4.3. MAC

- **MAC** specifies the access that subjects have to access objects based on subjects and objects classification.
  - A subjects has access to the object for which it has a clearance.
- This type of security has also been referred to as ***MultiLevel Security (MLS)***
- Database systems that satisfy multilevel security properties are called multilevel secure database management systems (MLS/DBMSs)
- Many of the MLS/DBMSs have been designed based on the Bell-LaPadula (BLP) model.

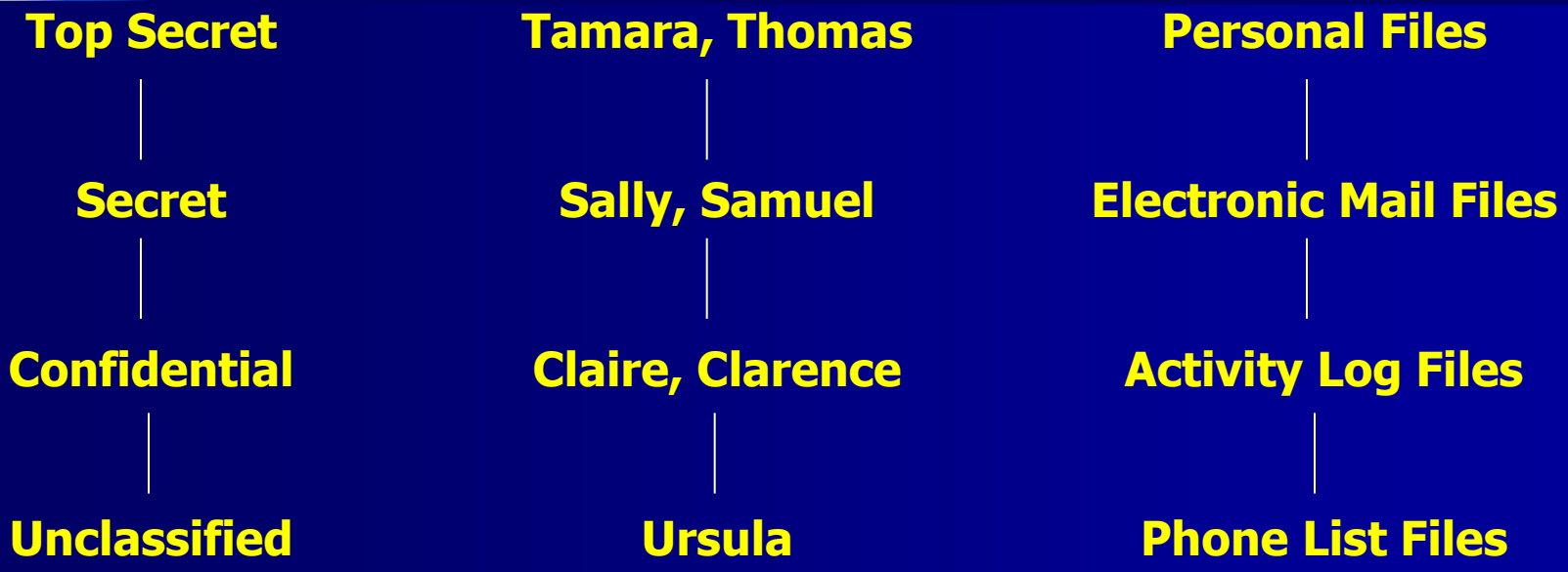
Bell-LaPadula: Basics

- Mandatory access control
  - Entities are assigned security levels
  - Subject has security clearance $L(s) = l_s$
  - Object has security classification $L(o) = l_o$
  - Simplest case: Security levels are arranged in a linear order $l_i < l_{i+1}$

- Example

  Top secret > Secret > Confidential >Unclassified

Bell-LaPadula: Basics



- Tamara and Thomas are cleared into (are in or have clearance at) the security level Top Secret.
- Electronic Mail Files are in the security level Secret.
- …

Bell-LaPadula: Basics

- The goal of the Bell-LaPadula security model is to prevent read access to objects at a security classification higher than the subject's clearance
- Examples
  - Claire can read Activity Log Files and Phone List Files, but can't read Personal Files and Mail Files.
- Bell-LaPadula combines 2 MAC rules and one DAC rule with three security properties:
  - **The Simple Security Property,**
  - **The *-property, and**
  - **The the Discretionary Security Property**

Bell-LaPadula: Basics

- ***Simple security property,*** preliminary version:
  - *s* can read *o* if and only if
    - $l_o \leq l_s$ and
    - *s* has read access to *o*
  - Combines mandatory *(security levels)* and discretionary *(permission required)*
  - Prevents subjects from reading objects at higher levels (*No Read Up rule*)
- Examples:
  - Ursula can read only phone list file
- Information is allowed to flow *up*, not *down*

Bell-LaPadula: Basics

- ***-property*** , preliminary version:
  - *s* can write *o* if and only if
    - $l_s \leq l_o$ and
    - *s* has write access to *o*
  - Combines mandatory *(security levels)* and discretionary *(permission required)*
  - Prevents subjects from writing to objects at lower levels (*No Write Down rule*)
- Examples:
  - Ursula can write any where but Clear can't write in the Phone List Files
- Information is allowed to flow *up*, not *down*

Bell-LaPadula: Basics

Basic Security Theorem: Let Σ be a system with secure initial state σ_0, and let T be a set of state transformations. If every element of T preserves the simple security condition, preliminary version, and the *-Property, preliminary version, then every state σ_i, $i \geq 0$, is secure.

Bell LaPadula: Expanded

- Total order of classifications not flexible enough
  - Alice cleared for missiles; Bob cleared for warheads; Both cleared for targets
- Solution: Categories
  - Use set of compartments
  - Enforce "*need to know*" principle
  - Security levels (level, category set)
    - (Top Secret, {Nuc, Eur, Asi})
    - (Top Secret, {Nuc, Asi})
- ***Dominates relation***
  - (L, C) ***dominates*, or *Dom*,** $(L', C') \Leftrightarrow L' \leq L$ and $C' \subseteq C$
  - Induces lattice of security levels

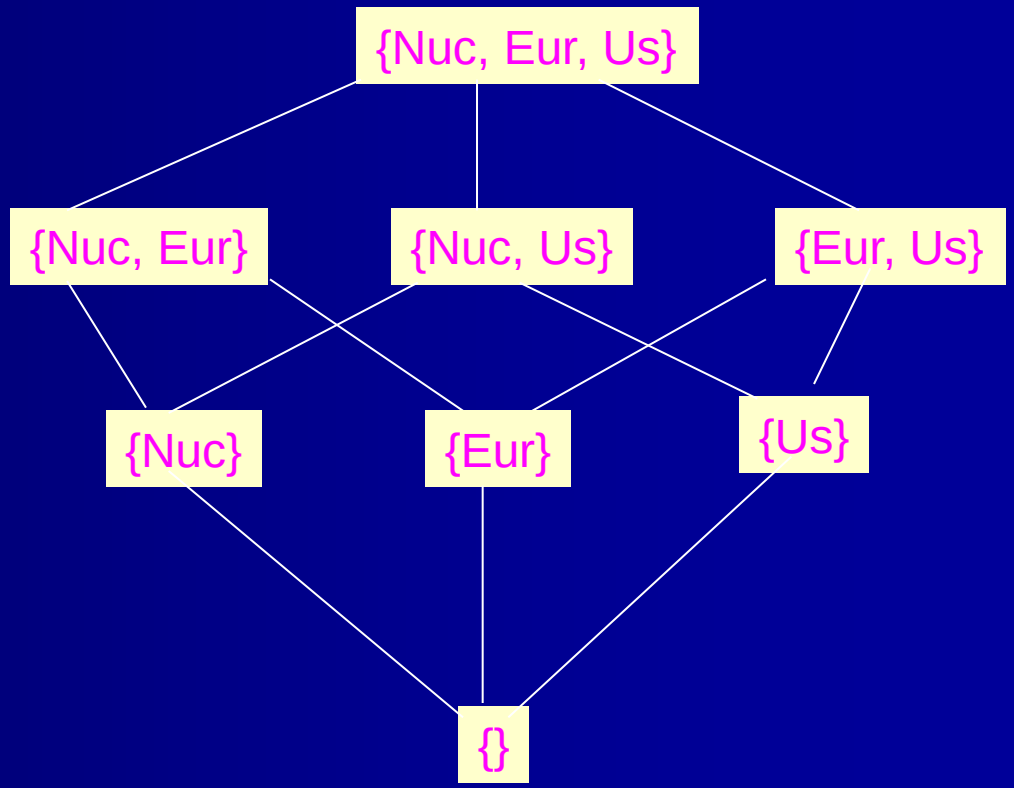
Bell LaPadula: Expanded

- Dominate relation
  - George is cleared into security level (Secret, {NUC,EUR}),
  - DocA is classified as (Confidential, {NUC})
  - DocB is classified as (Secret, {EUR, US})
  - DocC is classified as (Secret, {EUR})

  - Then
    - George *Dom* DocA
    - George ¬*Dom* DocB
    - George *Dom* DocC

Bell LaPadula: Expanded Lattice of categories

- Examples of levels:
  - (Top Secret, {Nuc,Asi}) *dom* (Secret, {Nuc})?
  - (Secret, {Nuc, Eur}) *dom* (Confidential, {Nuc,Eur})?
  - (Top Secret, {Nuc}) *dom* (Confidential, {Eur}) ?



Bell LaPadula: Expanded:

- *Simple Security Condition*: *S* can read *O* if and only if
  - *Clearance of S dominates classification of O* and
  - *S* has read access to *O*
- **-Property*: *S* can write *O* if and only if
  - *Classification of O dominates clearance of S* and
  - *S* has write access to *O*

Basic Security Theorem: Let Σ be a system with secure initial state σ_0, and let *T* be a set of state transformations. If every element of *T* preserves the simple security condition and the *-Property, then every state σ_i, $i \geq 0$, is secure.

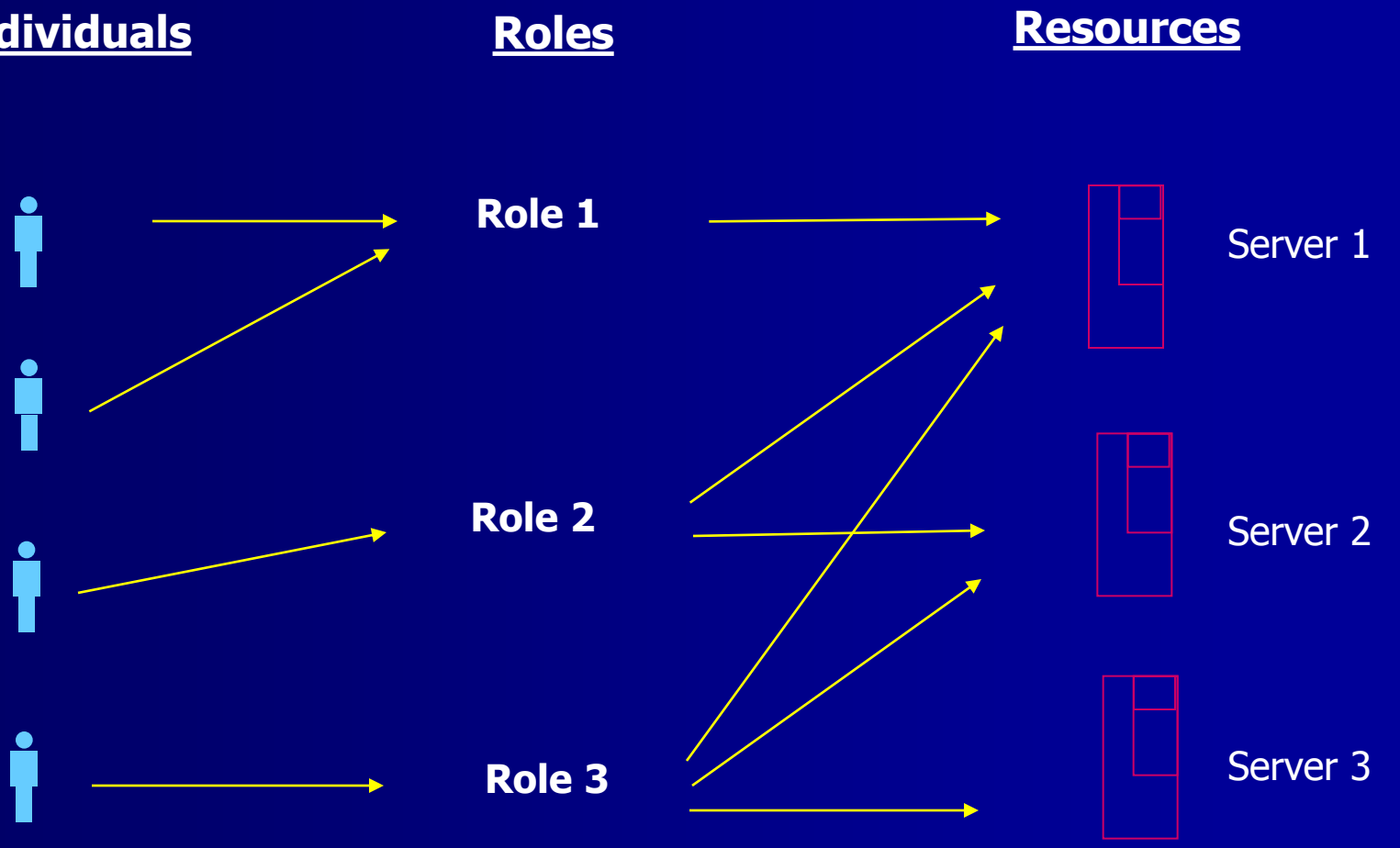
4.4. RBAC

- One challenging problem in managing large systems is the complexity of security administration.
- End users often do not own the information for which they are allowed access. The corporation or agency is the actual owner of data objects.
- Control is often based on employee functions rather than data ownership.
- RBAC has been proposed as an *alternative* approach to DAC and MAC, both to
  - Simplify the task of access control management,
  - Directly support function-based access control.

RBAC

- ***Roles*** represent functions within a given organization and authorizations are granted to roles instead of to single users
- Users are thus simply authorized to "play" the appropriate roles, thereby acquiring the roles' authorizations
- RBAC: Benefits
  - Because roles represent organizational functions, an RBAC model can directly support security policies of the organization
  - Granting and revoking of user authorizations is greatly simplified
- Most commercial DBMSs support RBAC features at some extents

RBAC



Users change frequently, roles don't

RBAC: SQL Commands

- `CREATE ROLE role-name IDENTIFIED BY passwd |NOT IDENTIFIED;`
- Example:
  - CREATE ROLE teller IDENTIFIED BY cashflow;
- `DROP ROLE role-name;`
- `GRANT role TO user | role | PUBLIC [WITH ADMIN OPTION];`
- To perform the grant of a role, a user must have the privilege for the role with the ADMIN option, or the system privilege GRANT ANY ROLE
  - The ADMIN option allows the receiver to modify or drop the role

RBAC: SQL Commands

- Example:
  - `GRANT teller TO Bob WITH ADMIN OPTION;`
- The *grant* command for authorization granting can have roles as subjects
- Example:
  - `GRANT select ON Employee TO teller;`

5. Conclusion

- Database security Based on Access Controls
- 4 access control models
  - DAC
  - CBAC
  - MAC
  - RBAC


Questions?