Access Control and Information Flow

UNIVERSITY OF
MARYLAND

Permissions

- How to describe a system’s protection mechanism
  - Such as who has what access rights to which objects
- Access control model
  - A model for security policy specification
  - Basic model: Access control matrix
- Security policy
  - Specifying who has the access rights to what
- Security mechanism
  - Enforce security policies

Access Control Matrix (ACM)

- S: subjects, users or processes
- O: objects, resources such as files, devices, messages, etc.
- A: access matrix A: $S \times O \rightarrow R$ (rights)
- Example:

S \ O	file 1	file 2	file 3
process 1	R W	R	R W E
process 2	R	R W	R

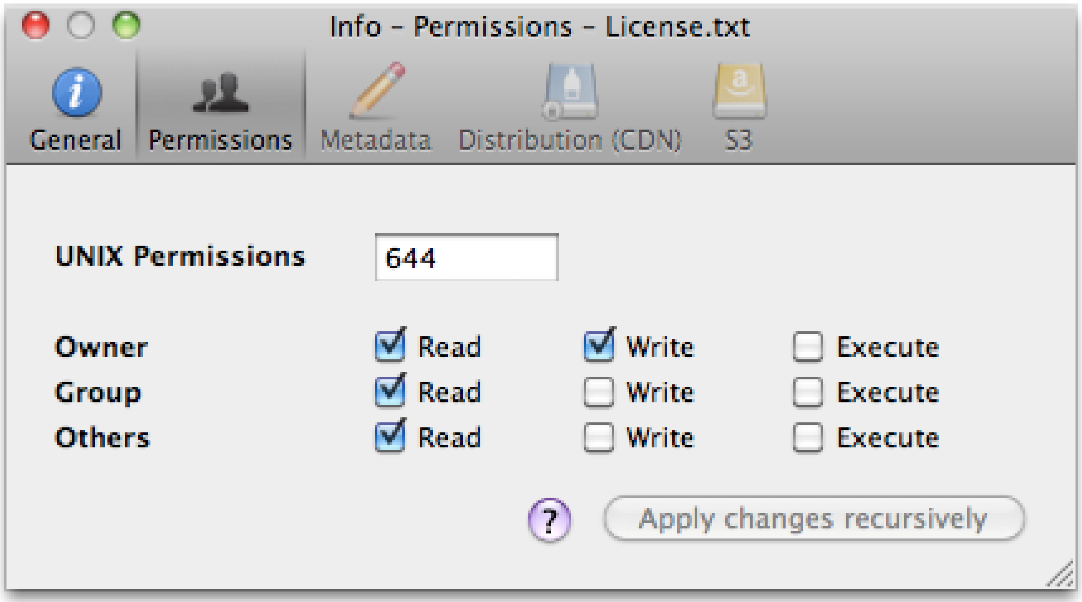
ACM

- ACM implementation
  - Space requirements: For m objects and n subjects: m x n
  - Generally the matrix is very sparse
  - **Access control list (ACLs):** describe the access policies for each object
  - **Capabilities:** describe the access rights each subject has

- ACM does not cover
  - Time constraints
    - E.g., only allowed to access at day time
- Advantages of ACLs? Disadvantages of ACLs?
- Advantages of Capabilities? Disadvantages of Capabilities?

ACL in Unix

- In a real system
  - Too many subjects and objects
- Unix
  - Classify subjects into: owner, group, world
  - Use ACL for each object, but in terms of owner, group, world



Setting Special Permissions

suid	sgid	stb	r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1	4	2	1
7			7			7			7		
Special			user			group			others		

Use the “chmod” command with octal mode:

chmod 7777 filename

uids and effective uids

- Every user has a user id that is called uid.
- When user A executes program B, program B is using A's uid
- However:
- Programs can change to use the effective user id euid
  - Effective user id euid is the uid of the program owner
  - e.g. the `passwd` program changes to use its effective uid (`root`) so that it can edit the `/etc/passwd` file
  - This special permission allows a user to access files and directories that are normally only available to the owner
  - SUID bit enables this functionality

Sample SETUID Scenario

- `/dev/lp` is owned by root with protection `rw-------`
  - This is used to access the printer
- `/bin/lp` is owned by root with `rwsr-xr-x` (with SETUID=1)
- User A issues a print process B
- Process B has the same UID as user A
- Process B executes `exec("/bin/lp",…)`
- But lp is a setuid program and now B is using root's UID
- Consequently, `/dev/lp` can be accessed to print
- When `/bin/lp` terminates so does B
- User never got the access to `/dev/lp`

A simple program

- Say I (cpap) own the program

```
FILEWRITE(file,uid,data):   rwx--x--x
IF write_access(file,uid) = 0
        exit;
ELSE
        open_for_write(file);
        write_data(file,data);
```

- This program can only write to Bob’s file if executed by Bob.
- Can it write to cpap’s file **private** if executed by Bob?
    - NO!! It is going to exit after the first access control check
- What if cpap decides to make it setuid?

Problem with setUiD: Race conditions

- Now, let's see the setuid program

```
FILEWRITE(file,uid,data):   rws--x--x
IF write_access(file,uid) = 0
        exit;
ELSE
        open_for_write(file);
        write_data(file,data);
```

Attacker enters symbolic link
symlink(file,cpap/private)

- This program can be executed by Bob
- And it can write to cpap's file **private** due to race condition
- CAREFUL with SETUID programs!!

Access Control Models

- Discretionary access control (DAC)
  - Owner determines access rights
  - Typically identity-based access control: access rights are assigned to users based on their identity
  - E.g., ACM
- Mandatory access control (MAC)
  - System enforce system-wide rules for access control
  - E.g., law allows a court to access driving records without the owners' permission
- Role based access control
  - Identity governed by the roles a user assumes
  - E.g., children under 13

DAC and MAC

- When is DAC insufficient?
  - When owner cannot be trusted for the discretion of the data and external protection of the data is necessary
  - E.g., DAC has the danger of right propagation
    - A can read X and write Y
    - B can read Y, but no access to X
    - A reads X, write the content of X to Y, B got access to X
- MAC
  - Non-discretionary
  - Labels are assigned to subjects and objects
  - Owner has no special privileges
  - E.g., Bell-Lapadula, lattices models

Traditional Models for MAC

- Bell-LaPadula (BLP)
  - Address confidentiality
- Biba
  - Address integrity with static/dynamic levels

Bell-LaPadula Security Model

- The Bell-LaPadula (BLP) model is about information *confidentiality*
- It was developed to formalize the US Department of Defense multilevel security policy

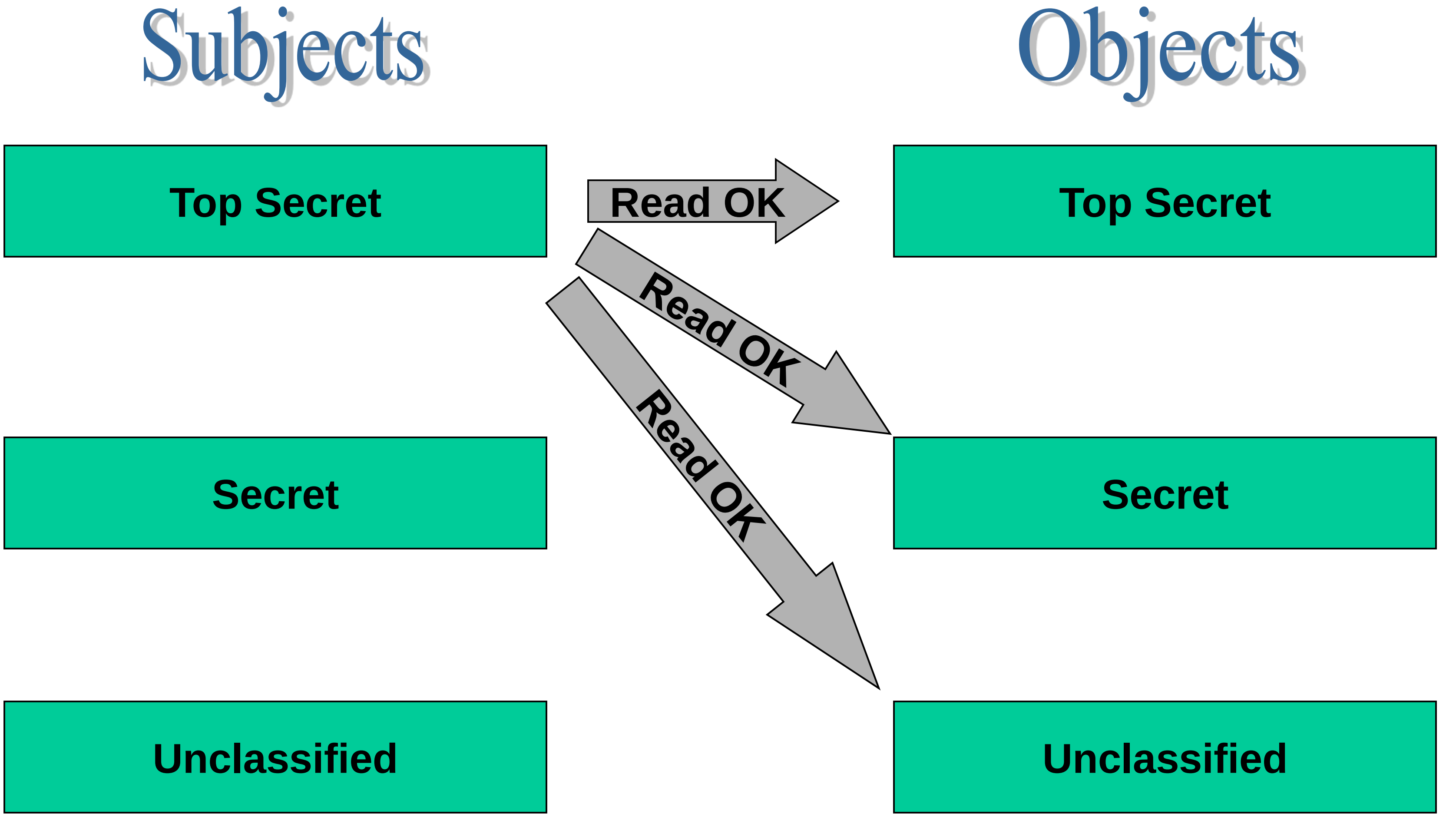
Bell – LaPadula - Details

- Each user subject and information object has a fixed security class – labels
- Use the notation $\leq$ to indicate dominance
- Simple Security (ss) property:

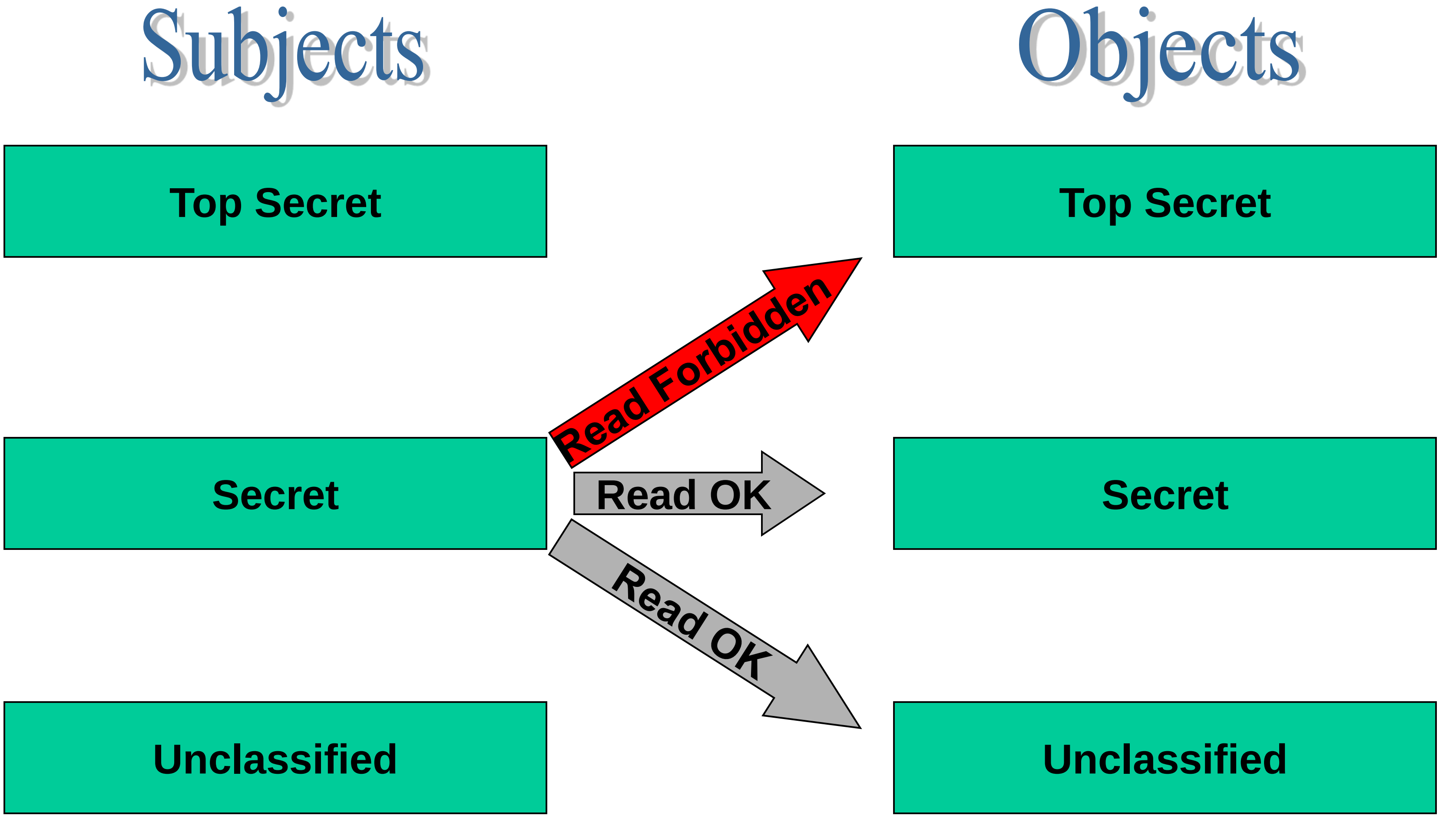
  the no read-up property

  - A subject s has read access to an object o iff the class of the subject C(s) is greater than or equal to the class of the object C(o)
  - i.e. Subjects can read Objects iff $C(o) \leq C(s)$

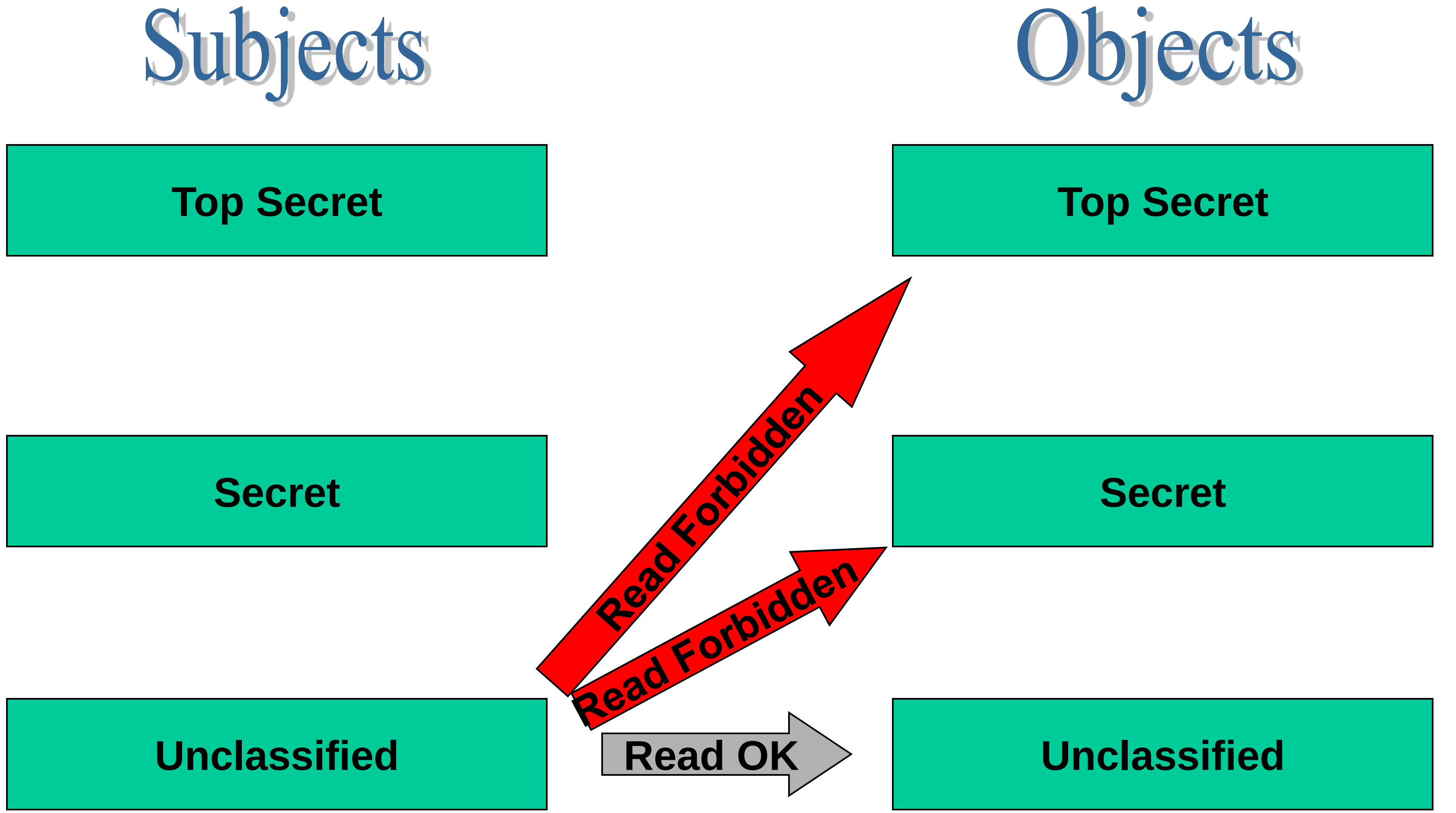
Access Control: Bell-LaPadula



Access Control: Bell-LaPadula



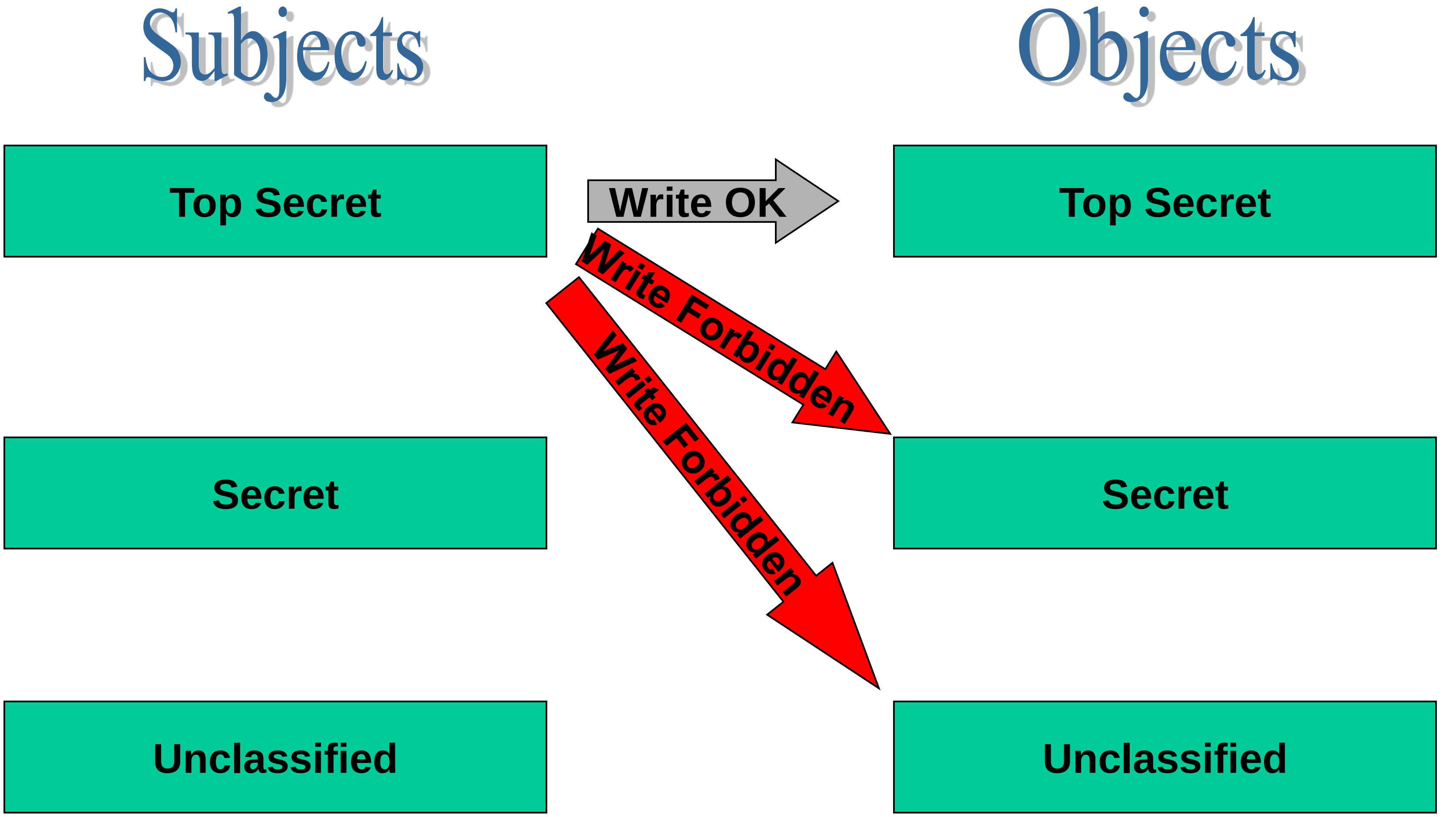
Access Control: Bell-LaPadula



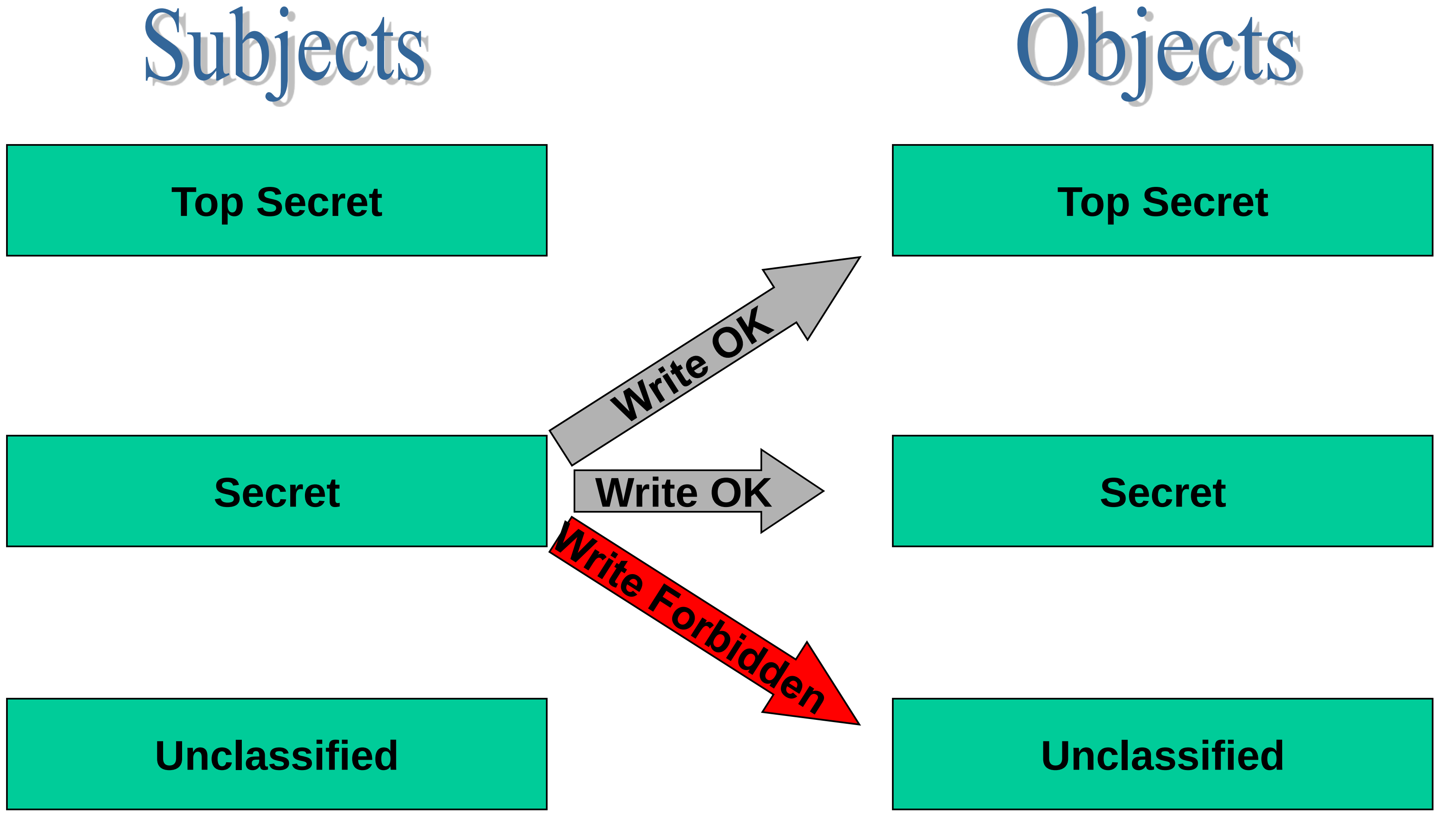
Bell - LaPadula (2)

- * property (star):
  the no write-down property
  - A subject s can write to object *p* if $C(s) \leq C(p)$

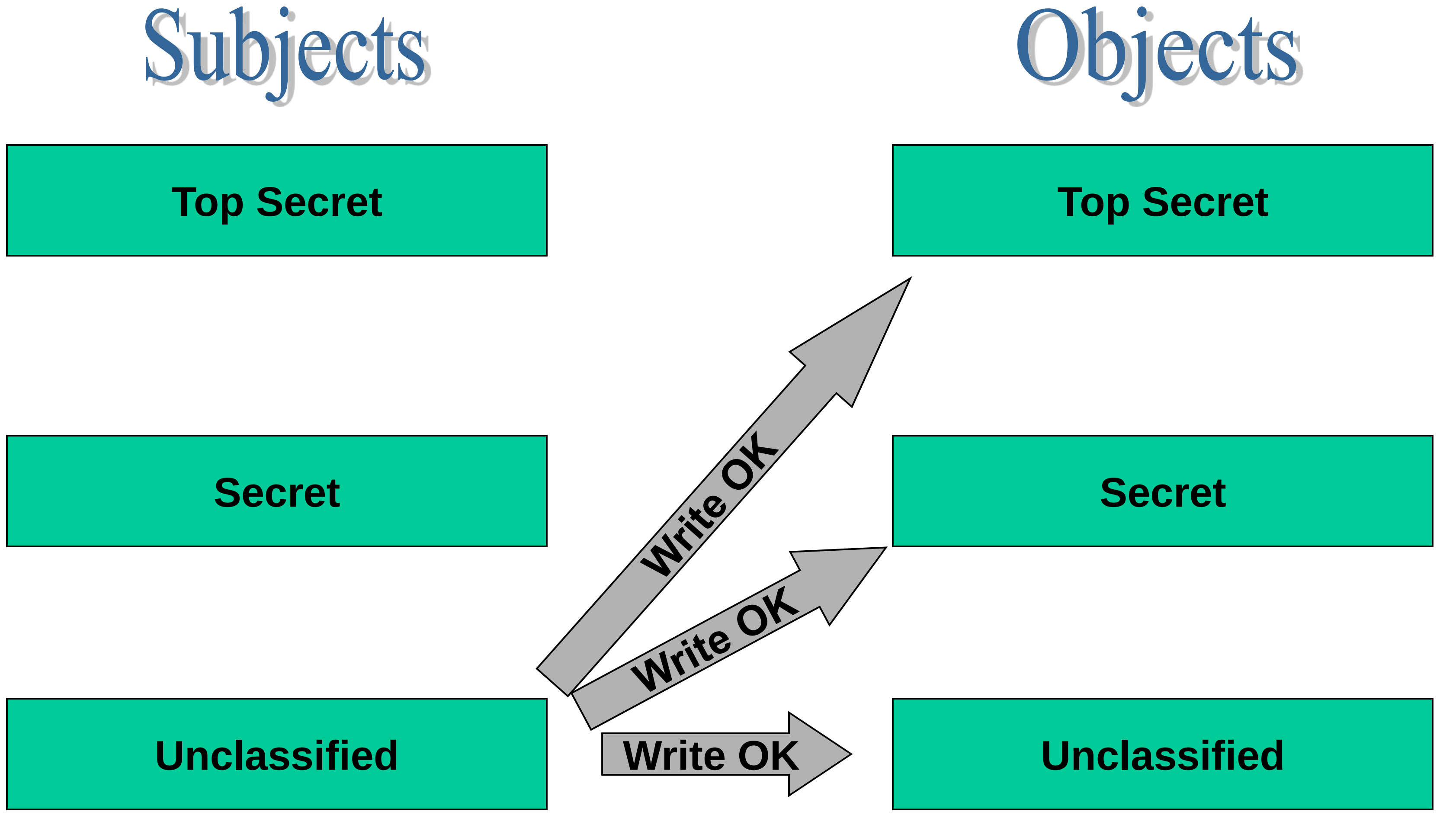
Access Control: Bell-LaPadula



Access Control: Bell-LaPadula



Access Control: Bell-LaPadula



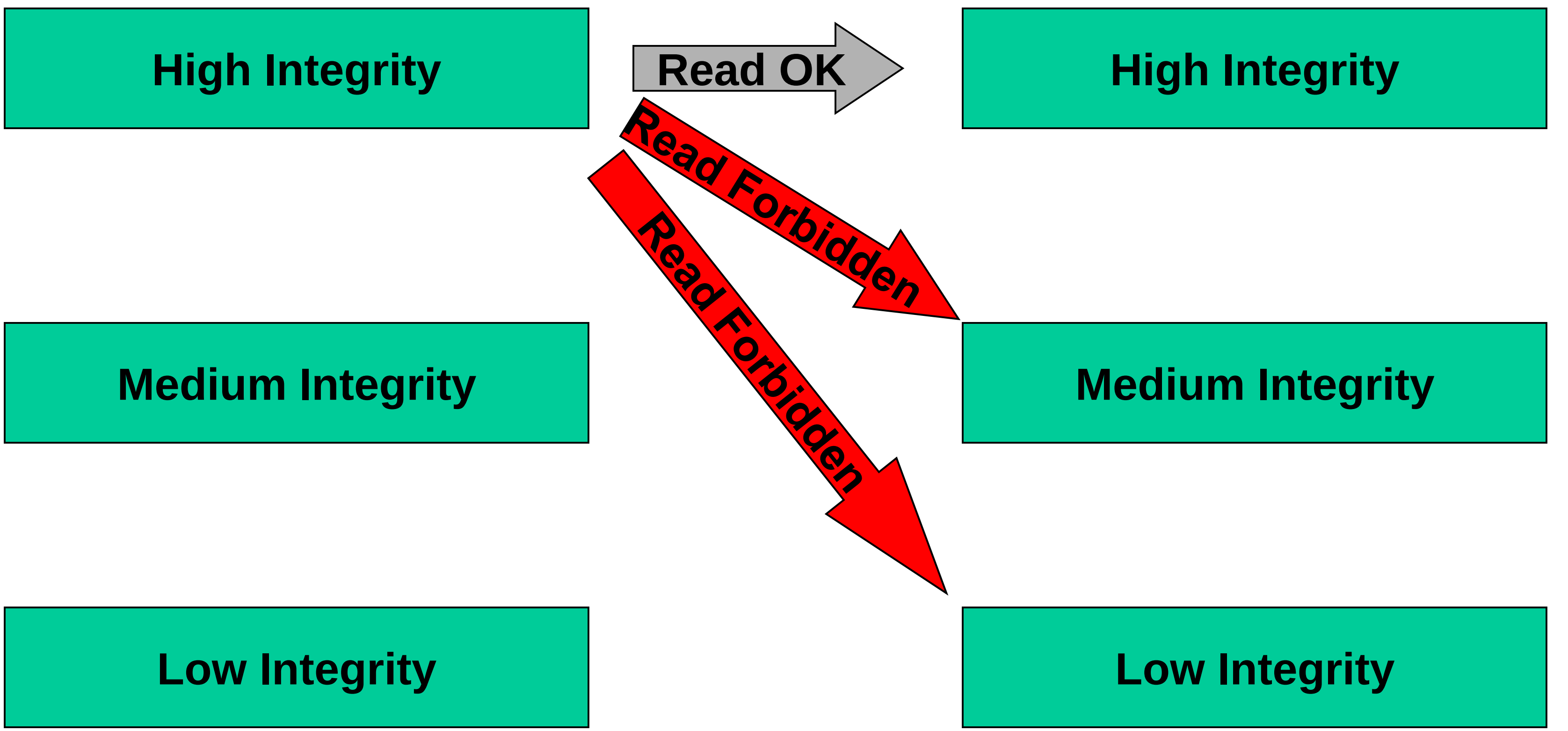
Security Models - Biba

- Based on the Cold War experiences, information *integrity* is also important, and the Biba model, complementary to Bell-LaPadula, is based on the flow of information where preserving integrity is critical.
- The “dual” of Bell-LaPadula

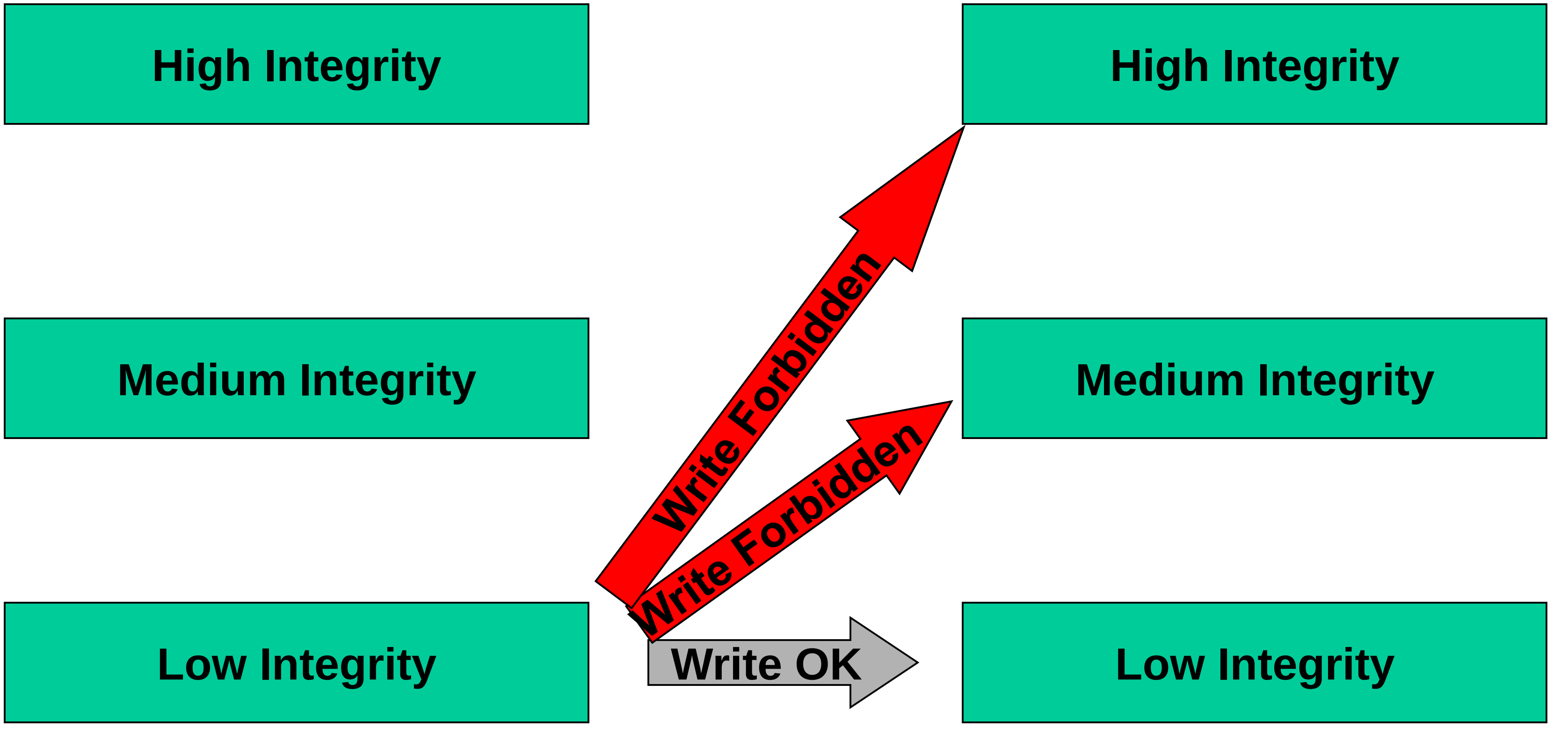
Integrity Control: Biba

- Designed to preserve integrity, not limit access
- Three fundamental concepts:
  - Simple Integrity Property – no read down
  - Star Integrity Property (*) – no write up
  - No execute up

Integrity Control: Biba



Integrity Control: Biba



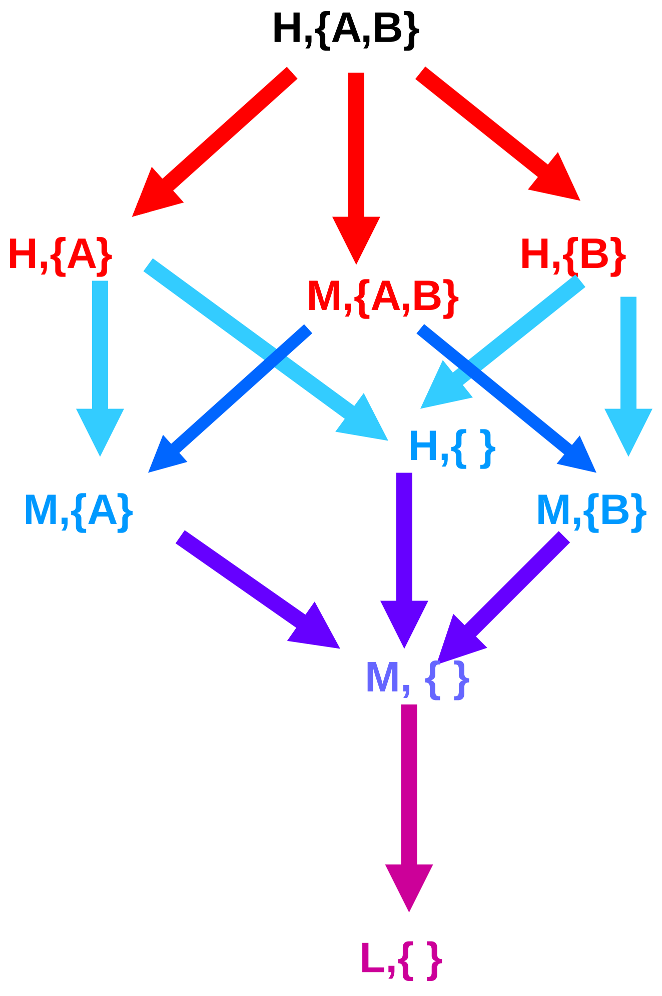
Combining integrity and privacy into a lattice

- Integrity
  - High Integrity (H)
  - Medium Integrity (M)
  - Low integrity (L)
  - No integrity (N)
- Confidentiality
  - {A,B} can be read by both A and B
  - {A} can be read only by A
  - {B} can be read only by B

Security Lattice

- *S* is the set of all security levels
  - Suppose the integrity categories are H (high integrity), M (medium integrity), L (low integrity)
  - Suppose the confidentiality categories are {A},{B},{A,B} and {}.
  - Then States = [(H, {}), (H,{A}), (H,{B}), (H,{A,B}), (M, {}), (M,{A}), (M,{B}), (M,{A,B}), (L, {})].

Information Flow in a security lattice



Information Flow – Informal

- What do we mean by information flow?
  - `y = x;`
  - `y = x/z;`
- A command sequence *c* causes a flow of information from x to y if the value of y after the commands allows one to deduce information about the value of x
  - `tmp = x;`
  - `y = tmp;`
  - Transitive

Information Flow *Models*

- Two categories of *information flows*
  - explicit – operations causing flow are independent of value of x, e.g. assignment operation, `x=y`
  - implicit - conditional assignment
    - (`if x = 5 then y=1 else y=0`)
- Components
  - Lattice of security levels (L, ≤)
  - Set of labeled objects
  - Security policy