# Authorisation Models

Vítor Francisco Fonte, vff@di.uminho.pt, University of Minho, 2025

# Authorisation Models

- Focused on confidentiality, integrity or on both properties
  - E.g. Bell-LaPadula vs. Biba, Clark-Wilson
- Static or Dynamic Policies (access rights may change in runtime)
  - E.g. Bell-LaPadula vs. Chinese Wall
- Formal or informal
  - Bell-LaPadula, Harrison-Ruzzo-Ullman vs. Clark-Wilson
- Offer a balance between confidentiality and integrity
- Don’t define methods or implementation techniques

# Bell-LaPadula (US DoD, 1975)

- BLP was the basis for standards such as TCSEC (Trusted Computer System Evaluation Criteria), or “Orange Book” (US DoD 1983).
- Confidentiality in multi-user environments (e.g. operating systems)
- Combines DAC and MAC features
- Access permissions as ACL matrices and security levels
- Multi-level security (MLS) as mandatory policies preventing information from drifting from high security levels to lower levels
  - Requires operation *sanitisation* (eg. redaction of sensitive information)

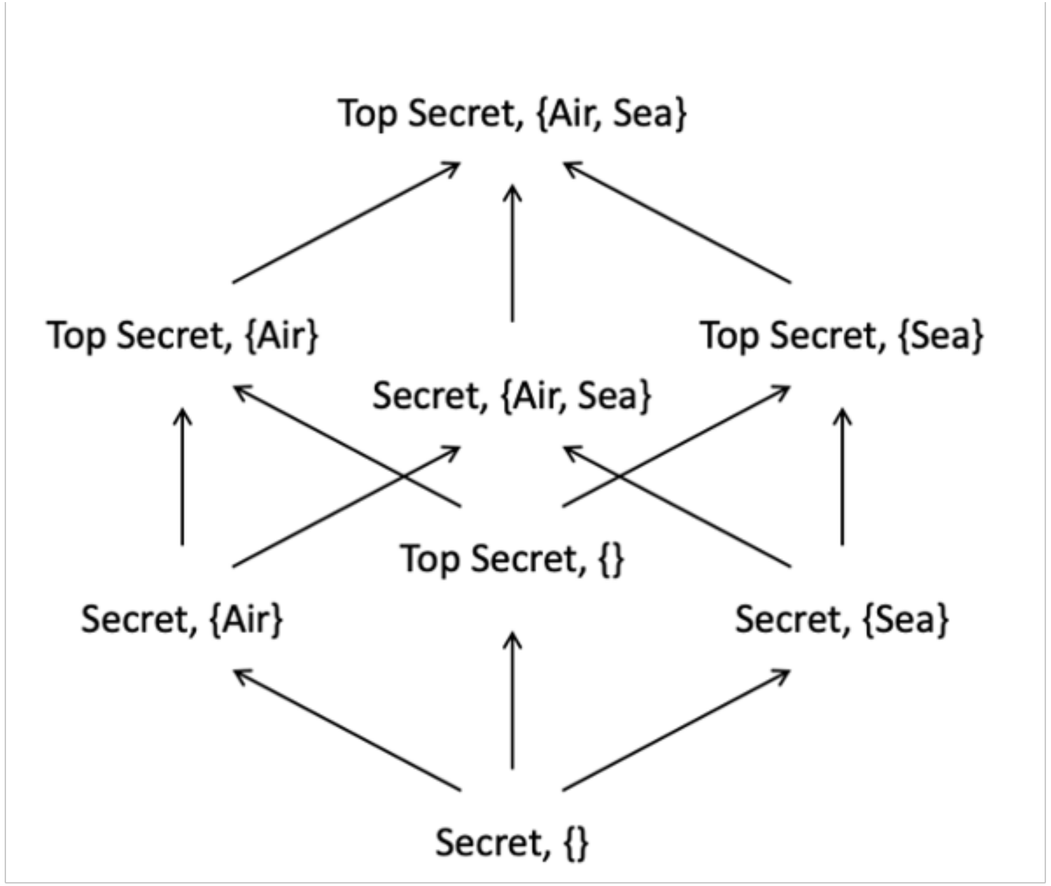
# Bell-LaPadula

- Formal state transition model for specifying computer security policies
- Based on the definition of ‘secure state’ and transitions without security breaches
- Its static nature is its main limitation (it is not possible to create or delete security labels, or change rules)

# Bell-LaPadula

## Main constructs

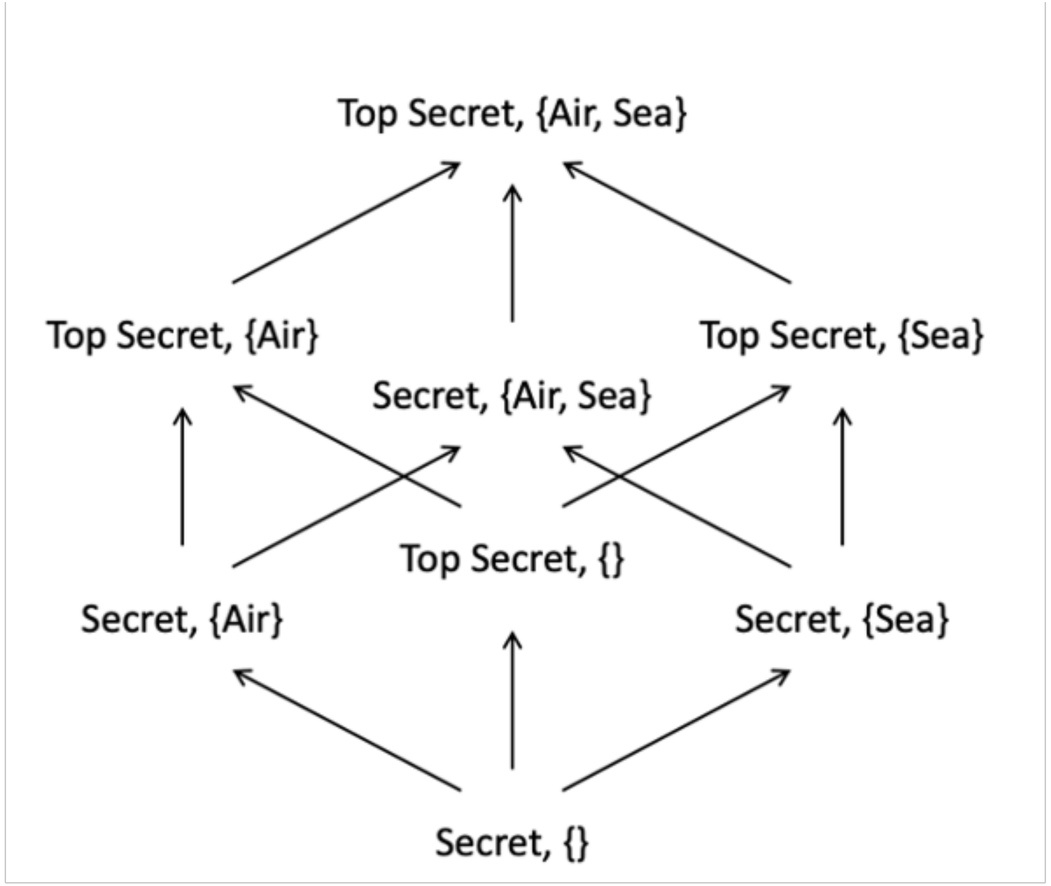
- Security label: aggregation of an authorisation level with a set of application domains
  - E.g. Top Secret, {Air, Sea}
- *Top Secret* corresponds to the authorisation level (there is an order relationship between all the levels)
- *Air* may correspond to the air domain and *Sea* may correspond to the maritime domain
- The set of all possible labels shows an order relationship that allows them to be represented in the form of a lattice



# Bell-LaPadula

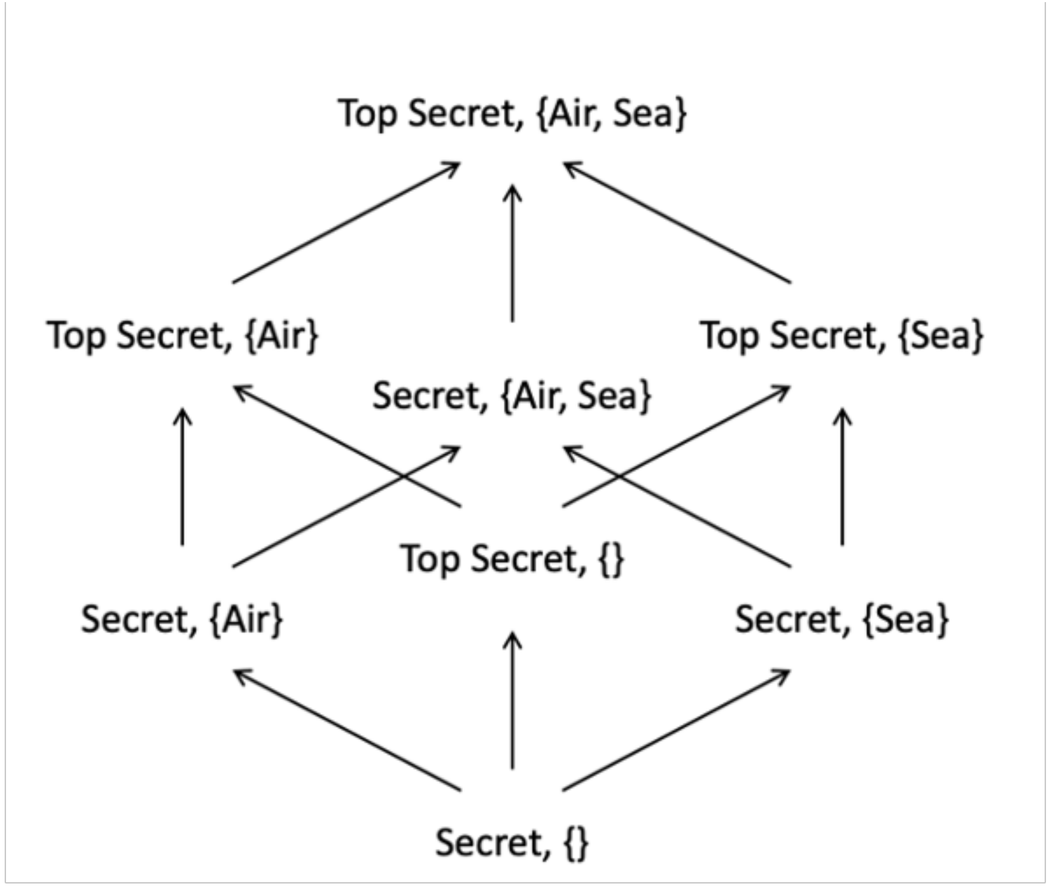
## Rules

- **Simple Security Property:** no subject at a given security level (label) can read an object with a security level that dominates it (higher) - aka *“no read up”*
- **Security Property ***: no subject at a given security level (label) can write to an object with a security level that is dominated (aka lower) - aka *“no write down”*
- **Discretionary security property**: at the same security level, authorisation is defined by an access matrix
- **Principle of tranquillity:** the classification of subjects and objects does not change during access (weak tranquillity), or during the life of the system (strong tranquillity) - hence, its *static nature*



# Bell-LaPadula

- Important theoretical model that helps to understand the threats the model is protecting against (confidentiality)
- Simple property is easy to understand and to implement
- But the * property prevents leakage but renders the model impractical
- Real-world confidentiality models have to make adaptations to this theoretical model
- By the way, SELinux can be used to implement BLP-based access control



# Bell-LaPadula

- Captures multi-level security policies for classified data
- Multics: a real-world example of implementation of BLP

# State-Machines Models

- Popular tool for modelling many aspects of computing systems
- Basis for some of the most important security models (e.g. BLP)
- Basic concepts:
  - State: representation of a system under investigation
  - State transition: function that defines the next state based on the current state and a given input

# State-Machine Models

## Example: a light switch

- States: “on” and “off”
- Input: button press
- Output: none
- State transition:
  - if current the state is “on” then the next state will be “off”
  - if current the state is “off” then the next state will be “on”

# State-Machine Models

## Example: a simplified ticket vending machine

- State:
  - Set *s* ticket and price tuples, *s*
  - Requested ticket, *t*
  - Money still to be payed, *m*
- Inputs:
  - Ticket request, *r*
  - Insertion of coin, *c*
- Outputs:
  - Ticket
  - Coin change
- State transition:
  - If ticket r was requested then set *t* to *r* and set *m* to $s[r]_{price}$
  - If coin c was inserted then
    - If c > m then
      - output *c* - *m* and *t*
      - Set *t* to *nil* and *m to 0*
    - else
      - Set *m* to *m-c*

# The Biba Model (Kenneth Biba, 1977)

- Aimed at maintaining data integrity
  - Opposite of the Bell-LaPadula model (confidentiality)
  - Ensures data is not altered by unauthorised users.
  - Information can only flow down.
- Multi-level security (MLS)
  - Subjects have security clearances and can only access objects that matches their security clearance levels.

# The Bell-LaPadula Model

## Overviwq

- State machine model capturing **confidentiality** aspects of access control
- Access permissions defined through:
  - an access control matrix, and
  - an ordered set of security levels.
- Subjects and objects are classified with those security levels.
- Information can only from downwards (from high to low security levels).

# The Biba Model

## Rules

- **Simple Integrity Property** (“No Read Up”):
  - A subject at a higher integrity level cannot read data from a lower integrity level.
- **Star Integrity Property** (“No Write Up”):
  - A subject at a lower integrity level cannot write data to a higher integrity level.
- **Integrity Invocation Property:**
  - A subject can only request services from objects of equal or higher integrity.

# The Biba Model

## Field of application

- Used in critical systems where data integrity is crucial:
  - Financial Systems: Prevents unauthorised changes to transaction records.
  - Medical Records: Ensures patient data cannot be altered by untrusted sources.
  - Military & Government Systems: Protects classified data from unauthorised modifications.

# The Biba Model

## Example Scenario

- A high-integrity user (e.g., financial analyst) cannot access untrusted data (NRD).
- A low-integrity user (e.g., customer support) cannot modify high-integrity financial records (NWU).
- Prevents corrupt or malicious data from spreading to trusted levels.

# The Biba Model

- Strengths:
  - Ensures accuracy and reliability of data.
  - Prevents malicious modifications and data corruption.
  - Simple, rule-based structure makes enforcement straightforward.
  - Works well in high-trust environments where data integrity is a priority.
- Limitations:
  - Does not focus on confidentiality, only integrity.
  - Can be restrictive, limiting how data is accessed and modified.
  - Difficult to implement in dynamic or collaborative work environments.
  - Requires strict classification of data and users, which can be complex.

# Biba vs. BLP

| Feature | Biba Model (Integrity) | Bell-LaPadula Model (Confidentiality) |
|---|---|---|
| **Goal** | Protect data integrity | Protect data secrecy |
| **No Read Rule** | No Read Down (NRD) | No Read Up (NRU) |
| **No Write Rule** | No Write Up (NWU) | No Write Down (NWD) |
| **Use Case** | Banking, critical systems | Military, government secrets |

# The Biba Security Model

## Overview

- Unlike confidentiality models, Biba prioritises integrity:
  - Ensuring data is **accurate** and **trustworthy**.
  - Preventing **unauthorised changes** or **tampering**.
- Used in systems where **data reliability is crucial** (e.g., financial, healthcare).

# The Biba Security Model

## Two main rules

1. **Simple Integrity Property (No Read Down)**
   - A subject at a higher integrity level cannot read data from a lower level.
   - Prevents exposure to untrusted data.
2. **Star Integrity Property (No Write Up)**
   - A subject at a lower integrity level cannot write to a higher integrity level.
   - Prevents corruption of trusted data.

# The Biba Security Model

## An example

- A **high-integrity** government database should not accept modifications from a low-integrity user.
- A **financial system** should not allow **low-trust applications** to modify critical records.
- Helps ensure **data accuracy** and **reliability**.

# The Biba Security Model

## Comparison to the BLP model

| Feature | Biba Model (Integrity) | Bell-LaPadula Model (Confidentiality) |
|---|---|---|
| Focus | Data integrity | Data confidentiality |
| No Read Down | ✅ (Protects integrity) | ❌ |
| No Write Up | ✅ (Protects integrity) | ❌ |
| No Read Up | ❌ | ✅ (Protects secrecy) |
| No Write Down | ❌ | ✅ (Protects secrecy) |

# The Biba Security Model

## Implementation in Real-World Systems

- Mandatory Access Control (MAC) environments.
- Used in secure operating systems (e.g., SELinux, Trusted Solaris).
- Implemented in financial and healthcare systems to protect critical data.

# The Biba Security Model

## Strengths and limitations

- **Strengths:**
  - Ensures **high-trust data remains unaltered**.
  - Prevents **data contamination** from untrusted sources.
  - Supports **compliance with regulatory standards**.
- **Limitations:**
  - **Restrictive**, limits **data sharing**.
  - **Not suitable for all environments**, especially those prioritising confidentiality.

# The Chinese Wall Security Model

## Overview

- Security framework designed to **prevent conflicts of interest**.
- Proposed by David Brewer and Michael Nash in 1989.
- Primarily used in **financial institutions**, **law firms**, and **consulting firms**.
- Ensures users **cannot access conflicting sets of sensitive data**.

# The Chinese Wall Security Model

## Core concept

- The model groups sensitive data into **conflict of interest classes (COIs)**.
- Users can **access data of a company but not from a competitor** in a COI**.**
- Protects **confidential business information** and maintains **ethical integrity**.

# The Chinese Wall Security Model

## Rules

1. **Read Access Restriction:**
    - A user can only access **one company's data** within a conflict class.
    - If a user reads data from **Company A**, they cannot access data from **Company B** in the same group.
2. **Write Access Restriction:**
    - Users can only **modify data** if they have read access to **only one company** in that conflict class.
    - Prevents data leakage between competitors.

# The Chinese Wall Security Model

## Example: a financial analyst

- Analyst accesses **Company A's financial reports**.
- They are now **restricted from accessing Company B's financial reports,** if A and B are competitors.
- They can still access **unrelated companies** in other industries.

# The Chinese Wall Security Model

## Strengths and limitations

- **Strengths:**
  - Prevents conflicts of interest in industries like banking, consulting, and law.
  - Protects sensitive business data from cross-contamination.
  - Balances security and accessibility while allowing work in multiple sectors.
  - Supports ethical decision-making by restricting access to competing firms.
- **Limitations:**
  - Complex to manage in large organisations with multiple conflict classes.
  - Potential productivity restrictions as users may be blocked from necessary data.
  - Difficult to enforce dynamically in real-time collaborative environments.
  - Not suitable for all security scenarios, particularly in government and military applications.

# The Chinese Wall Security Model

## Implementation in real-world systems

- Financial firms & investment banks:
  - Prevents advisors from accessing competing clients' data.
- Law firms:
  - Ensures lawyers handling one company's case cannot access competitor's legal data.
- Corporate mergers & acquisitions:
  - Restricts access to prevent insider trading risks.

# The Chinese Wall Security Model

## Comparison to BLP and Bina

| Feature | Chinese Wall Model | Bell-LaPadula (Confidentiality) | Biba (Integrity) |
|---|---|---|---|
| **Focus** | Conflict of interest prevention | Confidentiality | Data Integrity |
| **Access Control** | Based on **conflict classes** | Mandatory security levels | Data trust levels |
| **Best for** | Finance, law, consulting | Government, military | Banking, healthcare |
| **Restrictive?** | Yes (for conflicts) | Yes (for secrecy) | Yes (for trustworthiness) |

# The Chinese Wall Security Model

## Takeways

- The Chinese Wall Model is **well suited for preventing conflicts of interest**.
- Ensures users cannot **access competing firms'** sensitive data.
- Used in **finance**, **law**, and **corporate security** to maintain **ethical boundaries**.
- Effective when **combined with other security models** for a comprehensive security approach.

# The Clark-Wilson Security Model

## Overview

- Security framework designed to **ensure data integrity**.
- Developed by David D. Clark and David R. Wilson in 1987.
- Focuses on preventing **unauthorised or improper modifications to data**.
- Used in **financial**, **healthcare**, and **enterprise systems**.

# The Clark-Wilson Security Model

## Key principles

1. **Data Integrity**
   - Ensuring authorised and well-formed transactions.
2. **Separation of Duties**
   - Different users have different roles in handling data.
3. **Access Control through Transactions**
   - Users cannot modify data directly; they must use authorised programs.

# The Clark-Wilson Security Model

## Core components

1. **Constrained Data Items (CDIs):**
    - Protected, high-integrity data.
2. **Unconstrained Data Items (UDIs):**
    - Unprotected, general data.
3. **Transformation Procedures (TPs):**
    - Controlled processes that modify CDIs.
4. **Integrity Verification Procedures (IVPs):**
    - Audits to ensure integrity rules are followed.

# The Clark-Wilson Security Model

## Rules

1. **Users cannot modify CDIs directly:** they must go through TPs.
2. **Only authorised TPs can operate on CDIs.**
3. **Separation of duties:** No single user has total control over data processes.
4. **IVPs check and enforce integrity periodically.**

# The Clark-Wilson Security Model

## Rules

- A user **cannot modify an account balance directly**.
- Users must use an **authorised transaction program (TP)**, such as a funds transfer system.
- The system ensures **valid transactions** and **prevents fraud**.
- Auditors use **IVPs** to verify that transactions follow security rules.

# The Clark-Wilson Security Model

## Strengths and limitations

- **Strengths:**
  - Strong data integrity enforcement
  - Prevents unauthorised data manipulation
  - Provides built-in auditing and accountability
  - Works well in structured business environments
- **Limitations:**
  - Requires strict process enforcement
  - Can be complex to implement in dynamic environments
  - Relies on well-defined roles and access control

# The Clark-Wilson Security Model

## Comparison with other security models

| Feature | Clark-Wilson Model | Bell-LaPadula | Biba |
|---|---|---|---|
| **Focus** | Data integrity & authorized processes | Data confidentiality | Data integrity |
| **Direct data modification?** | No – must use TPs | Yes (if permitted) | Yes (with restrictions) |
| **Best for** | Finance, healthcare, corporate systems | Government, military | Banking, critical systems |
| **Audit & verification** | ✅ Required | ❌ Not required | ❌ Not required |

# The Clark-Wilson Security Model

## Takeaways

- The **Clark-Wilson Mode**l ensures **data integrity** through controlled transactions.
- **Separation of duties** prevents **fraud and unauthorised modifications**.
- Used in **financial**, **healthcare**, and **corporate security**.
- Effective when **combined with other security models** for a comprehensive security approach.

# The Harrison-Ruzzo-Ullman (HRU) Security Model

## Overview

- Security framework that defines **access control rules** in computer systems.
- Developed by Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman in 1976.
- **Expands on the Access Control Matrix Model** to include **dynamic changes** to permissions.
- Used in **operating systems**, **databases**, and **role-based access control** (RBAC) systems.

# The Harrison-Ruzzo-Ullman (HRU) Security Model

## Access Control Matrix (ACM)

- **HRU is based on the ACM**, which represents:
  - **Subjects** (users, processes)
  - **Objects** (files, memory, databases)
  - **Permissions** (read, write, execute, own)
- Each subject-object pair has s**pecific rights** defined in the matrix.

# The Harrison-Ruzzo-Ullman (HRU) Security Model

## Key features

- Allows changes to access rights dynamically.
- Supports creation and deletion of subjects and objects.
- Defines rules for transferring privileges between users.
- Can be analysed to determine security vulnerabilities.

# The Harrison-Ruzzo-Ullman (HRU) Security Model

## Operations

- **Create Object**: Introduces a new object in the system.
- **Create Subject**: Introduces a new subject (user/process).
- **Delete Object**: Removes an object from the system.
- **Delete Subject**: Removes a subject from the system.
- **Enter Right**: Assigns a new access right to a subject-object pair.
- **Delete Right**: Revokes an existing access right.

# The Harrison-Ruzzo-Ullman (HRU) Security Model

## Example: a multi-user file system

- User A creates a file F.
- The system assigns read & write permissions to A.
- User B requests access to F.
- System checks HRU rules before allowing A to grant read access to B.
- If a security policy allows it, A can transfer the permission to B.

# The Harrison-Ruzzo-Ullman (HRU) Security Model

## Strengths and limitations

- **Strengths**
  - Supports dynamic access control modifications.
  - More flexible than static access control models.
  - Helps detect potential security vulnerabilities.
- **Limitations**
  - Computational complexity: hard to determine if a system can be kept secure.
  - May require extra mechanisms to prevent unauthorised privilege escalation.
  - Less practical for modern systems without additional enhancements.

# The Harrison-Ruzzo-Ullman (HRU) Security Model

## Comparison with other security models

| Feature | HRU Model | Bell-LaPadula | Biba |
|---|---|---|---|
| **Focus** | Access control & dynamic rights | Data confidentiality | Data integrity |
| **Supports dynamic permissions?** | ✅ Yes | ❌ No | ❌ No |
| **Best for** | Operating systems, databases | Government, military | Banking, critical systems |
| **Handles subject/object creation?** | ✅ Yes | ❌ No | ❌ No |

# The Harrison-Ruzzo-Ullman (HRU) Security Model

## Takeaways

- The Harrison-Ruzzo-Ullman Model extends access control by allowing dynamic changes.
- Defines six primitive operations for managing permissions.
- Useful for operating systems and databases with evolving access needs.
- Helps identify potential security risks in complex systems.

# The Lipner’ Security Model, 1982

- Lipner's model attempts to adapt military-style security (Bell-LaPadula) to commercial environments by ensuring **confidentiality**, **integrity** and **accountability**.

- To achieve this, Lipner combines the Bell-LaPadula model with Biba’s integrity model, using two label structures to enforce access control.

# The Lipner’ Security Model

## Two-Label System: Confidentiality + Integrity

- **Confidentiality Levels (MAC - Bell-LaPadula)**
  - Information is classified into different security levels.
  - Follows the "No Read Up, No Write Down" rules of BLP
  - A user can only read information at their level or below (no read up).
  - A user can only write information at their level or above (no write down).
- **Integrity Levels (MAC - Biba Model)**
  - Follows the "No Read Down, No Write Up" rules of Biba:
  - A user can only read data from their level or above (no read down).
  - A user can only write to their level or below (no write up).

# The Lipner' Security Model

## Role-Based Access Control with mutual exclusive roles

**System Management:**

- System Manager (S) – Full control over security policies and system settings.
- Application Development (A) – Develops applications but cannot modify system settings.

**Application and Data Integrity:**

- Application Production (P) – Maintains production applications but cannot develop them.
- Application Use (U) – Uses applications but cannot modify them.

**Operational Integrity:**

- Operations (O) – Handles sensitive tasks like data input and report generation.
- Audit (A) – Reviews and logs access for security compliance.

**Control and Accountability:**

- Control (C) – Verifies compliance with security policies.
- System Low (L) – Represents the lowest clearance level (public users).

# The Lipner' Security Model

## Separation of duties and prevention of tampering

- Enforces strict role separations to prevent conflicts of interest.
  - Example: A user with System Management (S) role cannot perform auditing (A) functions.
  - No single user has complete control over sensitive operations.
- Integrity model (Biba) prevents unauthorised modifications to security logs.
  - Audit records cannot be erased or altered by a malicious insider.

# The Lipner' Security Model

## Accountability

- The dedicated Audit (A) role is responsible for reviewing security logs.
- All user actions (access attempts, modifications, and security policy violations, etc) are recorded in an audit log.
- The logs help detect suspicious activities (e.g., an unauthorised user trying to modify critical system files).
- Audit trails allow administrators to trace actions back to specific users.

# The Lipner' Security Model

## Accountability

- All actions are traceable, auditable, and performed by authorised users:
  - Role-based controls
  - Mandatory access policies
  - Logging
  - Authentication
  - Integrity protection