

# Bell LaPadula Model

MCS 7102 Data Security and Privacy

Lecture 3

27th August 2024



# Objectives for this Lecture

- Understand basic operation of the models below.
- Bell LaPadula Model
- Biba Model
- Chinese Wall

# Bell LaPadula Model

- “The quality or state of being secure—to be free from danger”
- The Bell-LaPadula Model corresponds to military-style classifications
- The Bell-LaPadula security model combines mandatory and discretionary access controls

# Security Levels

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist of security clearance L(s)
- Objects have security classification L(o)



# History of BLP

- Developed in 1970s
- Formal model for access control
- Subjects and objects are assigned a security class
- Form a hierarchy and are referred to as security levels
- A subject has a security **clearance**
- An object has a security **classification**
- Security classes control the manner by which a subject may access an object

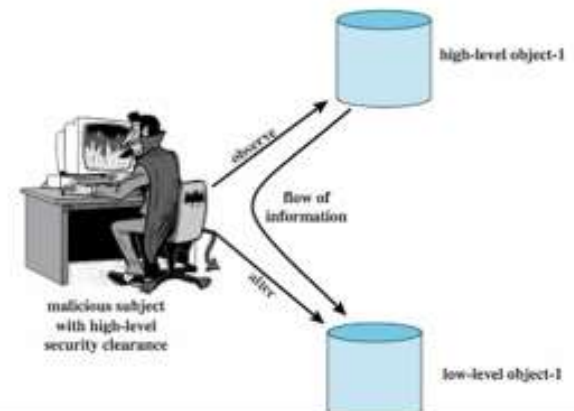
# A BLP Example

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- James can only read Telephone Lists

Security level	Subject	Object
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	James	Telephone Lists

# Multilevel Security

- Multiple levels of security and data
- Subject at a high level may not convey info to a subject at a non-comparable level:
  - No read up (ss-property): a subject can only read an object of less or equal security level
  - No write down (\*-property): a subject can only write into an object of greater or equal security level



# BLP Formal Description

- Based on current state of system  $(b, M, f, H)$ :
  - Current access set  $b$  (*subj, objs, access-mode*); it is the **current** access (not permanent)
  - Access matrix  $M$  (same as before)
  - Level function  $f$ : *assigns security level to each subject and object; a subject may operate at that or lower level*
  - Hierarchy  $H$ : *a directed tree whose nodes are objects:*
    - *Security level of an object must dominate (must be greater than) its parents*



# BLP Properties

- Three BLP properties: ( $c = \text{current}$ )
  1. ss-property:  $(S_i, O_j, \text{read})$  has  $f_c(S_i) \geq f_o(O_j)$   
simple security property: a subject at a specific classification level cannot read data with a higher classification level. This is often shortened to “no read up.”
  2. \*-property:  $(S_i, O_j, \text{append})$  has  $f_c(S_i) \leq f_o(O_j)$  and  
 $(S_i, O_j, \text{write})$  has  $f_c(S_i) = f_o(O_j)$   
Star property: states that a subject at a specific classification level cannot write data to a lower classification level. This is often shortened to “no write down.”
  3. ds-property:  $(S_i, O_j, A_x)$  implies  $A_x \in M[S_i, O_j]$   
Discretionary-security (ds-property) rule is mainly based on named subjects and objects. It states that specific permissions allow a subject to pass on permissions at its own discretion.
- BLP give formal theorems
  - Theoretically possible to prove system is secure

# BLP Operations

1. **get access:** add (subj, obj, access-mode) to  $b$  (current access)
  - used by a subj to initiate an access to an object
2. **release access:** remove (subj, obj, access-mode)
3. **change object level**
4. **change current level**
5. **give access permission:** Add an access mode to  $M$ 
  - used by a subj to grant access to on an obj
6. **rescind access permission:** reverse of 5
7. **create an object**
8. **delete a group of objects**

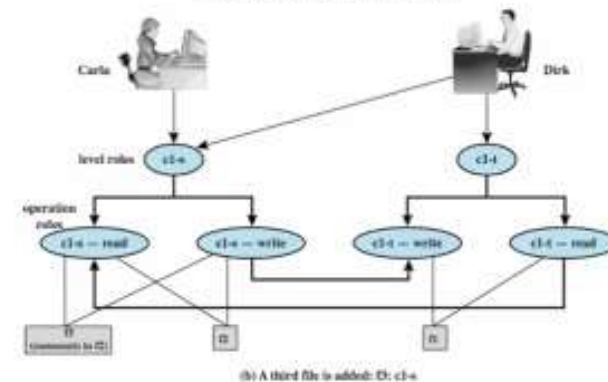
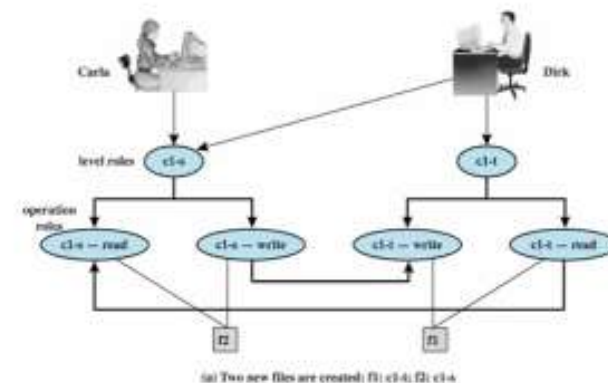
# BLP Example

- A role-based access control system
- Two users: Carla (student) and Dirk (teacher)
  - Carla (Class: s)
  - Dirk (Class: T); can also login as a students thus (Class: s)
- A student role has a lower security clearance
- A teacher role has a higher security clearance

- We assume a role-based access control system. Carla and Dirk are users of the system. Carla is a student (s) in course c1. Dirk is a teacher (t) in course c1, but may also access the system as a student; thus two roles are assigned to Dirk: Carla: (c1-s); and Dirk: (c1-t), (c1-s).
- The student role is assigned a lower security clearance and the teacher role a higher security clearance. Let us look at some possible actions:

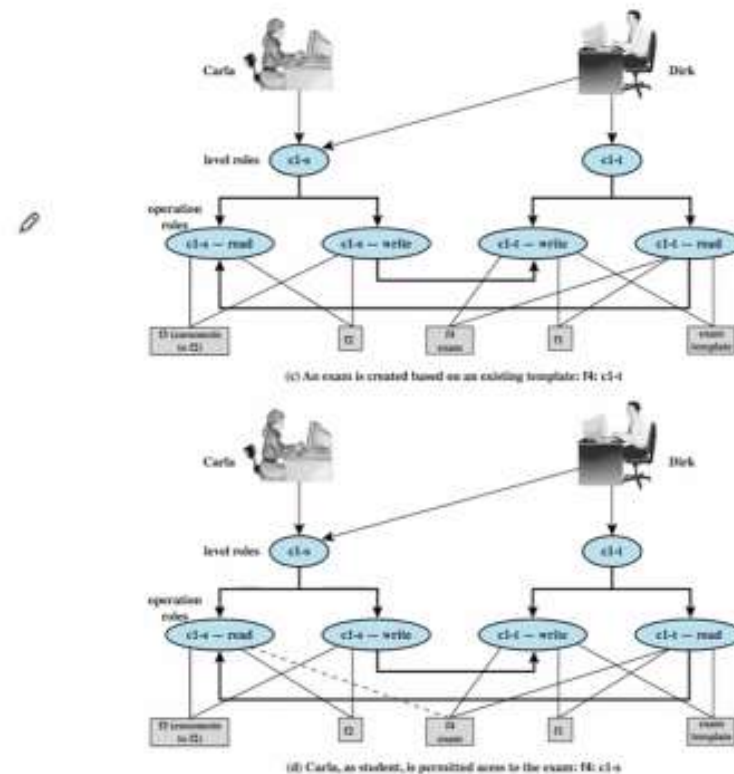
1. Dirk creates a new file f1 as c1-t; Carla creates file f2 as c1-s (Figure a). Carla can read and write to f2, but cannot read f1, because it is at a higher classification level (teacher level). In the c1-t role, Dirk can read and write f1 and can read f2 if Carla grants access to f2. However, in this role, Dirk cannot write f2 because of the \*-property; neither Dirk nor a Trojan horse on his behalf can downgrade data from the teacher level to the student level. Only if Dirk logs in as a student can he create a c1-s file or write to an existing c1-s file, such as f2. In the student role, Dirk can also read f2.

2. Dirk reads f2 and wants to create a new file with comments to Carla as feedback. Dirk must sign in student role c1-s to create f3 so that it can be accessed by Carla. In a teacher role, Dirk cannot create a file at a student classification level.



3. Dirk creates an exam based on an existing template file store at level c1-t. Dirk must log in as c1-t to read the template and the file he creates (f4) must also be at the teacher level.

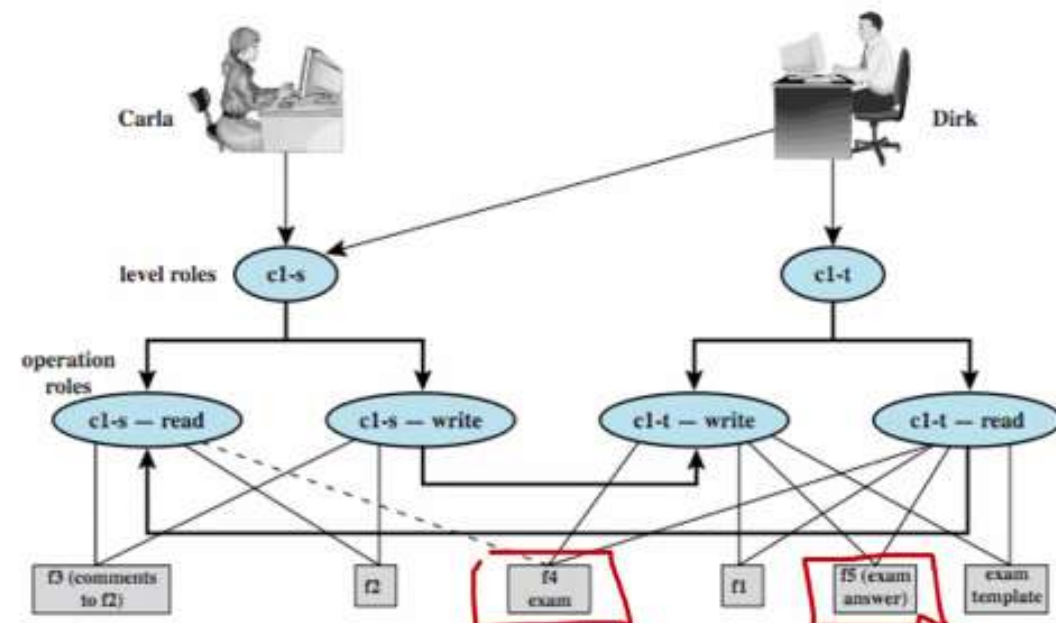
4. Dirk wants Carla to take the exam and so must provide her with read access. However, such access would violate the ss-property. Dirk must downgrade the classification of f4 from c1-t to c1-s. Dirk cannot do this in the c1-t role because this would violate the \*-property. Therefore, a security administrator (possibly Dirk in this role) must have downgrade authority and must be able to perform the downgrade outside the BLP model. The dotted line in Figure d connecting f4 with c1-s-read indicates that this connection has not been generated by the default BLP rules but by a system operation.



5. Carla writes the answers to the exam into a file f5. She creates the file at level c1-t so that only Dirk can read the file. This is an example of writing up, which is not forbidden by the BLP rules. Carla can still see her answers at her workstation but cannot access f5 for reading.

This discussion illustrates some critical practical limitations of the BLP model. First, as noted in step 4, the BLP model has no provision to manage the “downgrade” of objects, even though the requirements for multilevel security recognize that such a flow of information may be required, provided it reflects the will of an authorized user. Hence, any practical implementation of a multilevel system has to support such a process in a controlled and monitored manner.

While the BLP model could in theory lay the foundations for secure computing within a single administration realm environment, there are some important limitations to it: an incompatibility of confidentiality and integrity within a single MLS system; and the so called cooperating conspirator problem in the presence of covert channels. In essence, the BLP model effectively breaks down when (untrusted) low classified executable data are allowed to be executed by a high clearance (trusted) subject.



(e) The answers given by Carla are only accessible for the teacher: f5: c1-t



# BLP Dominate (dom) Relationship

- Captures the combination of security classification and category set
- $(A, C) \text{ dom } (A', C')$  iff  $A' \leq A$  and  $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI})  $\text{dom}$  (Secret, {NUC})
  - (Secret, {NUC, EUR})  $\text{dom}$  (Confidential, {NUC, EUR})
  - (Top Secret, {NUC})  $\neg \text{dom}$  (Confidential, {EUR})

# An Example of dom Relationship

- George is cleared into security level  $(S, \{NUC, EUR\})$ 
  - DocA is classified as  $(C, \{NUC\})$
  - DocB is classified as  $(S, \{EUR, US\})$
  - DocC is classified as  $(S, \{EUR\})$
- George *dom* DocA as  $CONFIDENTIAL \leq SECRET$  and  $\{NUC\} \subseteq \{NUC, EUR\}$
- George  $\not\text{dom}$  DocB as  $\{EUR, US\} \not\subseteq \{NUC, EUR\}$
- George *dom* DocC as  $SECRET \leq SECRET$  and  $\{EUR\} \subseteq \{NUC, EUR\}$

Let  $C(S)$  be the category set of subject  $S$ , and let  $C(O)$  be the category set of object  $O$ .



# Reading Information - New

- Information flows *up*, not *down*
  - “Reads up” disallowed, “reads down” allowed
- Simple Security Condition

Subject  $s$  can read object  $o$  iff  $L(s) \text{ dom } L(o)$  and  $s$  has permission to read  $o$  (*Simple Security Condition:  $S$  can read  $O$  if and only if  $S \text{ dom } O$  and  $S$  has discretionary read access to  $O$ .*)

- Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
- Sometimes called “no reads up” rule

# Writing Information - New

- Information flows up, not down
    - “Writes up” allowed, “writes down” disallowed
  - \*-Property (Step 2)
    - Subject  $s$  can write to object  $o$  iff  $L(o) \text{ dom } L(s)$  and  $s$  has permission to write  $o$
- (\*-Property:**  $S$  can write to  $O$  if and only if  $O \text{ dom } S$  and  $S$  has discretionary write access to  $O$ .)
- Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called “no writes down” rule

# Integrity Policies

- Integrity policies focus on integrity rather than confidentiality
- Most commercial and industrial firms are more concerned with accuracy than disclosure

# Goals—Identify five requirements


1. Users will not write their own programs, but will use existing production programs and databases
2. Programmers will develop and test programs on a non-production system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system
3. A special process must be followed to install a program from the development system onto the production system
4. The special process in requirement 3 must be controlled and audited
5. The managers and auditors must have access to both the system state and the system logs that are generated


# Requirements suggest principles of operation

1. *Separation of duty*—principle of separation of duty states that if two or more steps are required to perform a critical function, at least two different people should perform the steps
2. *Separation of function*—Developers do not develop new programs on production systems because of the potential threat to production data
3. *Auditing*—Commercial systems emphasize recovery and accountability

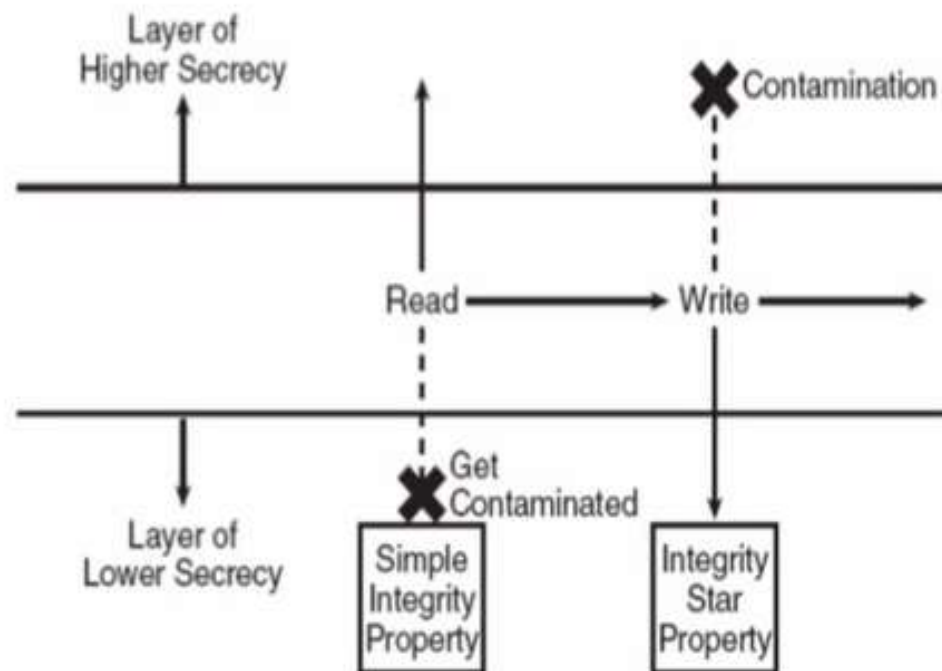
# Biba Integrity Model

- Various models dealing with integrity
- Strict integrity policy:
  - Simple integrity: *modify only if*  $I(S) \geq I(O)$
  - Integrity confinement: *read only if*  $I(S) \leq I(O)$
  - Invocation property: *invoke/command only if*  $I(S_1) \geq I(S_2)$


- 
- The Biba model deals with integrity and is concerned with the unauthorized modification of data. The basic elements of the Biba model are like the BLP model. As with BLP, the Biba model deals with subjects and objects
  - Each subject and object is assigned an integrity level, denoted as  $I(S)$  and  $I(O)$  for subject  $S$  and object  $O$ , respectively. A simple hierarchical classification can be used, in which there is a strict ordering of levels from lowest to highest. Biba then proposes a number of alternative policies that can be imposed on this model

- 
- The most relevant is the strict integrity policy, based on the following rules:
    - Simple integrity: A subject can modify an object only if the integrity level of the subject dominates the integrity level of the object:  $I(S) \geq I(O)$
    - Integrity confinement: A subject can read an object only if the integrity level of the subject is dominated by the integrity level of the object:  $I(S) \leq I(O)$
    - Invocation property: A subject can invoke another subject only if the integrity level of the 1st subject dominates the integrity level of the 2nd subject:  $I(S1) \geq I(S2)$





- Simple integrity: *modify only if*  
 $I(S) \geq I(O)$
- Integrity confinement: *read only if*  $I(S) \leq I(O)$
- Invocation property:  
*invoke/comm only if*  $I(S_1) \geq I(S_2)$

- 
- The simple integrity rule is the logical write-up restriction that prevents contamination of high-integrity data. The need for the integrity confinement rule
  - A low-integrity process may read low-integrity data but is prevented from contaminating a high-integrity file with that data by the simple integrity rule
  - If only this rule is in force, a high-integrity process could conceivably copy low-integrity data into a high-integrity file. Normally, one would trust a high-integrity process to not contaminate a high-integrity file, but an error in the code or a Trojan horse could result in such contamination. Hence the need for the integrity confinement rule

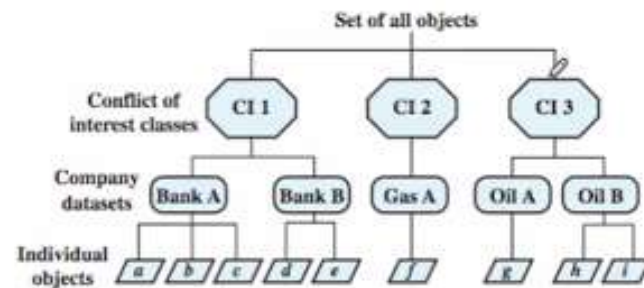
# The Chinese Wall Model

- Hybrid model: addresses integrity and confidentiality
- Addresses conflict of interest (CI or Col)
- Model elements
  - **subjects**: active entities interested in accessing protected objects
  - **information**
    - objects: individual data items, each about a company
    - datasets (DS): all objects concerning one company
    - CI class: datasets for companies that are in competition (conflict of interest or CI)
  - **access rules**: rules for reading/writing data

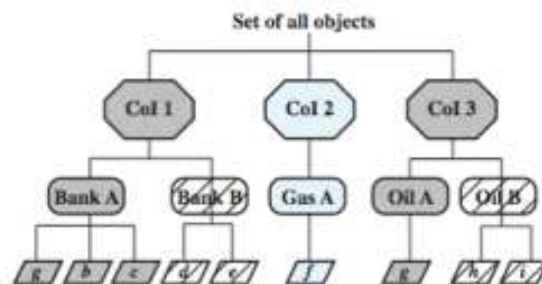
- Not a true multilevel secure model
  - the history of a subject's access determines access control
- Subjects are only allowed access to info that is not held to conflict with any other info they already possess
- Once a subject accesses info from one dataset, a *wall* is set up to protect info in other datasets in the same CI

**Simple sec rule (read):** S can read O if O is in the same DS as an object already accessed by S OR O belongs to a Col from which S has not yet accessed any info

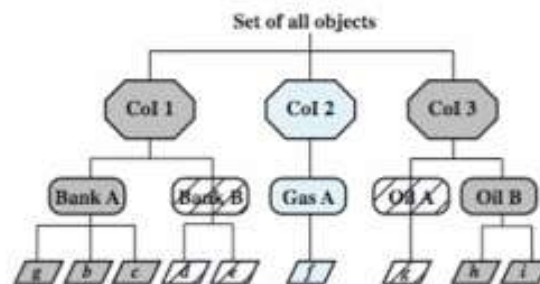
**\*-property (write):** S can write O only if S can read O and all objects that S can read are in the same DS as O.



(a) Example set



(b) John has access to Bank A and Oil A



(c) Jane has access to Bank A and Oil B

- 
- The Chinese Wall Model (CWM) takes a quite different approach to specifying integrity and confidentiality than any of the approaches we have examined so far. The model was developed for commercial applications in which conflicts of interest can arise.
  - The principal idea behind the CWM is the use of a Chinese wall to prevent a conflict of interest. There are datasets representing banks, oil companies, and gas companies. All bank datasets are in one CI, all oil company datasets in another CI, and so forth. In contrast to the models we have studied so far, the history of a subject's previous access determines access control.
  - The basis of the Chinese wall policy is that subjects are only allowed access to information that is not held to conflict with any other information that they already possess.



- Once a subject accesses information from one dataset, a wall is set up to protect information in other datasets in the same CI. The subject can access information on one side of the wall but not the other side.
- Further, information in other CIs is initially not considered to be on one side or the other of the wall but out in the open. When additional accesses are made in other CIs by the same subject, the shape of the wall changes to maintain the desired protection. Further, each subject is controlled by his or her own wall - the walls for different subjects are different.