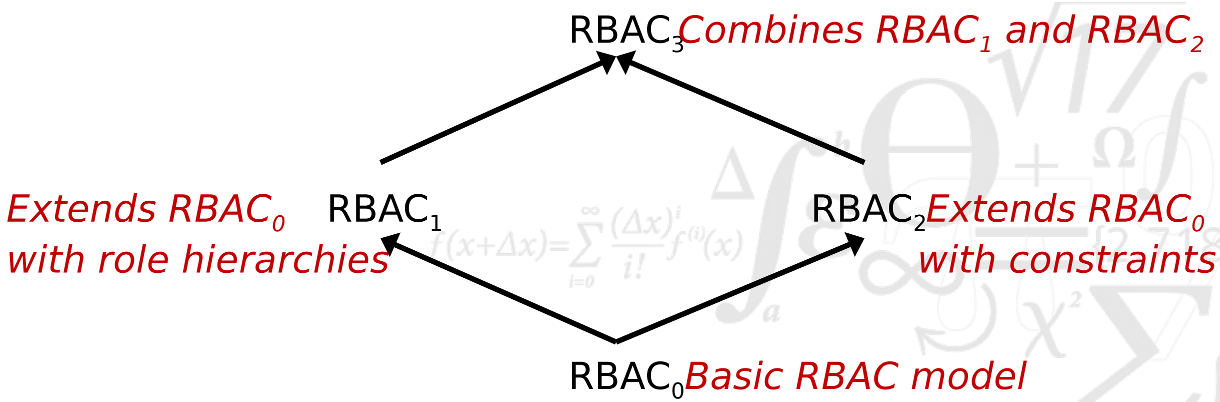
3. **Transaction authorization:**

   $\forall s : subject, t : transaction \ (\mathrm{exec}(s,t) \Rightarrow t \in \mathrm{TA}(\mathrm{AR}(s)))$

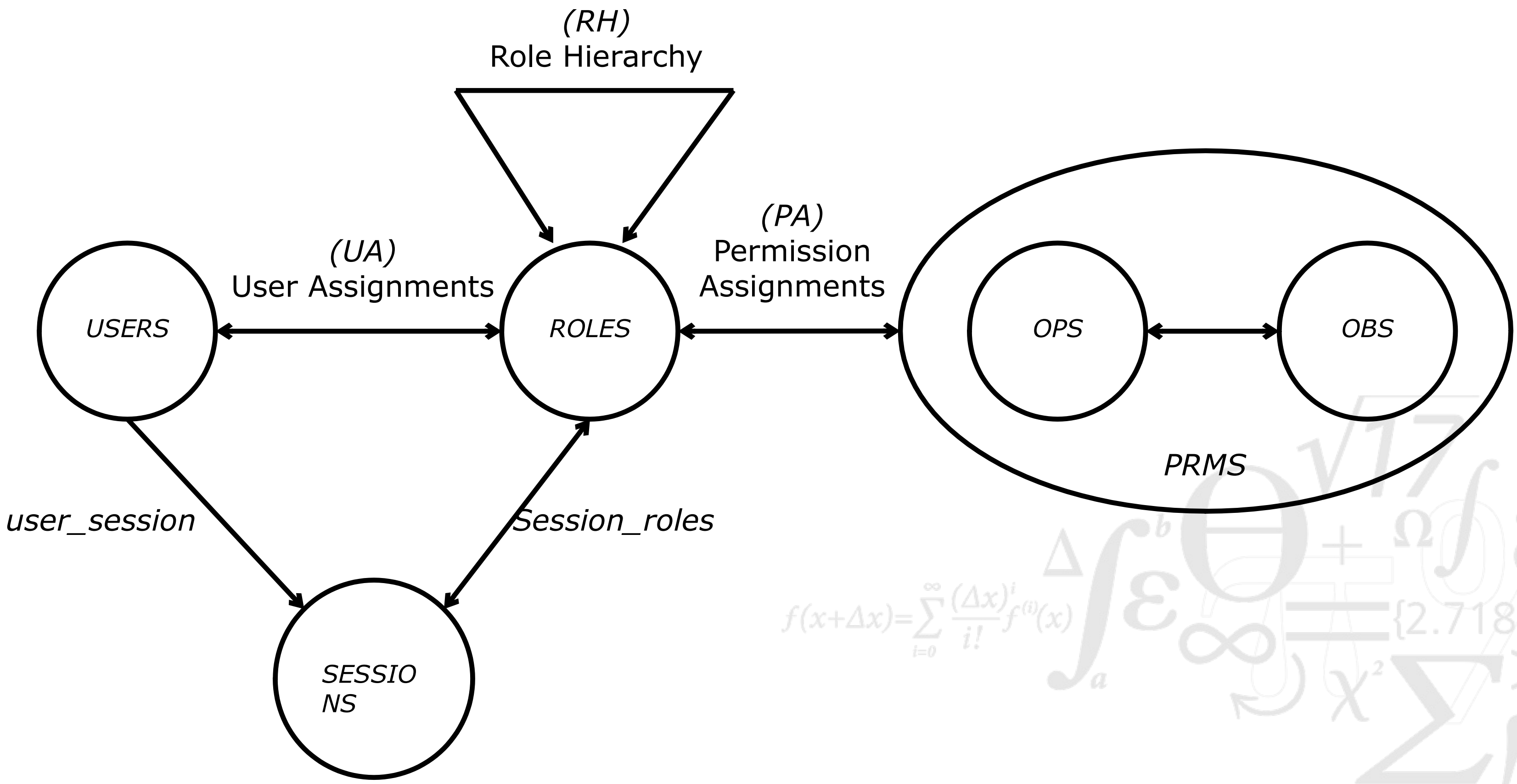
   A subject can only execute a transaction if it is authorized for its active role

RBAC96

- Role-Based Access Control was initially defined by Ferraiolo & Kuhn from NIST in 1992
- A family of related RBAC models were defined by Sandhu et al. in 1996 – this family is commonly known as RBAC96
  - RBAC96 forms the basis for most of the continued work on Role-based Access Control
- RBAC96 defines the following models:



RBAC


(RH)
Role Hierarchy
(UA)
User Assignments
(PA)
Permission
Assignments
USERS
ROLES
OPS
OBS
PRMS
user_session
Session_roles
SESSIO
NS

Attribute Based Access Control (ABAC)

- KeyNote [RFC 2704] builds on ”assertions” (credentials)
  - Blaze, Feigenbaum, Ioannidis, Keromytis; 1999
- An assertion consists of two parts
  - Identification of an agent (could be the public-key)
  - Specification of an allowed operation on a resource
  - An assertion is digitally signed by the issuer
- Assertions may be provided by:
  - The system (from the security policy)
  - The agent itself (”credential”)
- An operation is allowed if there exists an assertion that permits the operation
  - Explicit permission from the issuer
  - Implicit through other assertions from the same issuer
    - *This requires an inference engine to derive new assertions.*

Monotony in ABAC

- Assertions are Monotonous
  - Addition of an assertion never disallows an operation
  - Deletion of an assertion never allows a prohibited operation
  - Everything is prohibited unless explicitly allowed
- Significance of monotony
  - Safe to use in distributed systems
    - *Lost assertions cannot break a policy*
  - Set of assertions that combines to allow an operation constitutes a proof that the security policy is enforced
  - Clients may collect signed assertions and send them to the server
    - *Offloads work from server to clients*
  - No conflicts are possible
    - *If an operation can be allowed based on the system's assertions, the operation will be allowed*

ABAC in Practice

- Suitable for large distributed systems
  - Decentralized specification of security policies
  - Decentralized (autonomous) enforcement of security policies
- Simultaneously gives permission and the justification for allowing an operation
  - Set of assertions used to authorize the operation
- Allows dynamic evolution of security policies
  - Addition of new assertions may add new users, roles permissions or resources
- Not obvious how context may be encoded in assertions
  - This is one potential obstacle to its application in pervasive computing environments

Chinese Wall Model

- Developed to avoid conflict of interest in consultants
- The consultancy firm divides clients into business areas
- Each consultant may work for several clients
  - a priori, no limitations are assumed
  - only *one* client in each business area is allowed

consultant

Oil Company	Bank	Car Company	Fast Food
Q8	Danske Bank	Audi	Burger King
Shell	Nordea	BMW	McDonald’s
Statoil	Jyske Bank	Mercedes	Pizza Hut
OK		Renault	Sticks ‘n’ Sushi
		Volvo	

A consultant may work for any one company in each class

Authorization with JWT Tokens

- JWT (JSON Web Tokens) are widely use by web application to enforce access control in the authorization header
- Typically a JWT has the form: header.payload.signature
- The header contains two information:
  - The type of the token (JWT)
  - The algorithm used for signature (e.g., HS256)
- The payload can contain various information and can also be encrypted:
  - Registered claims (issuer, expiration time, subject, audience, ...)
  - Public claims (name, …)
  - Private claims (role, etc.)
- The last part contains the signature (usually a HMAC) of the header

Authorization with JWT Tokens

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Iktpb
mcgRWxlc3NhciBUZWxjb250YXIiLCJyb2xlIjoi
SGlnaCBLaW5nIG9mIEdvbmRvciBhbmQgQXJub3I
ifQ.mR-fnxHpcDhkHUp7-
Qlq08fYzFXFzu0wy7gjTNGR3J4

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```


PAYLOAD: DATA


```
  "sub": "1234567890",
  "name": "King Elessar Telcontar",
  "role": "High King of Gondor and Arnor"
}
```


VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  strider
) ☐ secret base64 encoded
```

Bibliography

- [BLD73] D.E. Bell, L.J. LaPadula, and United States. Air Force. Systems Command. Electronic Systems Division. Secure Computer Systems: Mathematical Foundations. National Technical Information Service, 1973.
- [Bib77] K. J. Biba. Integrity considerations for secure computer systems. Technical report, MITRE Corp., 04 1977.
- [BN89] D.F.C. Brewer and M.J. Nash. The chinese wall security policy. In Proceedings on IEEE Symposium on Security and Privacy, pages 206 –214, may 1989.
- [YT05] Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In Proceedings of the IEEE International Conference on Web Services, ICWS '05, pages 561–569, Washington, DC, USA, 2005. IEEE Computer Society.