CSCI262 : System Security

Week 4: Access Control 2

Schedule

- Security Model
- Confidential Policies
- Integrity Policies
- Hybrid Policies

Security Policies

- A **security policy** is a statement that partitions the states of the system into a set of **authorised**, or **secure**, states and a set of **unauthorised**, or **nonsecure**, states.
- A **secure system** is a system that starts in an authorised state and cannot enter an unauthorised state.
- Example: (from [B18])
  - Authorised state set $\{s_1,s_2\}$
  - Unauthorised state set $\{s_3,s_4\}$
  - The system is not secure

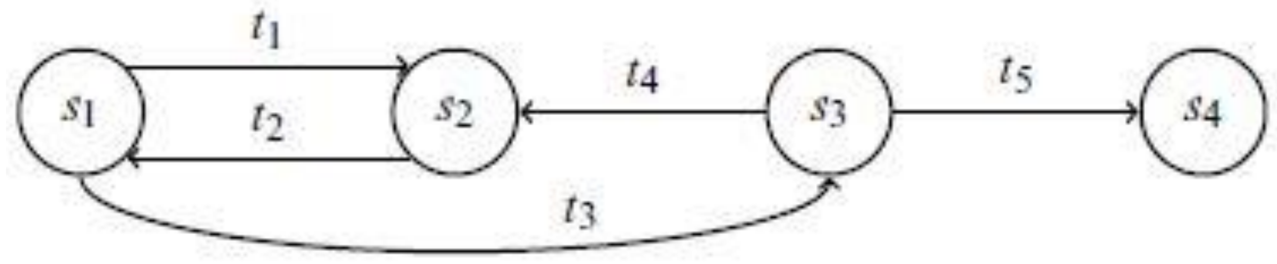


Figure 4-1 in [B18]

Security model

- A **security model** is a precise representation of the security requirements (security policy).
- Characteristics:
  - Simple and abstract.
  - Precise and unambiguous.
  - Generic.
    - Deals with security properties.
    - Does not unduly constrain system functions or implementation details.
- In state machine representations, we:
  - Describe the system as an abstract mathematical state machine.
  - Represent the state of the machine using state variables.
  - Describe how variables, and thus the state, changes using transition functions.

Traditional Security Models

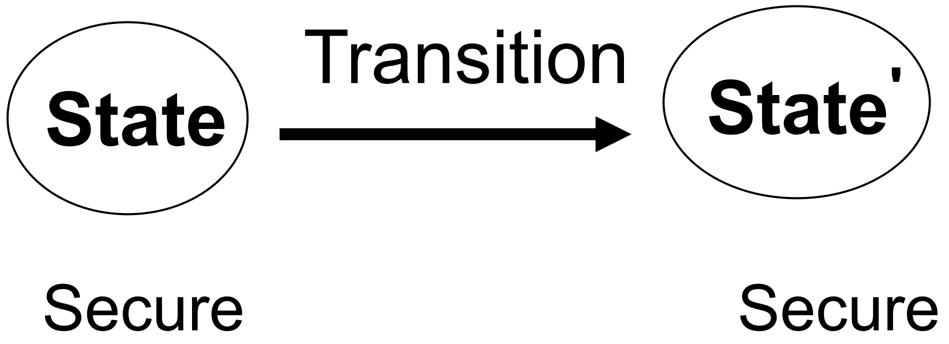
- System : <Subjects, Objects>
- Subjects : Active Entities
  : Users, Processes
- Objects : Passive Entities
  : Files, Records
- Security Relevant State Variables:
  <Subjects, Objects, Security Attributes>

State machine based security model

To prove security of a particular state machine based system we use the following procedure:

1. Define appropriate state variables.
   - State relationships between them.
2. Define conditions for a secure state:
   - This includes the relationships between values of state variables that must be maintained during state transitions.
   - These are security constraints.
3. Define state transition functions:
   - Define the mechanisms by which state variables can change.

4. Prove each transition function maintains a secure state, when acting on a secure state.



5. Define an initial state.

6. Show the initial state is secure.

We then apply induction to show evolution of the system must leave it secure, by 4. and 6.

Example

- S = {S1, S2, S3}
- O = {O1 , O2 , O3 }
- A ={read, write, execute}

- The state variables are combinations $<S_i,O_j,A_k>$ of these which can effectively be on or off, with on corresponding to that action currently being active between S and O.

- Conditions for a secure state (we will see shortly):
  - ss-property.
  - *-property.
  - ds-property.

Confidentiality policies

- A **confidentiality policy**, also called **information flow policy**, prevents the unauthorised disclosure of information
- **Bell-LaPadula (BLP) Model**
  - is a confidentiality based access control model
  - corresponds to military-type classifications
    - Developed primarily to provide confidentiality
  - has influenced the development of many other models and computer security technologies

The (simplified) Bell-LaPadula Model

- The Bell-LaPadula Model (1973, 1975) is a multilevel security model which works by specifying allowable paths of information flow in a secure system.
- This is an important model when a system/machine has to concurrently handle data at different sensitivity levels. For example, a machine processing confidential and top-secret files at the same time.

- The components of the model are as follows:
  - A set of objects $O=\{O_i\}$.
  - A set of subjects $S=\{S_i\}$.
  - A set of access operations A = {execute, read, write, append}.
    - Append is writing only. Write is being allowed to read and write ☹
  - A set of security levels L with a partial order $\leq$.
    - $(L, \leq)$ defines a lattice.
    - In this simplified model the lattice is reduced to a linear structure.

Lattices

- A **lattice** (L, ≤) consists of a set L and a partial order ≤ (generally a reflexive, antisymmetric, and transitive relation), so that for every two elements a, b ∈ L there exists:
- A least upper bound u ∈ L.
- A greatest lower bound l ∈ L.
- Formally:
  - $a \leq u$, $b \leq u$, and for all $v \in L : (a \leq v \wedge b \leq v) \rightarrow (u \leq v)$
  - $l \leq a$, $l \leq b$, and for all $k \in L : (k \leq a \wedge k \leq b) \rightarrow (k \leq l)$

Properties of the relation ≤

- Reflexive: a ≤ a.
- Antisymmetric: If a ≤ b and b ≤ a then a = b.
- Transitive: If a ≤ b and b ≤ c then a ≤ c.
- An important term:
  - If a ≤ b, b dominates a → b *dom* a
    - Domination can be interpreted as meaning requiring a higher security level.

Another example

- Consider a set S
- Let L(S) be the set of all subsets of S
- Let ≤ be the inclusion relation on L(S), i.e., for A, B in L(S), A ≤ B means that A is a subset of B. (A ⊆ B)
- ≤ is a partial order on L(S)
- Hence (L(S), ≤) is a lattice. (Why?)

Lattice diagram for S = {1,2,3}

- What is L(S)?
- L(S) consists of the following sets:
  - ∅ : the empty set
  - S : the set S itself
  - {1}, {2}, {3}
  - {1,2}, {1,3}, {2,3}

$$\begin{tikzcd}
 & S=\{1,2,3\} & \\
\{1,2\} \arrow[ur] & \{1,3\} \arrow[u] & \{2,3\} \arrow[ul] \\
\{1\} \arrow[u] \arrow[ur] & \{2\} \arrow[ul] \arrow[ur] & \{3\} \arrow[ul] \arrow[u] \\
 & \{\varnothing\} \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

BLP system state

- The state of a system is described by the 4-tuple (b,M,f,H):
- **Current access set** b:
  - Triples (subject, object, operation).
  - The triple (S_i,O_j,a) is interpreted as "S_i is *currently* doing a with respect to O_j".
  - The only triples that exist are allowed ones, otherwise the activity couldn't take place.
- **Access matrix** M:
  - As described earlier, this indicates which operations a subject can perform an object. Each element of the matrix is labelled M[S_i,O_j] and contains a list of allowed actions.

- **Level function** f: This consists of three mappings with overall responsibility for assigning security levels to each subject and to each object.
  - $f_o(O_i)$ produces the classification level of O_i.
  - $f_s(S_j)$ produces the security clearance of S_j.
  - $f_c(S_j)$ produces the *current security level* of S_j. We necessarily have $f_c(S_i) \leq f_s(S_i)$, so that a subject can operate at a lower security clearance than their maximum.
- **Hierarchy H:**
  - A directed rooted tree with the nodes being the objects.
  - This ties into the lattice.
  - The BLP model requires that the security level of an object dominates the security level of its parent.
  - Domination effectively means must be higher.

BLP properties: 2 Mandatory

- The mandatory properties of BLP are characterised by the phrase:

  “No write down, no read up!” **(ss and *)**

- As a whole the properties are designed to protect against unauthorized disclosure of information.

- **ss – property:**

  (S_i, O_j, read) can be in b iff $f_o(O_j) \leq f_c(S_i)$.

- Turning this around: The state (b,M,f,H) has the ss–property if, for every $(S_i, O_j, \text{read}) \in b$, we have that $f_o(O_j) \leq f_c(S_i)$.

- *** – property:**

  (S_i,O_j,append) can be in b iff $f_c(S_i) \leq f_o(O_j)$.

  (S_i,O_j,read/write) can be in b iff $f_c(S_i)=f_o(O_j)$.

- Notice that we have used = in the last line.

  We effectively define = by

  A = B iff (A ≤ B AND B ≤ A)

- Again we can turn this around to determine if the current state (b,M,f,H) has the * – property.

BLP properties: 1 discretionary

- **ds – property.**
- This is designed to capture the idea that permission may be passed from an authorised subject to another, level authorised, subject.
- (S_i,O_j,a) can be in b only if $a \in M[S_i,O_j]$.
- This can also be turned around to determine if the current state (b,M,f,H) has the ds – property.

An example

Top secret (TS)	Tam, Tom	Personnel files
Secret (S)	Sal, Sam	Email files
Confidential (C)	Cam, Cal	Activity log files
Unclassified (UC)	Uma, Una	Phone lists

- Cam and Cal cannot read personnel files, that would be reading up.
- Tam, Sam and Cam can all read the activity log files, if the access control matrices allow them to.
- Tam and Tom cannot write to the activity log files, that would be writing down.
- Uma and Una can write to the activity log files, if the access control matrices allow them to.

Discretionary example …

Top secret (TS)	Tam	Personnel file
Secret (S)	Sam	Email file
Confidential (C)	Cam	Activity log
Unclassified (UC)	Uma	Phone list

	Personnel file	Email file	Activity log	Phone list
Tam	Read, Write			
Sam		Read		
Cam				
Uma			Write	

Tam can read and write the Personnel file.
Sam cannot currently write the Email file.
Cam cannot do anything.
Uma cannot read the Phone list.

Criticism of McLean

- What happens if we
  - downgrade all subjects to lowest security level
  - downgrade all objects to lowest security level
  - enter all access rights in the ACM M
- Is the system secure?

Tranquility

- Consider a system with s_1,s_2,o_1,o_2
  - $f_s(s_1)=f_c(s_1)=f_o(o_1)$=high
  - $f_s(s_2)= f_c(s_2)= f_o(o_2)$ =low
- And the following execution:
  - s_1 gets access to o_1, read something, release access, then change current level to low, get write access to o_2, write to o_2
- Every state is secure
- **Solution: tranquillity principle**: subject cannot change current levels, or cannot drop to below the highest level read so far

Covert channels

- is a type of attack that creates a capability to transfer information objects between processes that are not supposed to be allowed to communicate by the computer security policy.
- Require two active agents, one at a low level and the other at a high level and an encoding scheme to pass on information about the high level to the low level

- Low-level subject s_1 creates object o
- High-level accomplice s_2 either
  - reclassifies o to its own level (Message 1)
  - leaves o unchanged (Message 0)
- s_1 tries to access o, which is either
  - success (Message 0)
  - access denied (Message 1)
- One bit of information is transmitted $s_2 \rightarrow s_1$

- The concern is with subjects not users
    - Users are trusted (must be trusted) not to disclose secret information outside of the computer system
    - Subjects are not trusted because they may have Trojan Horses embedded in the code they execute
- Covert channels are typically noisy but information theory techniques can be used to achieve error-free communication

Limitations of BLP

- only deals with confidentiality, not integrity
  - Confidentiality is not important as integrity in many situations
  - Limits the access and sharing of information
- assumes a fixed rights
  - assumes tranquillity
  - no model for access management
  - no model for policy making

Integrity based access control

- A different type of protection is integrity based access control.
  - In such models we protect against unauthorised modification of information.
- Notice how, for both models, the definitions hinge on the presence of the word *unauthorised*.

The Biba model

- The classical integrity based access control model is the Biba model (1977).
- Much of the basis for the model is the same as BLP.
- The access modes can be extended to include an **Invoke** instruction:

  {**Modify** (Write), **Observe** (Read), **Execute**, **Invoke** (subject to subject communication/use)}

- The rules to provide the appropriate policies are, in some sense, the reverse (or dual) of those for BLP.

  “No write up, no read down!”

- Biba is important now because a modified version of it is used in Vista and Windows 7.
  - Vista was probably the first commercial use of Biba.
- Microsoft has something called MIC: Mandatory Integrity Control.

Read http://www.symantec.com/connect/articles/introduction-windows-integrity-control

- Each object is assigned one of six integrity levels: Untrusted, Low, Medium, High, System and Installer.
  - Documentation at msdn.**microsoft**.com notes that in the absence of an integrity label, a medium label is assigned.
- Vista has some sort of token-based system for executing privileges and maintaining integrity.

- Expanding on the phrase **“No write up, no read down!”** we have:
  - Simple integrity.
  - Integrity confinement.
  - We also have the invocation property.
- Integrity confinement:
  - A subject can modify an object if the integrity level of the subject dominates the integrity level of the object. This is the **no write up** policy.
    - Integrity is to do with how much you can rely on something.
    - If A, as a process say, is trusted less then B as a resource, then B should not be modified on the basis of A. We shouldn’t contaminate B.

- Simple integrity:
  - A subject can read an object only if the integrity level of the subject is dominated by the integrity level of the object. This is the **no read down** policy.
    - Juries in court cases are sometimes told to disregard something that has been said, or to ignore some evidence.
      - Humans tend to take information into account whether they have been told to disregard it or not, so the **no read down** policy would imply the jury would never see the untrustworthy evidence.
  - Effectively this means a subject doesn’t trust information with a lower integrity level, so it shouldn’t even be influenced by it.

- The Invocation property:
  - A subject S_1 can invoke/execute/use another subject S_2 only if the integrity level of S_1 dominates the integrity level of S_2.
  - In other words, a process cannot use a process or entity that has higher integrity than it does.

Clark-Wilson

- Clark-Wilson is another integrity based access model.
- It’s also an accountability model providing a framework for addressing security requirements in primarily commercial applications.
  - Many early access control models were driven by the military, and even the title of Clark and Wilson’s work suggests a different direction: “*A Comparison of Commercial and Military Computer Security Policies.*”
  - [SB18, Figure 27.5] reproduces a diagram from Clark-Wilson that summarises the integrity rules in Clark-Wilson.

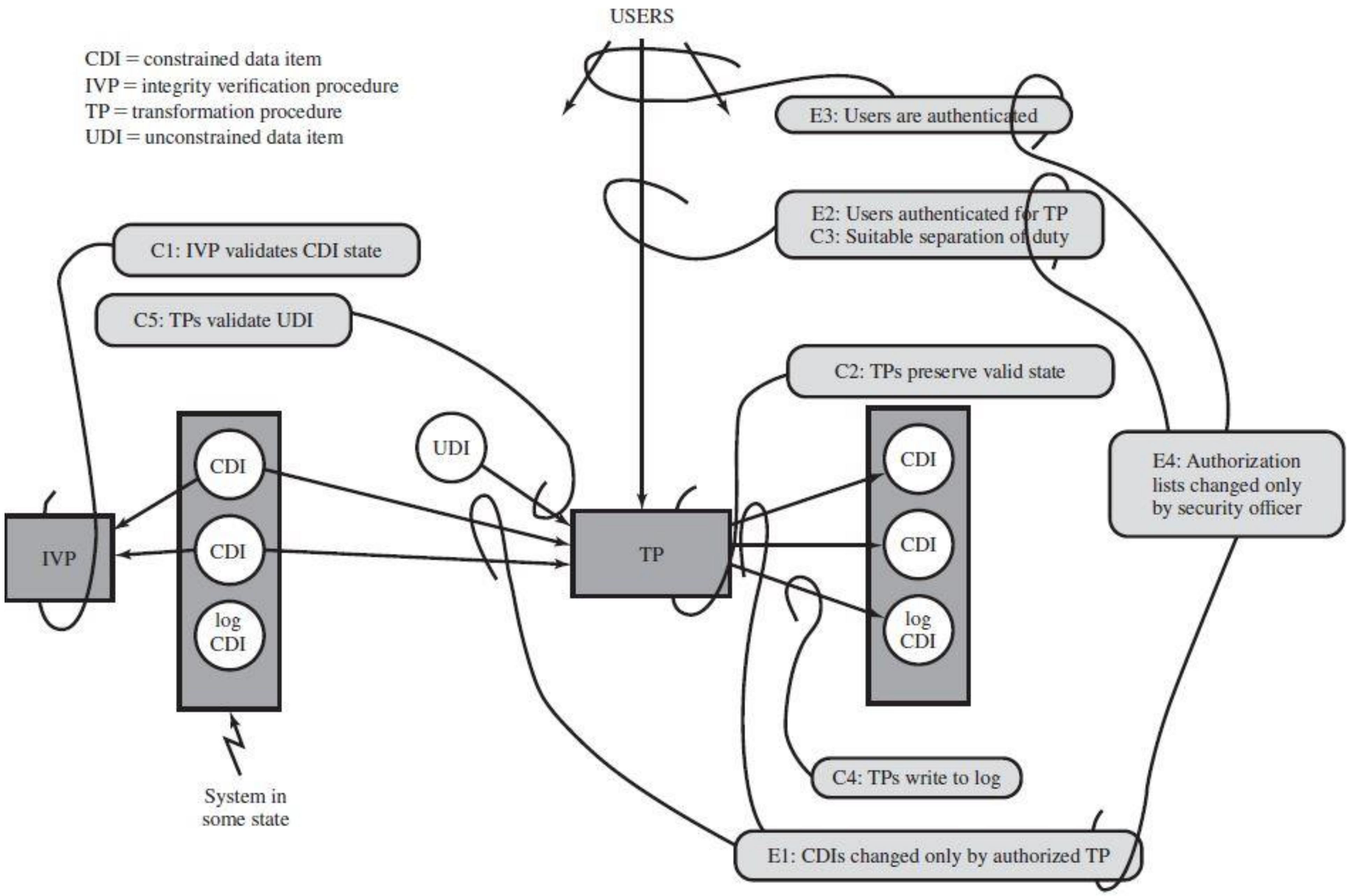

USERS
CDI = constrained data item
IVP = integrity verification procedure
TP = transformation procedure
UDI = unconstrained data item
E3: Users are authenticated
E2: Users authenticated for TP
C3: Suitable separation of duty
C1: IVP validates CDI state
C5: TPs validate UDI
C2: TPs preserve valid state
UDI
CDI
CDI
log CDI
IVP
TP
CDI
CDI
log CDI
E4: Authorization lists changed only by security officer
System in some state
C4: TPs write to log
E1: CDIs changed only by authorized TP

Figure 27.5 **Summary of Clark–Wilson System Integrity Rules**

Integrity and confidentiality

- What happens if we combine Biba and BLP?
- **“No read up, no write down!”**
- **“No write up, no read down!”**

- **Hmm ...**

  ... “Read and Write across”? ☹

Lipner’s model

- The difference lies in the clearances and classifications.
- We have a lattice here rather than the simple levels, and we have a lattice associated with integrity and a lattice associated with confidentiality.
- This can be illustrated using an example pulled straight out of [B18, Chapter 6].
  - Bishop demonstrates how some of the confidentiality categories can be eliminated by the introduction of integrity categories.
  - There are a lot of possible levels, but not all of them are needed/used in practice.

Labels in a combined model …

- The language used to describe the levels differs between a confidentiality setting and an integrity setting.
- For example, the sensitivity levels may be:

top secret	very reliable	totally trusted
Secret	reliable	mostly trusted
Confidential	a little bit reliable	somewhat trusted
Unclassified	unreliable	untrusted

- **BLP**: Confidentiality based so would likely use the first column.
- **Biba**: Integrity based so would so likely use the second and third columns.

Classifications for Confidentiality

- **CAM**: Audit Manager: System auditing and management functions.
- **CSL**: System Low: The lowest.

Categories for Confidentiality

- **CP**: Production: Production code and data.
- **CD**: Development: Production programs under development and testing, but not yet in production use.
- **CSD**: Systems Development: System programs under development, but not yet in production use.

Classifications for Integrity

- **ISP:** System programs.
- **IO**: Production programs and development software.
- **ISL**: System Low: The lowest.

Categories for Integrity

- **ID**: Development: Development entities.
- **IP**: Production: Production entities.

Subjects → Users

Users	Confidentiality Clearance	Integrity Clearance / Trust
Ordinary user	(CSL, {CP})	(ISL, {IP})
Application developer	(CSL, {CD})	(ISL, {ID})
System programmer	(CSL, {CSD})	(ISL, {ID})
System controller	(CSL, {CP, CD,CSD}) and downgrade privilege	(ISP, {IP, ID})
System manager, Auditor	(CAM, {CP, CD,CSD})	(ISL, ∅)
Repair	(CSL, {CP})	(ISL, {IP})

Objects → Code, data, program, logs

Objects	Confidentiality Clearance	Integrity Clearance / Trust
Development code/test data	(CSL, {CD})	(ISL, {ID})
Production code	(CSL, {CP})	(IO, {IP})
Production data	(CSL, {CP})	(ISL, {IP})
Software Tools	(CSL, ∅)	(IO, {ID})
System programs	(CSL, ∅)	(ISL, {IP, ID})
System programs in modification	(CSL, {CSD})	(ISL, {ID})
System and application logs	(CAM, {appropriate categories})	(ISL, ∅)
Repair	(CSL, {CP})	(ISL, {IP})

So ...

- The classifying and categorizing is carried out so typical behaviour is appropriately represented.
- For example, an ordinary user can alter production data, but cannot change production code.

The Chinese-Wall model

- Brewer and Nash (1989) proposed this model to handle the conflicts of interest that occur in many commercial environments.
- In this case ...
  - ... the subjects are analysts or consultants.
  - ... the objects are information sets for companies.

EQUITY ANALYST
I'LL RATE YOUR STOCK A "MUST BUY NOW" IF YOU GIVE US YOUR INVESTMENT BANKING BUSINESS.
scottadams@aol.com
www.dilbert.com
AREN'T YOU SUPPOSED TO HAVE A CHINESE WALL BETWEEN THOSE TWO BUSINESSES?
© 2001 United Feature Syndicate, Inc.
1/30/01
AM I TOO EARLY?
USE THE DOOR, IDIOT.

Dilbert: 30-Jan-2001

The Chinese Wall Model

- Hybrid model: addresses integrity and confidentiality
- Addresses conflict of interest
  - Models a consultancy business where analysts have to make sure that no conflicts arise when dealing with different clients (companies)
  - Conflicts arise when clients are direct competitors in the same market, or because of the ownership of companies

Model elements

- A set of subjects S: active entities interested in accessing protected objects
- A set of companies C
- A set of objects O:
  - individual data items, each about a company
  - The objects concerning the same company are called company datasets (DS)
  - The function y: O -> C gives the company dataset for each object
- The function x: O -> L(C) gives the conflict of interest classes for each object

- The security label of an object o is the pair (x(o), y(o))
- An object is sanitised if x(o) is empty
- Conflicts of interest may also arise from objects that have been accessed in then past.
- Let N(s,o) be true, if subject s has had access to object o, and false, if subject s never had access to object o

ss- property

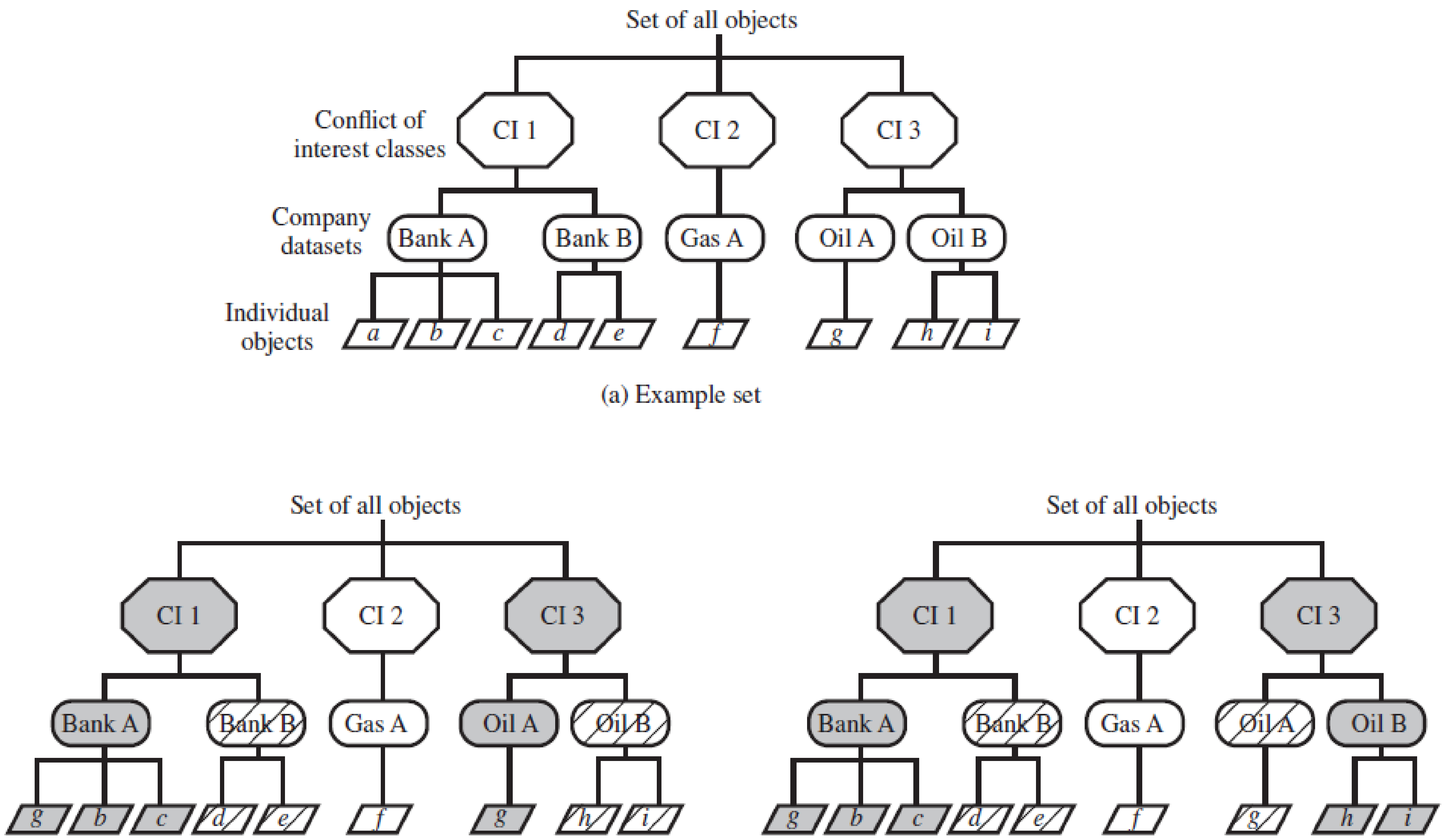
- A subject s is granted access to an object o only if for all objects o’ with N(s,o’)=true: either y(o) = y(o’) or y(o) does not belong to x(o’)
- That is, access is granted only if the object requested belongs to:
    - a company dataset already held by the subject (the analyst), or
    - an entirely different conflict of interest class.

Star property

- A subject s is granted write access to an object o only if s has no read access to an object o’ with y(o)≠y(o’) and x(o’) is not empty
- That is, write access to an object is only granted if no other object can be read which is in a different company dataset and contains unsanitised information.

The Chinese Wall Model

- Not a true multi-level secure model
- The history of a subject's access determines access control
- Subject 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 conflict of interest



(a) Example set

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

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

Figure 27.6 in [SB18]