Module II

Security policies and models: confidentiality policies, Bell-LaPadula model, Integrity policies, Biba model, Clark-Wilson models, Chinese wall model, waterfall model

GOALS OF CONFIDENTIALITY POLICIES

Confidentiality Policies emphasize the protection of confidentiality.

Confidentiality policy also called information flow policy prevents unauthorized disclosure of information.

Example: Privacy Act requires that certain personal data be kept confidential. E.g., income tax return info only available to IRS and legal authority with court order. It limits the distribution of documents/info.

Information security is an important issue in modern world. For this purpose many security models have been developed to enforce it.

What is a Security Model?

- A model describes the system
  - e.g., a high level specification or an abstract machine description of what the system does
- A security policy
  - defines the security requirements for a given system
- Verification techniques that can be used to show that a policy is satisfied by a system
- System Model + Security Policy = Security Model

Information Flow Models

In reality, there are state transitions

Key is to ensure transitions are secure

Models provide rules for how information flows from state to state

State Machine Model State is a snapshot of the system at one moment in time.

State transition is the change to the next state.

If all the state transitions in a system are secure and if the initial state of the system is secure, then every subsequent state will also be secure, no matter what input occurs.

Bell-LaPadula Model: A MAC Model for Achieving Multi-level Security

Bell-LaPadula model is a security method created for the US government to preserve the confidentiality of information

The BLP Security Model

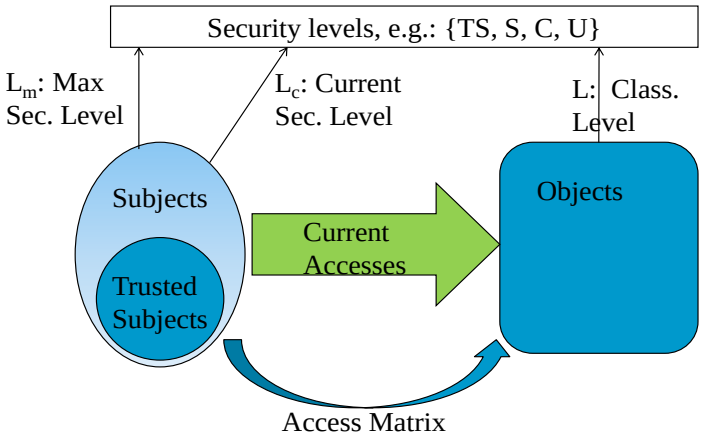
- A computer system is modeled as a state-transition system

  - There is a set of subjects; some are designated as trusted.
  - Each state has objects, an access matrix, and the current access information.
  - There are state transition rules describing how a system can go from one state to another
  - Each subject s has a maximal security level $L_m(s)$, and a current security level $L_c(s)$
  - Each object has a classification level
- There are security classifications or security levels
  - Users/principals/subjects have security clearances
  - Objects have security classifications
- Example of security levels
  - Top Secret
  - Secret
  - Confidential
  - Unclassified
- In this case Top Secret > Secret > Confidential > Unclassified
- **clearance level**,- the ability to have important information. The clearance level is measured from lowest to highest and includes: Confidential, Secret and Top Secret. The clearance level is assigned to a person you trust. The higher the clearance, the greater you trust them.
- Security goal (confidentiality): ensures that information do not flow to those not cleared for that level

Bell-LaPadula is a form of **multilevel security.**

The general statement of the requirement for multilevel security is that a subject (e.g military-field marshal –the highest rank) at a high level may not convey information to a subject at a lower(e.g. subedar) or incompatible level unless that flow accurately reflects the will of an authorized user

Elements of the BLP Model



CS526 Topic 17: BLP 11

The *Bell-LaPadula* state machine model enforces confidentiality.

The Bell-LaPadula model uses **mandatory access control** to enforce the DoD multilevel security policy. For a subject to access information, he must have a clear need to know and meet or exceed the information's classification level.

It also supports **discretionary access control** by checking access rights from an access matrix.

The Bell-LaPadula model is defined **by the following properties:**

- **Simple security property (ss property)**—This property states that a subject at one level of confidentiality is not allowed to read information at a higher level of confidentiality. This is sometimes referred to as "no read up."

  A person in one classification level, cannot read data in a higher classification level. If you have a Secret clearance, then you cannot read objects with a label of Top Secret. This is also known as No Read Up.

- **Star * security property**—This property states that a subject at one level of confidentiality is not allowed to write information to a lower level of confidentiality. This is also known as "no write down."

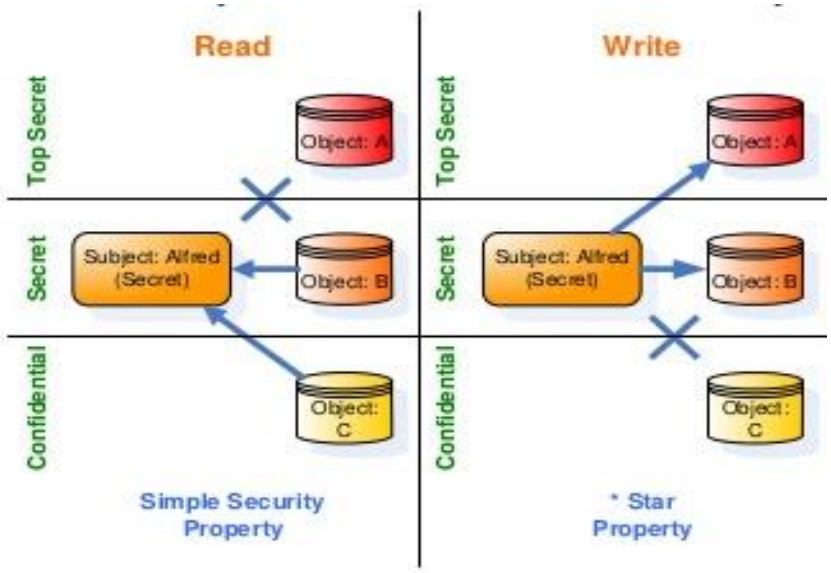
  A person in a higher classification level cannot write messages to someone in a lower classification level. If you have a clearance of Top Secret, then you cannot write messages to someone with a Secret clearance. This is known as No Write Down

- The Discretionary Security Property (ds-property) - An individual (or role) may grant to another individual (or role) access to a document based on the owner's discretion, constrained by the MAC rules.

Bell-LaPadula Model in Mathematics

The Simple Security Property: Subject S with clearance (LS , CS) may be granted read access to object O with classification (LO , CO) only if (LS , CS) $\geq$ (LO , CO).

The $*$-Property: Subject S with clearance (LS , CS) may be granted write access to object O with classification (LO , CO) only if (LS , CS) $\leq$ (LO , CO)



If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, and the * property, then every state of the system is secure

Example

security level	*subject*	*object*
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Alice	Telephone Lists

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

Bell-LaPadula model has two major limitations:

- It provides confidentiality only. (no integrity, authentication ,etc.)
- It provides no method for management of classifications:
  - It assumes all data are assigned with a classification
  - It assumes that the data classification will never change.

INTEGRITY POLICIES

- Integrity refers to the trustworthiness of data or resources.
- Integrity is usually defined in terms of preventing improper or authorized change to data.
- There are three main policies of integrity:
    - **Preventing unauthorized users from making modifications to data or programs.**
    - **Preventing authorized users from making improper or unauthorized modifications.**
    - **Maintaining internal and external consistency of data and programs.**

Integrity Levels

Integrity levels are defined by labels, consisting of two parts:

- a classification
- a set of categories.

- Integrity levels are given to the subjects and objects in the system.
- Integrity labels tell the degree of confidence that may be placed in the data.

Classification of Integrity

A classification is an element of hierarchical set of elements.

- It consists of these elements:
    - Crucial (c)
    - Very Important (VI)
    - Important (I)
- The relationship of elements is:
    - C > VI > I
- Each integrity level will be represented as L = (C, S) where:
    - L is the integrity level
    - C is the classification
    - S is the set of categories.

An example of two categories are category X = {Detroit, Chicago, New York} and category Y = {Detroit, Chicago}.

- The integrity levels then form a dominance relationship.
- Integrity level $L_1 = (C_1, S_1)$ dominates ($\geq$) integrity level $L_2 = (C_2, S_2)$ if and only if this relationship is satisfied: $C_1 \geq C_2$ and $S_1 \supseteq S_2$

BIBA MODEL

- The Biba integrity model was published in 1977 at the Mitre Corporation, one year after the Bell La-Padula model was published.
- The primary motivation for creating this model is the inability of the Bell-LaPadula model to deal with integrity of data.

- The Biba model addresses the problem with the star property of the Bell-LaPadula model, which does not restrict a subject from writing to a more trusted object.

The Biba model on the other hand ignores confidentiality all together and deals only with integrity. So, the main goal of the Biba model is to prevent unauthorized users from making modifications to a particular document.

Also it prevents authorized users from making improper modifications in a document. So, this Biba model is incorporated in Microsoft windows vista operating system.

Subjects and Objects

Like other models, the Biba model supports the access control of both subjects and objects.

- **Subjects** are the active elements in the system that can access information (processes acting on behalf of the users).
- **Objects** are the passive system elements for which access can be requested (files, programs, etc.).

Each subject and object in the Biba model will have a integrity level associated with it.

Access Modes

The Biba model consists of the following access modes:

Modify: the modify right allows a subject to write to an object. This mode is similar to the write mode in other models.

Observe: the observe right allows a subject to read an object. This command is synonyms with the read command of most other models.

Invoke: the invoke right allows a subject to communicate with another subject.

Execute: the execute right allows a subject to execute an object. The command essentially allows a subject to execute a program which is the object.

Biba Policies

- The Biba model is actually a family of different policies that can be used.
- The model supports both mandatory and discretionary policies.

The Mandatory Policies:

- Strict Integrity Policy
- Low-Watermark Policy for Subjects
- Low-Watermark Policy for Objects
- Low-Watermark Integrity Audit Policy
- Ring Policy

The Discretionary Policies:

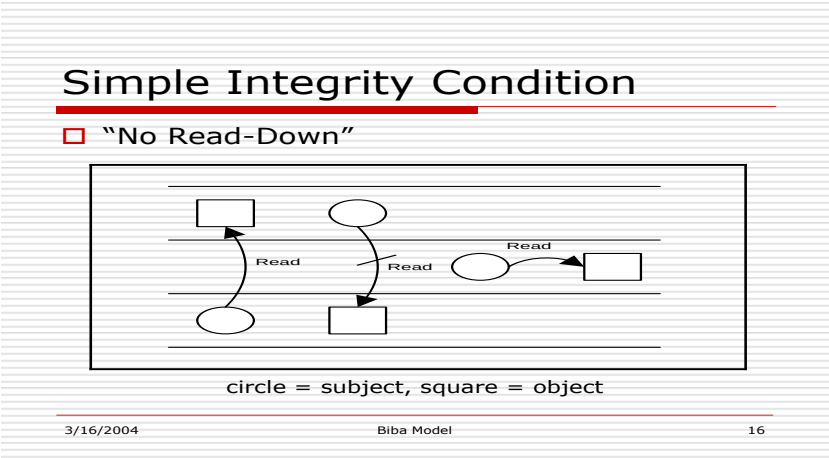
- Access Control Lists
- Object Hierarchy
- Ring

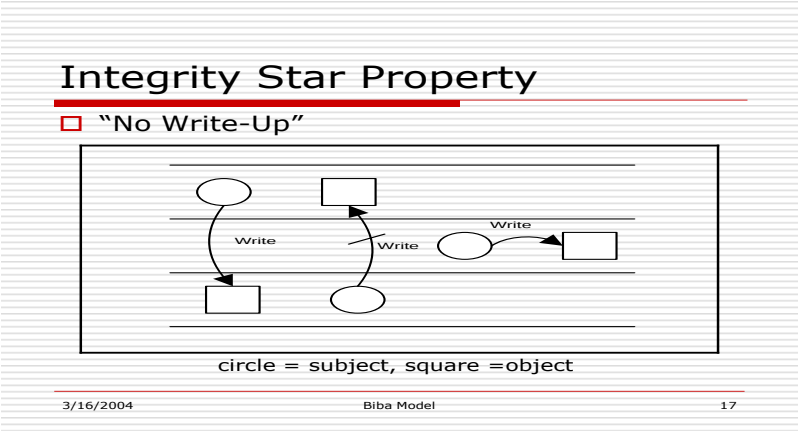
Strict Integrity Policy

The Strict Integrity Policy is the first part of the Biba model. The policy consists of:

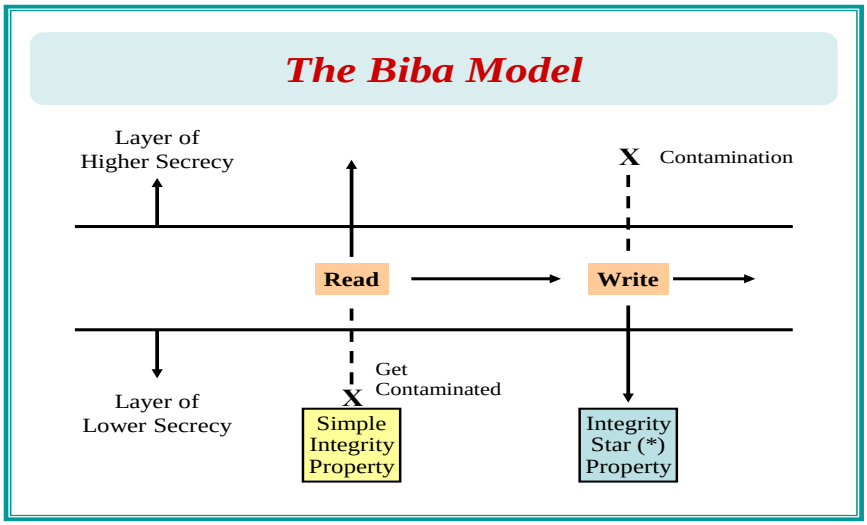
1. **Simple Integrity Condition:** $s \in S$ can observe $o \in O$ if and only if $i(s) \leq i(o)$ ("no read-down"). That means high integrity subject cannot read low integrity objects.

   i-Integrity level
2. **Integrity Star Property:** $s \in S$ can modify $o \in O$ if and only if $i(o) \leq i(s)$ ("no write-up"). Subject cannot move low integrity data to high integrity environment.
3. **Invocation Property:** $s_1 \in S$ can invoke $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$. That means a subject at one integrity level is prohibited from invoking or calling up a subject at higher level of integrity.





- When most people refer to the Biba model they are actually referring to the strict integrity model.
- This policy is the most common policy that used from the model.
- The strict integrity policy enforces "no write-up" and "no read-down" on the data in the system, which is the opposite of the Bell-LaPadula model.



So, what the Biba model defines is that the user can first of all read and write to any object within the same security class, and the user could write to a object in a lower security class, and read from an object present in a higher security class.

As an example, let us say the hierarchy in the military where you have a general right on top, then the captains and the privates who are right at the bottom of the hierarchy

Now, the Biba model allows read up meaning a document which is prepared by the general should be read by all, that is a document which is created by the general should be read by the captain as well as the privates.

However, no read down is permitted, that is a document written or modified by the privates at the lower end of the hierarchy should not affect the general's decision.

Low-Watermark Policy for Subjects

The low-watermark policy for subjects is a relaxed "no read-down".

The low-watermark policy for subjects contains these following rules:

1. **Integrity Star Property:** $s \in S$ can modify $o \in O$ if and only if $i(o) \leq i(s)$ ("no write-up").
2. A subject may examine any object. If $s \in S$ examines $o \in O$ then $i'(s) = min(i(s),i(o))$, where $i'(s)$ is the subjects integrity level after the read.
3. **Invocation Property:** $s_1 \in S$ can invoke $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

Low-Watermark Policy for Objects

The low-watermark policy for objects is a relaxed "no write-down".

The following rules make up the low-watermark for objects policy:

1. $s \in S$ can modify any $o \in O$ regardless of integrity level.
2. If $s \in S$ modifies $o \in O$ then $i'(o) = min(i(s),i(o))$, where $i'(o)$ is the objects integrity level after it is modified.

Ring Policy

The ring policy is the last mandatory policy in the Biba model. Integrity labels used for the ring policy are fixed similar to those in the strict integrity policy.

The Ring Policy consists of the following rules:

1. Any subject can observe any object, regardless of integrity levels.
2. **Integrity Star Property:** $s \in S$ can modify $o \in O$ if and only if $i(o) \leq i(s)$ ("no write up").
3. **Invocation Property:** $s_1 \in S$ can invoke $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

Advantages:

The Biba model is it simple and easy to implement.

The Biba model provides a number of different policies that can be selected based on need.

Disadvantages:

The model does nothing to enforce confidentiality.

The Biba model doesn't support the granting and revocation of authorization.

To use this model all computers in the system must support the labeling of integrity for both subjects and objects. To date, there is no network protocol that supports this labeling. So there are problems with using the Biba model in a network environment.

Bell-LaPadula versus Biba model

The Bell-LaPadula model is used to provide confidentiality. The Biba model is used to provide integrity. The Bell-LaPadula and Biba models are informational flow models because they are most concerned about data flowing from one level to another. Bell-LaPadula uses security levels and Biba uses integrity levels

CLARK-WILSON MODELS

It is an integrity model like biba model

- Here integrity is focuses on transactions instead of access rights.

Clark-Wilson Integrity Model Integrity defined by a set of constraints

Example: Bank –D today's deposits, W withdrawals, YB yesterday's balance, TB today's balance – Integrity constraint: D + YB –W

It focuses on **well formed transaction** move system from one consistent state to another and **separation of duties**

The three main rules of integrity models:

- Prevent unauthorized users from making modifications
- Prevent authorized users from making improper modifications (separation of duties)
- Maintain internal/external consistency (well-formed transaction)

Clark-Wilson model addresses each of these goals. Biba model only addresses the first goal.

- Clark-Wilson model enforces the three goals of integrity by using **access triple** (subject, software TP, and object), **separation of duties, and auditing**. It enforces integrity by using well-formed transactions (through access triple) and separation of user duties.
- Separation of duties: assigning different roles to different users.

Integrity goals of Clark–Wilson model:

- Prevent unauthorized users from making modification (Only this one is addressed by the Biba model).
- Separation of duties prevents authorized users from making improper modifications.
- Well-formed transactions: maintain internal and external consistency i.e. it is a series of operations that are carried out to transfer the data from one consistent state to the other.

The model uses the following elements:

- **Users:** Active agents.
- **Transformation Procedures (TPs):** Programmed abstract operations, such as read, write and modify.
    - TP takes as input a CDI or UDI and convert into valid CDIs
    - It must transition the system from one valid state to another valid state.
- **Constrained Data Item (CDI):** A data item whose integrity is to be preserved. Can only be manipulated by TPs.
- **Unconstrained Data Item (UDI):** Data items outside of the control area of the modeled environment such as input information. Can be manipulated by users via primitive read and write operations.
- **Integrity Verification Procedure (IVP):** Check the consistency of CDIs with external reality.
    - Assure all CDI s are valid

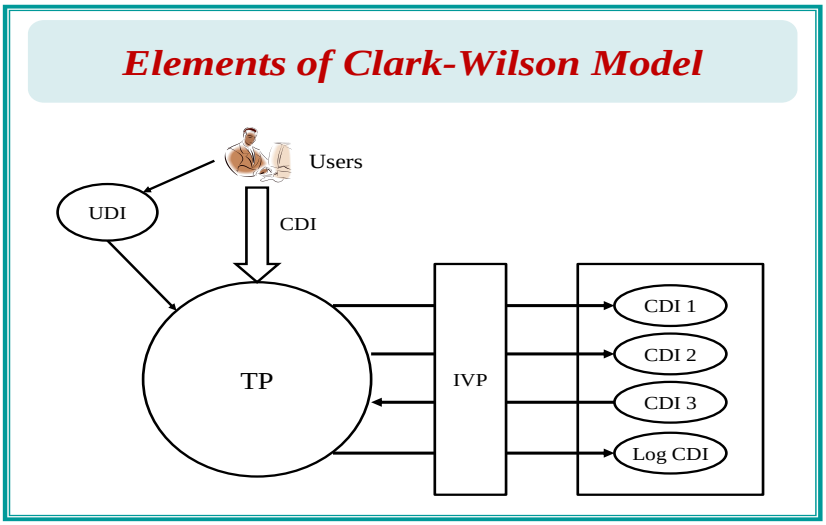
The model contains a number of basic constructs that represent both data items and processes that operate on those data items. The key data type in the Clark-Wilson model is a Constrained Data Item (CDI). An Integrity Verification Procedure (IVP) ensures that all CDIs in the system are valid at a certain state. Transactions that enforce the integrity policy are represented by Transformation Procedures (TPs). A TP takes as input a CDI or Unconstrained Data Item (UDI) and produces a CDI. A TP must transition the system from one valid state to another valid state. UDIs represent system input (such as that provided by a user or adversary). A TP must guarantee (via certification) that it transforms all possible values of a UDI to a "safe" CDI

Example

Consider a Banking system, here CDIs are account balance

TPs are- Depositing money, withdrawing money etc

UDI- Gifts given to account holders



Another Example

In a bank ATM, numbers entered at the keyboard are UDIs so cannot be input to TPs as such. TPs must validate numbers (to make them a CDI) before using them; if validation fails, TP rejects UDI

Certification or Enforcement Rule

The model consists of two sets of rules: Certification Rules (C) and Enforcement Rules (E). The nine rules ensure the external and internal integrity of the data items.

Certification Rule: How should the system behave?

Enforcement Rule: How do we make the system behave the way we want?

Certification Rule

C1: When any IVP is run, it must ensure all CDIs are in valid state

C2: A TP must transform a set of CDIs from a valid state to another valid state

C3: Duties are separated: Assigning different roles different users

C4: All access are logged: an audit log is maintained over external transaction

C5: Unconstrained data item are validated to produce a valid CDI

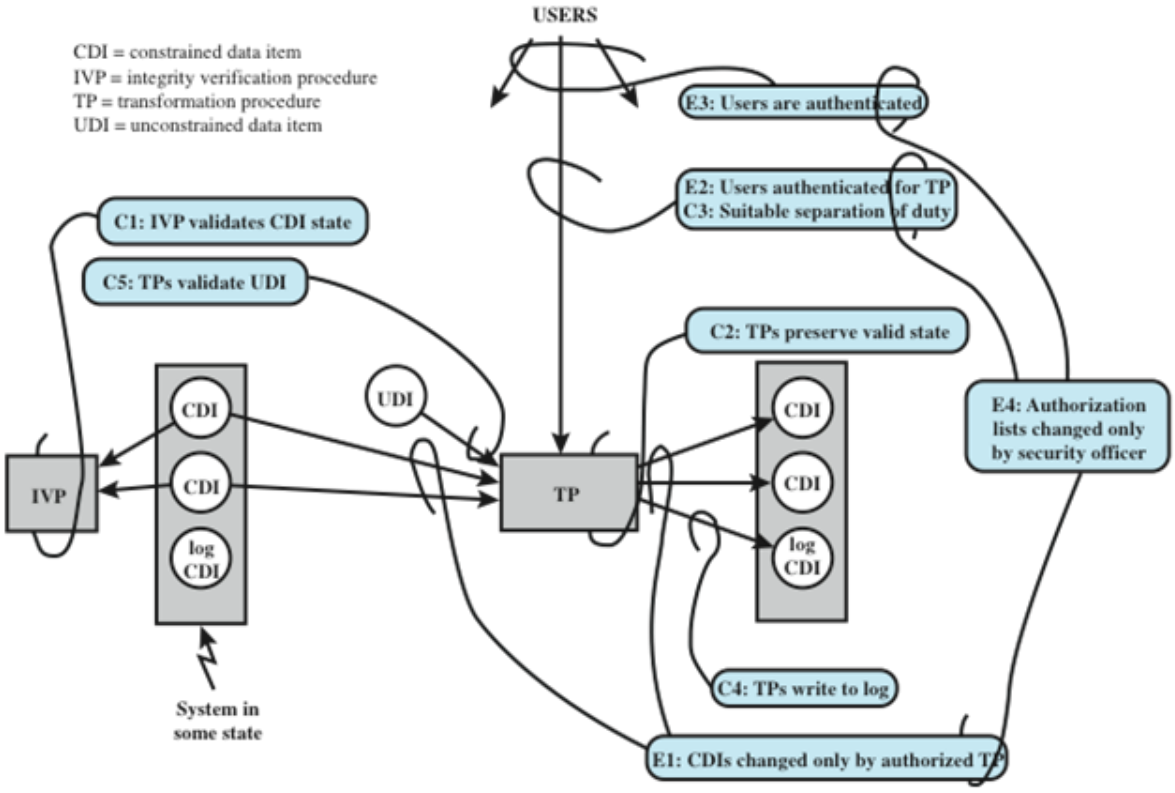
Enforcement Rule

E1 **CDIs changed only by authorized TP**
E2 **Users authorized to TP and CDI**
E3 **Users are authenticated**
E4 **Authorizations changed only by security officer**

Clark-Wilson versus Biba

In Biba's model, UDI to CDI conversion is performed by trusted subject only (e.g., a security officer), but this is problematic for data entry function.

In Clark-Wilson, TPs are specified for particular users and functions. Biba's model does not offer this level of granularity.



CHINESE WALL MODEL

Chinese Wall Model which addresses a conflicts of interest problem.

Strictly speaking, this is **not an integrity policy, but an access control confidentiality policy**.

The main idea is that you are able to access any information you want from any company but once you access that information, you are no longer allowed to access information from another company within that class of companies.

The security policy builds on three levels of abstraction.

- **Objects** such as files. Objects contain information about only one company.
- **Company groups** collect all objects concerning a particular company.
- **Conflict classes** cluster the groups of objects for competing companies. That means contain the CDs of companies in competition.

For example, consider the following conflict classes: { Dialog, Mobitel, Airtel }

{ Central Bank, HNB, HSBC }

{ Microsoft }

We have a simple access control policy: A subject may access information from any company as long as that subject has never accessed information from a different company in the same conflict class.

For example, if you access a file from Dialog, you subsequently will be blocked from accessing any files from Mobitel or Airtel.

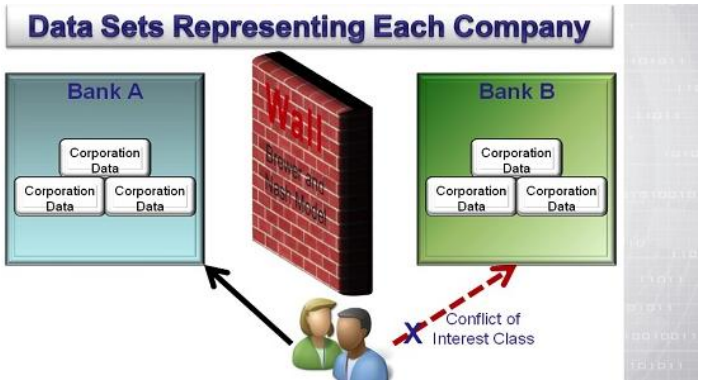
You are free to access files from companies in any other conflict class. Notice that permissions change dynamically. The access rights that any subject enjoys depends on the history of past accesses.

Chinese wall model is mainly focus on **conflicts of interest (COI)**

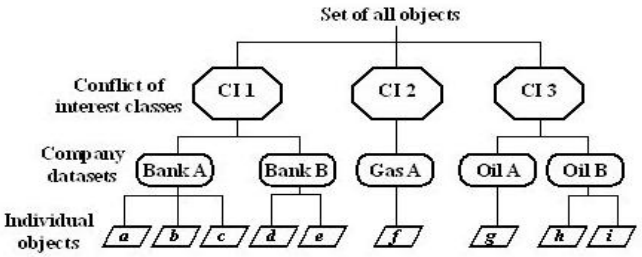
COI- conflict classes contains CDs of competitive companies

- Principle: Users should not access the confidential information of both a client organization and one or more of its competitors.
- How it works
  - Users have no "wall" initially.
  - Once any given file is accessed, files with competitor information become inaccessible.
  - Unlike other models, access control rules change with user behavior

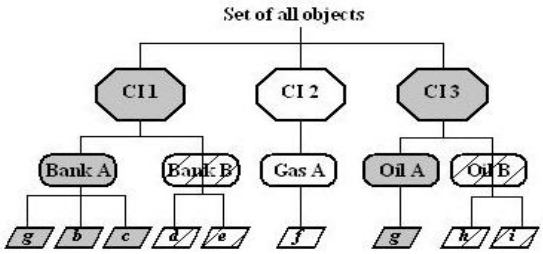
- A "wall" is defined by a set of rules that ensures no subject from one side of the wall can access objects on the other side of the wall.



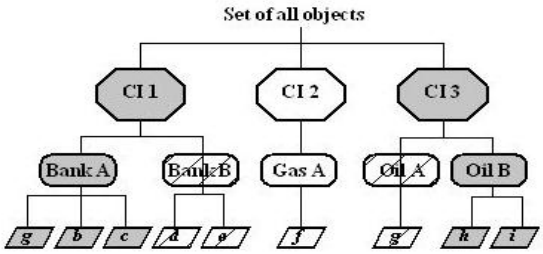
All corporate information is stored in a hierarchically arranged filing system as shown in Figure



(a) Example set



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



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

There are three levels of significance: at the bottom level, individual pieces of information are considered, each representing a single corporation.

This information is stored in filed called as objects; at the medium level, all the objects from the same corporation are grouped into one company dataset; at the top level, all these company datasets from competing corporations are grouped together .This group is known as **conflict of interest class**

There are two things that are always associated with the name of the object:

- Company dataset
- Conflict of Interest class

Thus, in consideration of the Bank-A, Bank-B, Gas Company-A, Oil Company-A and Oil Company-B datasets mentioned previously, a new user may freely choose to access whatever datasets he likes; concerning the computer a new user cannot have any conflicts since they does not possess any information. Sometime later, however, such a conflict may exist.

Suppose the user requests to get the data for Oil Company -A. So it will get the access to the data since he/she is a new user and thus no conflict exists. Now if after sometime the same user asks for the data of Bank-A then he/she will be granted the access to the data since they belong to different conflict of interest classes. Up till this point everything is fine since there is no conflict. Now if the user requests for accessing the data of Oil Company-B then it the request will be denied since they belong to the same conflict of interest class.

A new user has complete freedom to access anything he wants to choose. After the user makes the initial choice, Chinese wall is built around the dataset for that user and the opposite or the wrong side of this wall can be considered as any dataset in the same conflict of interest class. The user always have access to the dataset in the different conflict of interest class but whenever he/she accesses some new data from a different COI then the Wall around him/her changes to include that dataset. So it can be said that combination of mandatory control and free choice is Chinese wall.

To enforce the Chinese wall policy, two rules are needed. To indicate the similarity with the two BLP rules, the authors gave them the same names. The first rule is the simple security rule:

Simple security rule: A subject S can read on object O only if

- O is in the same DS as an object already accessed by S, **OR**
- O belongs to a CI from which S has not yet accessed any information

***-property rule:** A subject S can write an object O only if

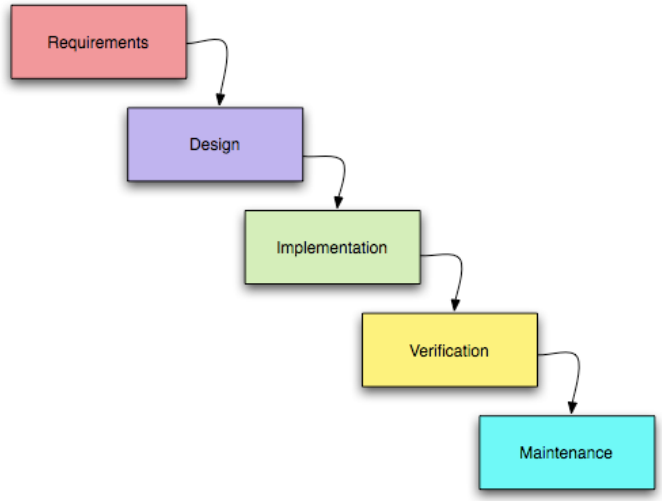
- S can read O according to the simple security rule, **AND**

Compare CW to Bell-LaPadula

- Fundamentally different
  - CW is based on access history, BLP is history-less
- BLP can capture CW state at any time, but cannot track changes over time
  - BLP security levels would need to be updated each time an access is allowed

WATERFALL MODEL

The Waterfall Model is the old method of structured system development. It's the base for all models although it has come under attack in recent years for being too rigid and unrealistic when it comes to quickly meeting customer's needs and development, the Waterfall Model is still widely used because of easy model for developers.



The Procedure of Waterfall Model for software development:

• **System Requirements:**

System Requirement refers to the consideration of all aspects of the targeted business function or process, with the goals of determining how each of those aspects relates with one another, and which aspects will be incorporated into the system. What is the essential thing needed in developing system

• **System Analysis:**

This step refers to the gathering of system requirements, with the goal of determining how these requirements will be accommodated in the system. Extensive communication between the customer and the developer is essential. Developer has to understand the exact requirement of user.

• **System Design:**

Once the requirements have been collected and analyzed, it is necessary to identify in detail how the system will be constructed to perform necessary tasks. More specifically, the System Design phase is focused on the data requirements, the software construction and the interface construction.

• **Coding:**

This step involves the creation of the system software. Requirements and systems specifications from the System Design step are translated into machine readable computer code.

• **Testing**

As the software is created and added to the developing system, testing is performed to ensure that it is working correctly and efficiently.

In every system development has its own strategy and methodology but where lies the security aspect in system development life cycle? This is the very big problem in system development in this paper we provide some security methods to enhance the water fall model.

Security in Design Phase:

In every phase we have to include the security enhancement. Even though it's very essential for every phase depth security features needed from the design phase onwards. In requirement gathering phase information fetching mostly happen between user and developer due to that need of security level will be less.

Apart from the user collecting information should be trust worthy as well as valid information should be taken for the system development.

In analysis phase, the information's what we obtain from the requirement phase that will give to the analysis phase. Collected information will be analysis used some valid documents materials, white papers, existing methods, etc. Information grouped into the structure form from the unstructured form. In the analysis phase itself we have to estimate what kind of security requirements need for our system. Security elements and features should be included in every aspect of the system like user, data, module, design, testing, etc.

Developers need to know secure software design principles and how they are employed in the design of resilient and trustworthy systems. Two essential concepts of design include abstraction and decomposition of the system using the architecture and constraints to achieve the security requirements obtained during the requirements phase. Most of the readers are probably familiar with these concepts.

Abstraction is a process for reducing the complexity of a system by removing unnecessary details and isolating the most important elements to make the design more manageable.

Decomposition is the process of describing the generalizations that compose an abstraction. One method, top-down decomposition, involves breaking down a large system into smaller parts. For object-oriented designs, the progression would be application, module, class, and method. Other secure software design principles are detailed in a multitude of books, white papers, web portals, and articles. Here we are providing some techniques to improve the SDLC (Software Development life cycle)

First thing is minimize the no of high consequence targets. Minimizes the number of actors in the system granted high levels of privilege, and the amount of time any actor holds onto its privileges. Ensures that

no single entity should have all the privileges required to modify, delete, or destroy the system, components and resources. Separation of domains makes separation of roles and privileges easier to implement.

Don't expose vulnerable or high-consequence components:

• Keep program data, executables, and configuration data separated .Reduces the likely hood that an attacker who gains access to program data will easily locate and gain access to program executables or control/configuration data.

• Segregate trusted entities from un trusted entities**,** Reduces the exposure of the software's high-consequence functions from its high-risk functions, which can be susceptible to attacks.

• Assume environment data is not trustworthy, reduces the exposure of the software to potentially malicious execution environment components or attacker-intercepted and modified environment data.

• Use only safe interfaces to environment resources; this practice reduces the exposure of the data passed between the software and its environment.

• Minimize the number of entry and exit points; this practice reduces the attack surface.