Database Security

Access control

Notes_Set3

Multi-Level Security, Inference Problems and Aggregation Issues

Mandatory Access Control

- **The discretionary access control techniques is an all-or-nothing method. A user either has or does not have a certain privilege.**
- The main drawback of **DAC** models is their vulnerability to malicious attacks, such as Trojan horses embedded in application programs
- Mandatory access control overcomes the shortcomings of discretionary access control.
- In the mandatory access control approach, access privileges cannot be granted or passed on by one user to another in an uncontrolled manner.
- **In many applications, and additional security policy is needed that classifies data and users based on security classes.**
- A well-defined security policy dictates which classes of data may be accessed by users at which clearance levels. The most popular method is known as the **Bell–LaPadula model**.
- Many of the commercial relational DBMSs do not currently provide for mandatory access control. However, government agencies, defense departments financial institutions, and intelligence agencies do require security mechanisms based on the mandatory control technique.

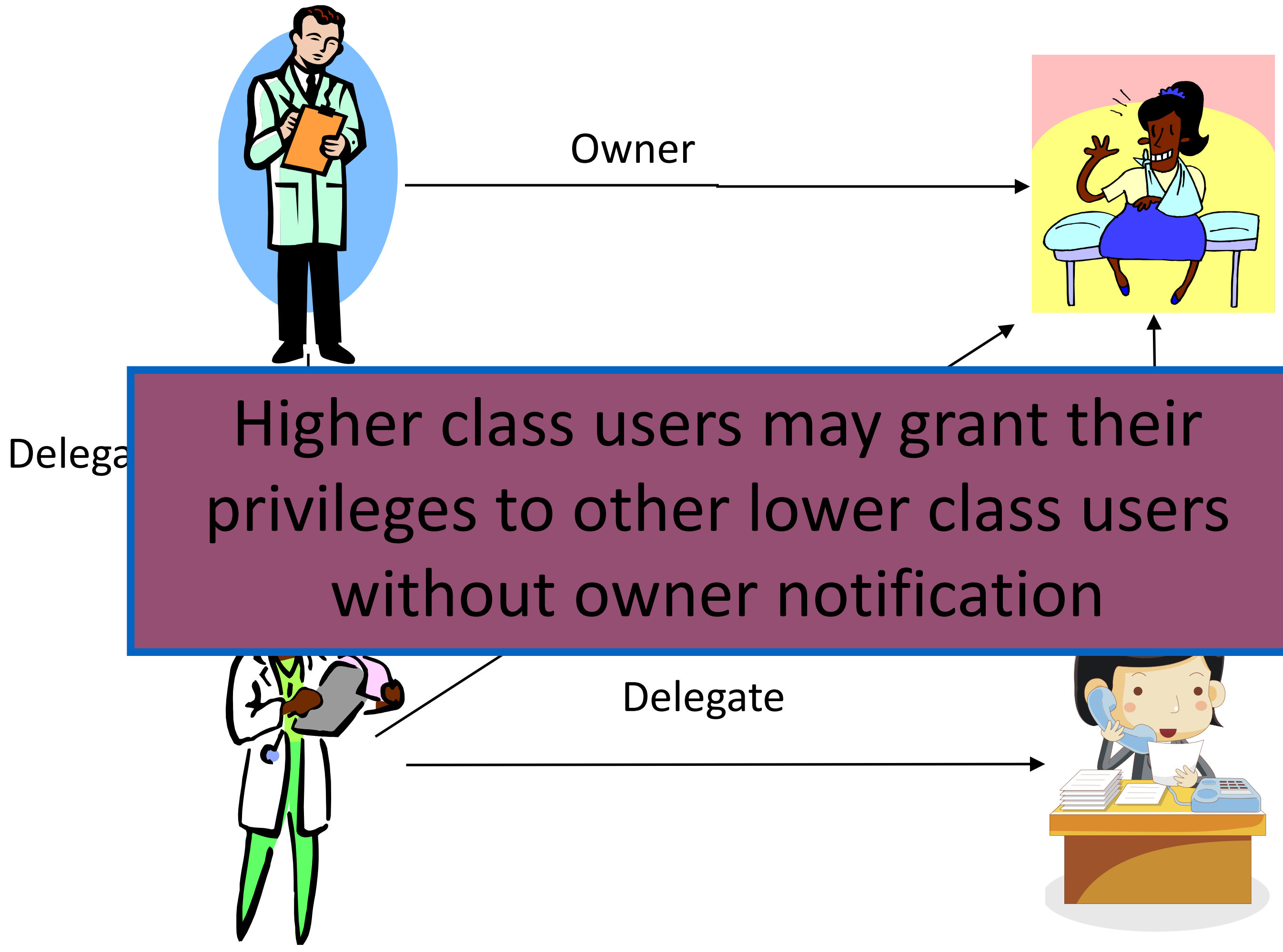
MAC Design Pattern

- The MAC pattern can solve the DAC problems with the DAC pattern in a multi-layered environment (e.g., military and government systems) by assigning security levels to users and objects.
- Use the MAC pattern:
  - Where the environment is multi-layered. For example, in the military domain, users and files are classified into distinct levels of hierarchy (e.g., Unclassified, Public, Secret, Top Secret), and user access to files is restricted based on the classification.
  - When security policies need to be defined centrally. Access control decisions are to be imposed by a mediator (e.g., security administrator), and users should not be able to manipulate them.

Mandatory Access Control

- Mandatory access controls (MAC) restrict the access of subjects to objects based on a system-wide policy
  - Denying users full control over the access to resources that they create. The system security policy (as set by the administrator) entirely determines the access rights granted
- Access Control at Different Abstractions
  - Using principals
    - Determines which principals (user accounts) can access what documents
  - Using subjects
    - Determines which subjects (processes) can access what resources
    - This is where BLP focuses on

Mandatory Access Control: Example



Higher class users may grant their privileges to other lower class users without owner notification

Definition and need for MLS

- Multilevel security involves a database in which the data stored has an associated classification and consequently constraints for their access
- MLS allows users with different classification levels to get different views from the same data
- MLS cannot allow **downward leaking**, meaning that a user with a lower classification views data stored with a higher classification

Multilevel Security (MLS)

  - Security Classification
  - Secrecy-Based Mandatory Policies: Bell-LaPadula Model
  - Integrity-based Mandatory Policies: The Biba Model
  - Limitation of Mandatory Policies
- Hybrid Policies
  - The Chinese Wall Policy

Definition and need for MLS

- Usually multilevel systems are with the federal government
- Some private systems also have multilevel security needs
- MLS relation is split into several single-level relations, A recovery algorithm reconstructs the MLS relation from the decomposed single-level relations
- At times MLS updates cannot be completed because it would result in leakage or destruction of secret information

Definition and need for MLS

- In relational model, relations are tables and relations consist of tuples (rows) and attributes (columns)
- Example:

  Consider the relation

  SOD(Starship, Objective, Destination)

Starship	Objective	Destination
Enterprise	Exploration	Talos
Voyager	Spying	Mars

Definition and need for MLS

- The relation in the example has no classification associated with it in a relational model
- The same example in MLS with classification will be as follows:

Starship		Objective		Destination	
Enterprise	**U**	Exploration	**U**	Talos	**U**
Voyager	**U**	Spying	**S**	Mars	**S**

Definition and need for MLS

- In MLS, access classes can be assigned to:
  - Individual tuple in a relation
  - Individual attribute of a relation
  - Individual data element of tuples in a relation
- Bell – LaPadula Model
- Biba Model

Bell – LaPadula Model

- Bell-LaPadula model was developed in 1973
- This is an extension of the Access Matrix model with classified data
- This model has two components:
  - Classification
  - Set of categories
- Bell-LaPadula model shows how to use Mandatory Access Control to prevent the Trojan Horse

Bell – LaPadula Model

- ***Classes and Clearances*** The mandatory access control model is based on the following components:
  - *Objects :* Database objects such as tables, views, rows, and columns
  - *Subjects:* Users, programs, and modules that need access privileges
  - *Classes :* Security classes for objects
  - *Clearances :* Security clearances for subjects
- Mandatory access controls are based on **security labels associated with each data item and each user**.
  - A label on a data item is called a security **classification**
  - A label on a user is called a security **clearance**.

Bell – LaPadula Model

- Each **database object** is assigned a **security class.**
  - Typical classes are (TS) top secret, (S) secret, (C) confidential, and (U) unclassified.
  - The data sensitivity sequence is as follows: TS > S > C > U.
- Each **subject** is assigned **clearance for a specific security class**.
  - We may represent these by the following notation:
    - Class (O) Security class for an object O
    - Class (S) Security clearance for a subject S
- **Security Level** represents a sensitivity assigned to users (subjects) and objects.
  - A security level consists of a classification and a category. While classifications are hierarchical, categories are non-hierarchical.
- The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of the security classifications TS, S, C, or U. We will refer to the clearance (classification) of a subject S as class(S) and to the classification of an object 0 as class(D).

Bell – LaPadula Model

- Labeling Granularity
  - Security labels can be assigned to data at different levels of granularity in relational databases
  - The finest granularity of labeling is at the level of individual attributes of each tuple (row) or element-level labeling.
  - This offers considerable flexibility.
  - Most of the products emerging in this arena offer labeling at the level of a tuple.

Bell – LaPadula Model

- Classification has four values {U, C, S, TS}
  - U = unclassified
  - C = confidential
  - S = secret
  - TS = top secret
- Classifications are ordered: TS > S > C > U
- Set of categories consists of the data environment and the application area, i.e., Nuclear, Army, Financial, Research

Example: In USA, a "SECRET" clearance involves checking FBI fingerprint files.

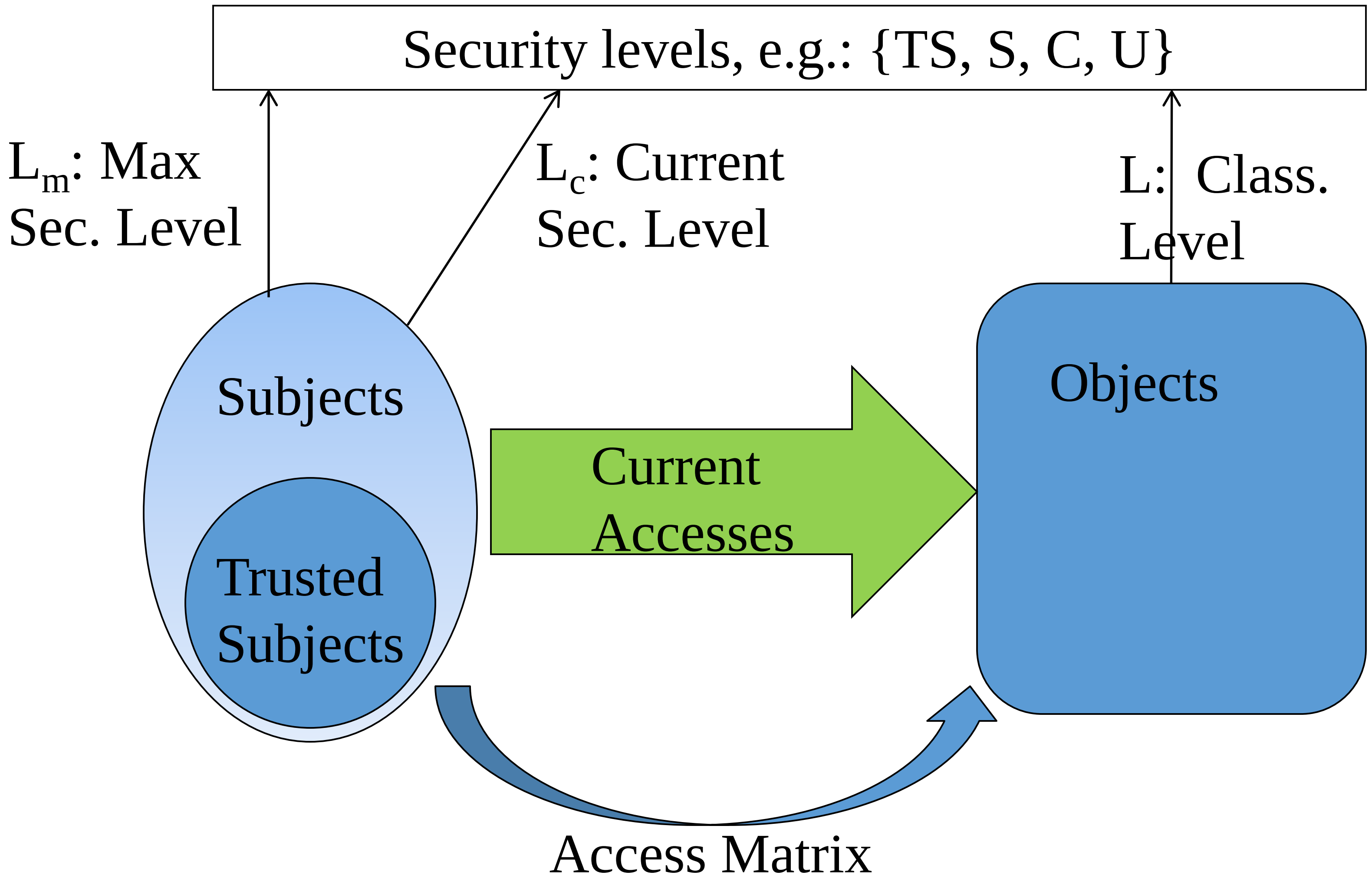
Approach of BLP

- Use state-transition systems to describe computer systems
- Define a system as secure iff. every reachable state satisfies 3 properties
  - simple-security property, *-property, discretionary-security property
- Prove a Basic Security Theorem (BST)
  - so that give the description of a system, one can prove that the system is secure

The BLP Security Model

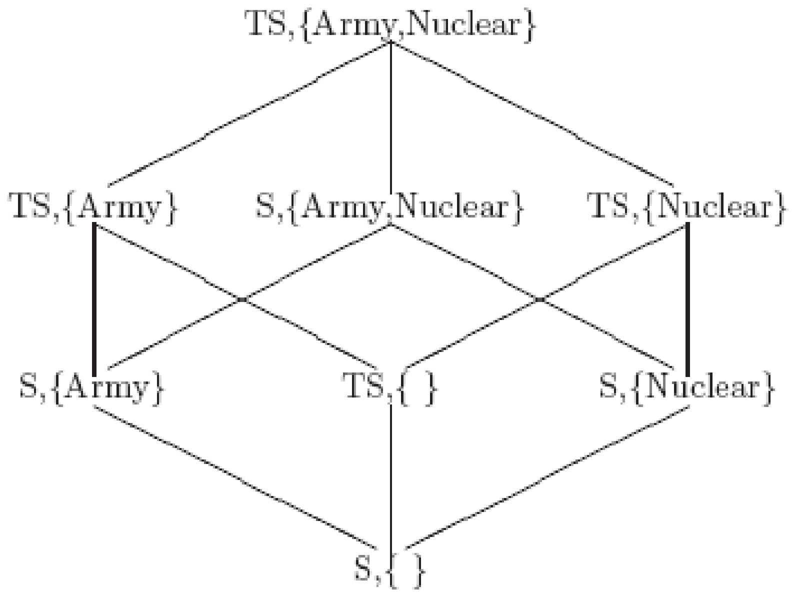
- A DB system is modeled as a state-transition system
  - There is a set of subjects; some are designated as trusted.
  - Each state has objects, an access matrix, and the current access information.
  - There are state transition rules describing how a system can go from one state to another
  - Each subject s has a maximal sec level $L_m(s)$, and a current sec level $L_c(s)$
  - Each object has a classification level

Elements of the BLP Model



Bell – LaPadula Model

- An access class c1 **dominates ≥** an access class c2 iff
  - Security level of c1 is **greater** than or equal to that of c2
  - The categories of c1 **include** those of c2



Bell – LaPadula Model

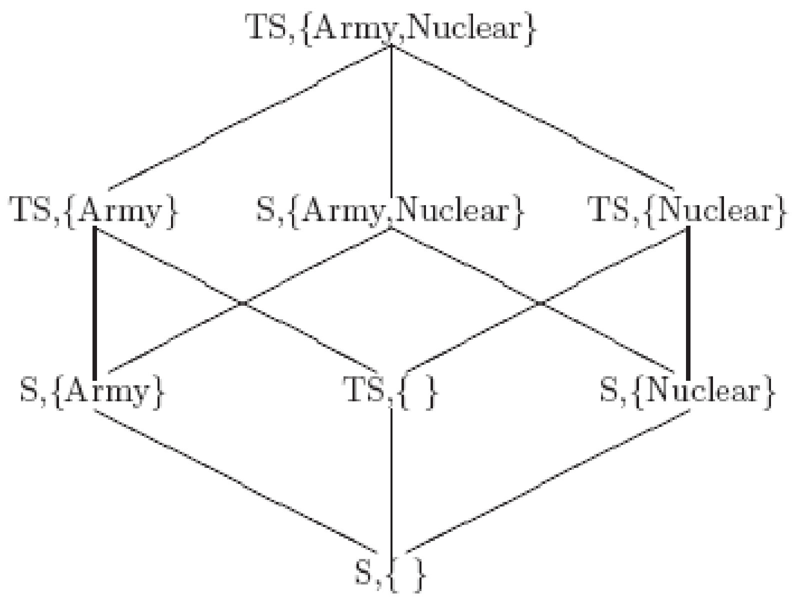
- Bell-LaPadula model is based on a subject-object paradigm
- Subjects are active elements of the system that execute actions
- Objects are passive elements of the system that contain information
- Subjects act on behalf of users who have a security level associated with them (indicating the level of system trust)

Bell – LaPadula Model

- Subjects execute access modes on objects
- Access modes are:
  - Read-only
  - Append (writing without reading)
  - Execute
  - Read-write (writing known data)
- Decentralized administration of privileges on objects

Bell – LaPadula Model

- **Control** direct and indirect **flows of information**
- Prevent leakage to unauthorized subjects
- User can connect to the system with any access class dominated by their clearance



Two Principles

- To protect information confidentiality
  - **No-read-up**, a subject is allowed a read access to an object only if the access class of the subject dominate the access class of the object
  - **No-write-down**, a subject is allowed a write access to an object only if the access class of the subject is dominated by the access class of the object

The BLP Security Policy

- A state is secure if it satisfies
  - Simple Security Condition (no read up):
    - S can read O iff $L_m(S) \geq L(O)$
  - The Star Property (no write down): for any S that is not trusted
    - S can read O iff $L_c(S) \geq L(O)$ (no read up)
    - S can write O iff $L_c(S) \leq L(O)$ (no write down)
  - Discretionary-security property
    - every access is allowed by the access matrix
- A system is secure if and only if every reachable state is secure.

How Mandatory Control Works ?

- **Reference Monitor** checks accessibility based on the following restrictions on all reads and writes.
  1. **Simple security property**
     - **A subject S is allowed read access to an object O only if Class(S) ≥ Class(O).**
     - No subject can read an object whose security classification is higher than the subject's security clearance
     - **No-Read-Up:** A subject S can read an object O if and only if the access class of the subject dominates the access class of the object.
  2. **Star property**
     - **A subject S is allowed write access to an object O only if Class(S) ≤ Class(O).**
     - **No-Write-Down**. A subject S can write an object O if and only if the access class of the object dominates the access class of the subject
     - This property prohibits a subject from writing an object at a lower security classification than the subject's security clearance. Otherwise, information may flow from a higher class to a lower class. Consider a user with S clearance.
     - Without the enforcement of the star property, this user can copy an object in S class and rewrite it as a new object with U classification so that everyone will be able to see the object.
  - Access is allowed when both the constraints are satisfied. Access is checked only if the user is in the same category as that of the object. With the categories matched, the accessibility of the user for the object is determined by the dominance relations of classifications in the above constraints.

Mandatory Access Control: Example

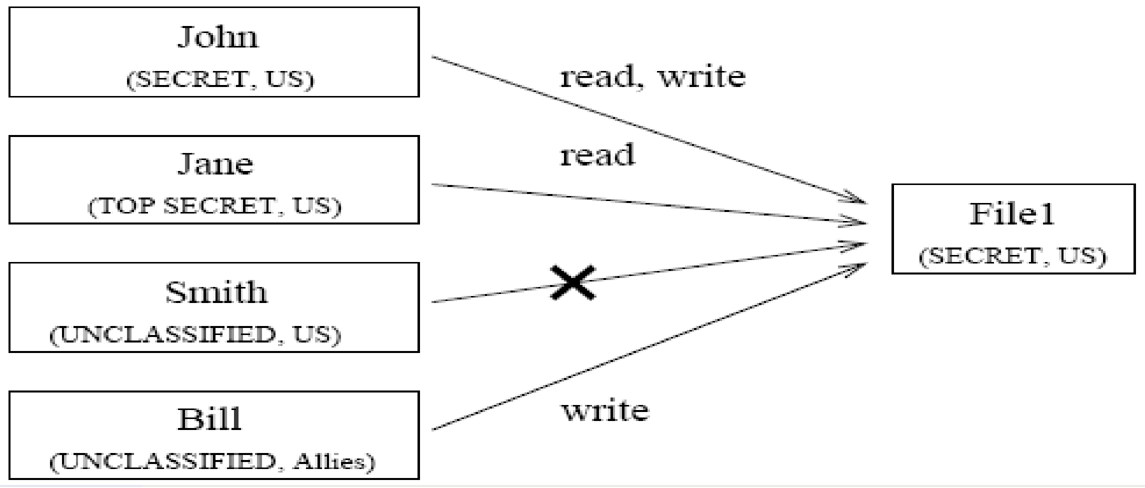
EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Smith U	40000 C	Excellent C	C
Brown C	80000 S	Good C	S

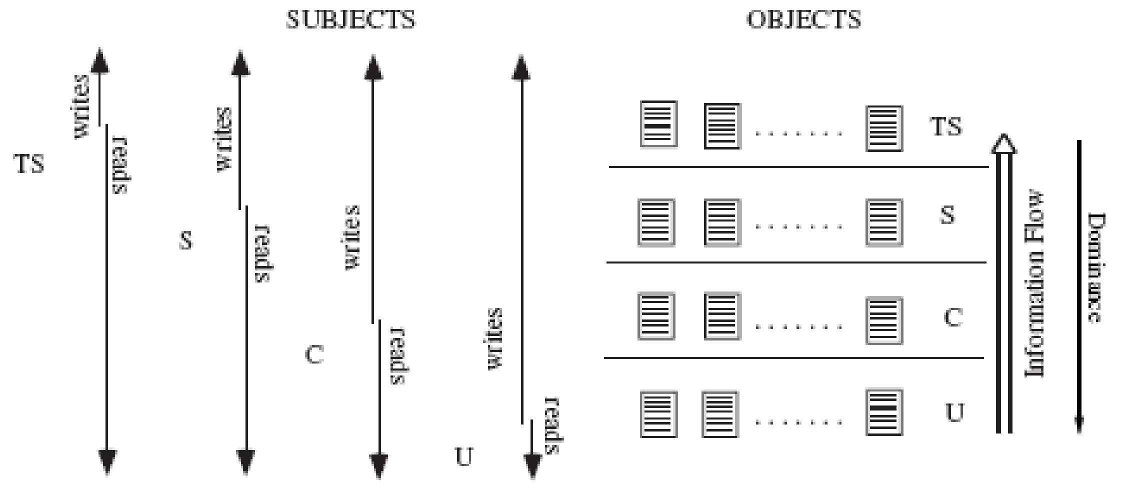
Classification	Category
UNCLASSIFIED	U.S.
CONFIDENTIAL	U.S.
SECRET	U.S.
TOP SECRET	U.S.
UNCLASSIFIED	Allies
CONFIDENTIAL	Allies
SECRET	Allies
TOP SECRET	Allies

User	Classification	Category
John	SECRET	US
Jane	TOP SECRET	US
Smith	UNCLASSIFIED	Allies
Bill	UNCLASSIFIED	US

File	Classification	Category
File1	SECRET	US



No-read-up & No-write-down



- Can TS subject write to S object?
- Can S subject write to U object?
- How to apply to the Trojan Horse case?

Solution to Trojan Horse

- Possible classification reflecting the access restrictions:
  - Secret for *Vicky* and “Market”
  - Unclassified to *John* and “Stolen”
- If Vicky connect to system as **secret**, <u>write is blocked</u>
- If Vicky connects to system as **unclassified**, <u>read is blocked</u>
- Is Vicky allowed to write to the unclassified object? How?

Key Points

- Confidentiality models restrict flow of information
- Bell-LaPadula (BLP) models multilevel security

  Cornerstone of much work in computer security

  - Simple security property says no read up and
  - Star property says no write down
  - Both ensure information can only flow up

Applying BLP: An Example

- Alice has (Secret, {NUC, EUR}) clearance
- David has (Secret, {EUR}) clearance
    - David can talk to Alice ("write up" or "read down")
    - Alice cannot talk to David ("read up" or "write down")
- Alice is a user, and she can login with a different ID (as a different principle) with reduced clearance
    - Alias1 (Secret, {NUC, EUR})
    - Alias2 (Secret, {EUR})

Example

- Get back to the case of Shady trying to access data from the EMPLOYEE table by tricking Miller. The mandatory access control method would spoil Shady’s plan as follows:
    - Classify EMPLOYEE table as S.
    - Give Miller clearance for S.
    - Give Shady lower clearance for C.
- Shady can therefore create objects of C or lower classification. MYTABLE will be in class C or lower. Miller’s program will not be allowed to copy into MYTABLE because Class (MYTABLE) < Class (Miller), violation of star property.

BLP: Problem

- If I can write up, then how about writing files with blanks?
  - Blind writing up may cause integrity problems, but not a confidentiality breach

Bell – LaPadula Model

- Two main properties of this model for a secure system are:
  - Simple security property
  - Star property
- **Simple security** means: a subject at a given security level may not read an object at a higher security level (**no read-up**).

Bell – LaPadula Model

- Star property means: a subject at a given security level must not write to any object at a lower security level (**no write-down**).
- This model guarantees secrecy by preventing unauthorized release of information
- This model does not protect from unauthorized modification of information

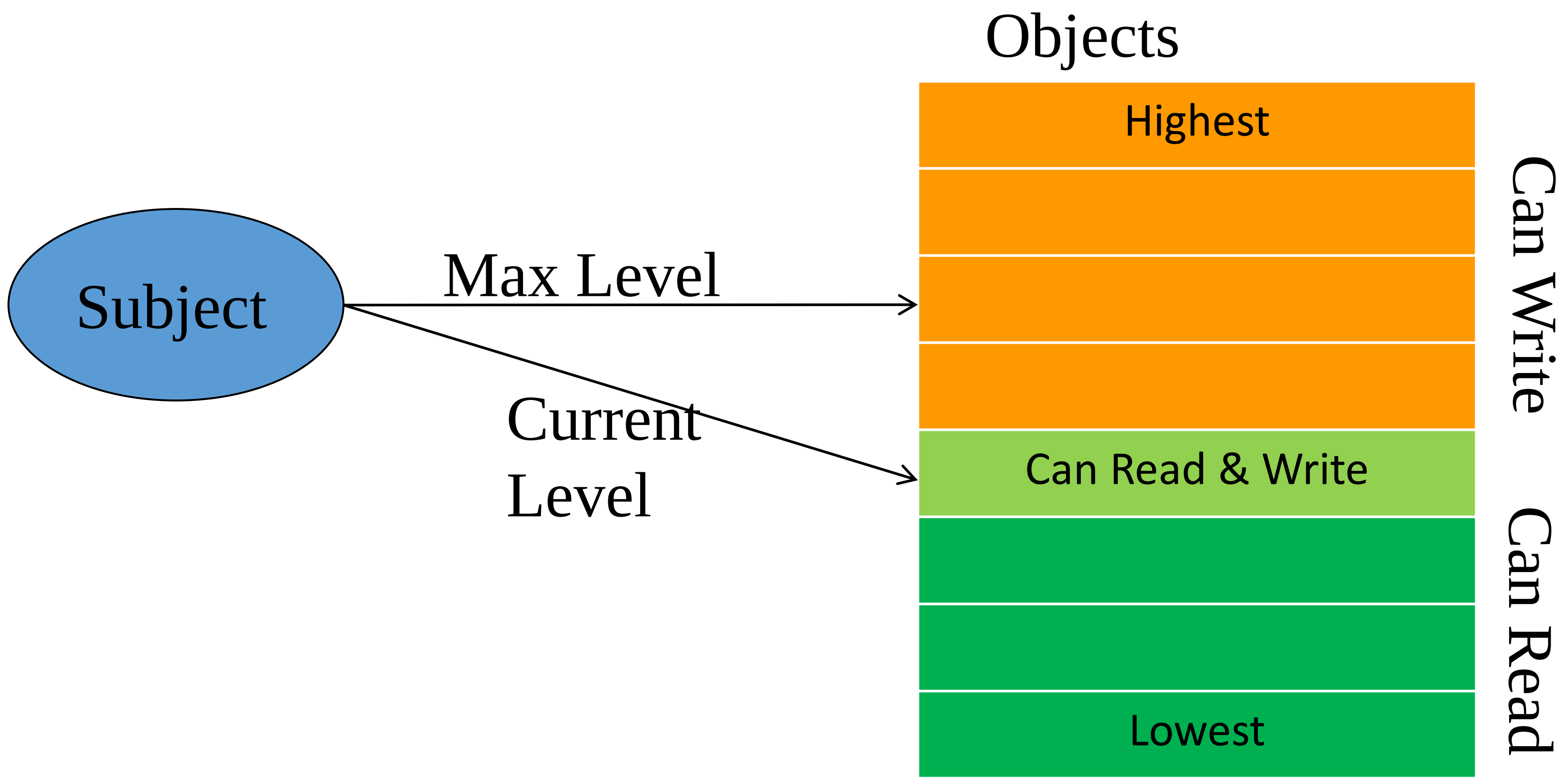
Key Points

- Confidentiality models restrict flow of information
- Bell-LaPadula (BLP) models multilevel security

  Cornerstone of much work in computer security

  - Simple security property says no read up and
  - Star property says no write down
  - Both ensure information can only flow up

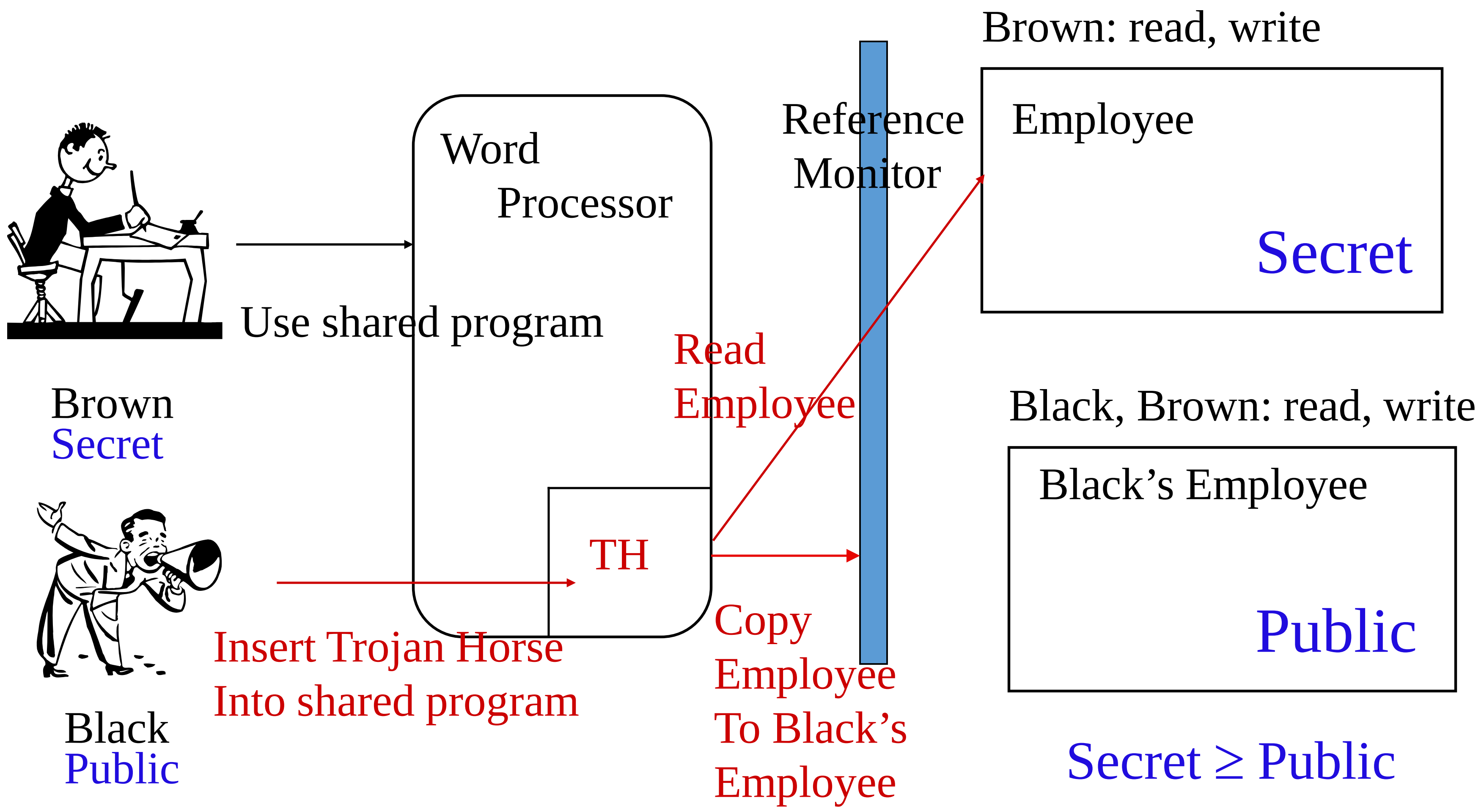
Implication of the BLP Policy



STAR-PROPERTY

- Applies to subjects (principals) not to 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
- Star-property prevents overt leakage of information and does not address the covert channel problem

Trojan Horse and BLP



BLP and Trojan Horses

- Return to the Trojan Horse problem:
    - Alice and Bob are secret level users, Eve is an unclassified user
    - Alice and Bob can have both secret and unclassified subjects (programs)
    - Eve can only have unclassified subjects
    - Alice creates secret file X
    - Simple security prevents Eve from reading X directly
    - Bob can either have a secret (S-Troy) or an unclassified (U-Troy) Trojan-Horse carrying program
    - S-Troy: Bob running S-Troy will create Y, which will be a secret file. Eve's unclassified subjects will not be able to read Y.
    - U-Troy: Bob running U-Troy won't be able to read X, and so won't be able to copy it into Y.
- Thus BLP prevents flow between security classes
- One problem remains: Covert Channels… but that's for another lecture…

Is BLP Notion of Security Good?

- The objective of BLP security is to ensure
  - a subject cleared at a low level should never read information classified high
- The ss-property and the *-property are sufficient to stop such information flow at any given state.
- What about information flow across states?

BLP Security Is Not Sufficient!

- 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, yet illegal information exists
- Solution: tranquility principle: subject cannot change current levels, or cannot drop to below the highest level read so far

Main Contributions of BLP

- The overall methodology to show that a system is secure
    - adopted in many later works
- The state-transition model
    - which includes an access matrix, subject security levels, object levels, etc.
- The introduction of *-property
    - ss-property is not enough to stop illegal information flow

Other Limitations with BLP

- It is a significant model and it has been used in both OS and DBMS
- Deal only with confidentiality, does not deal with integrity at all
  - Confidentiality is often not as important as integrity in most situations
  - Addressed by integrity models (such as Biba, Clark-Wilson, which we will cover later)
  - Containing covert channels
- Does not deal with information flow through covert channels

The Biba Model

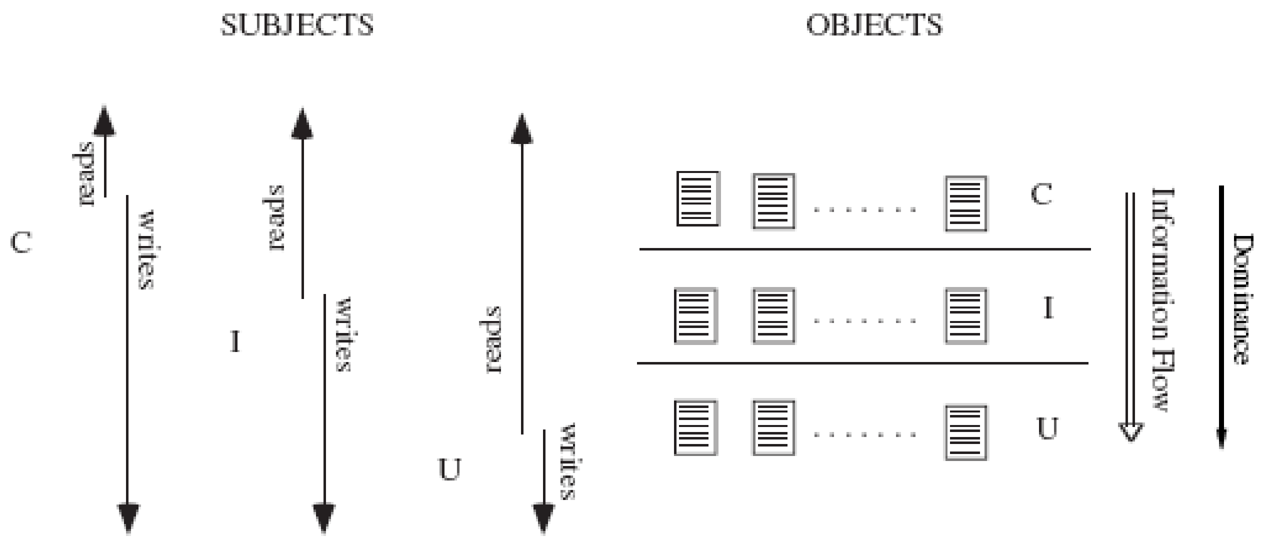
- A model due to Ken Biba which is often referred to as <u>“Bell-LaPadula upside down.”</u>
- It deals with integrity alone and ignores confidentiality entirely.
- Each subject and object in the system is assigned an integrity classification
    - Crucial
    - Important
    - Unknown

Integrity Level

- Integrity level of a user reflects user’s trustworthiness for inserting, modifying, or deleting information
- Integrity level of an object reflects both the degree of trust that can be placed on the info stored in the object, and the potential damage could result from unauthorized modification of info

Two principles

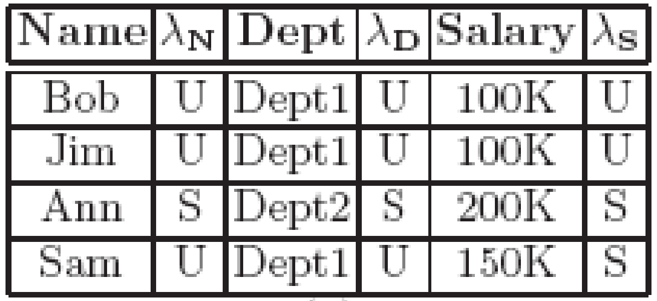
- No-read-down: A subject is allowed a <u>read</u> access to an object only if the access class of the object dominates the access class of the subject
- No-write-up: A subject is allowed a <u>write</u> access to an object only if the access class of the subject is dominated by the access class of the object



Q: How to control both the secrecy and integrity?

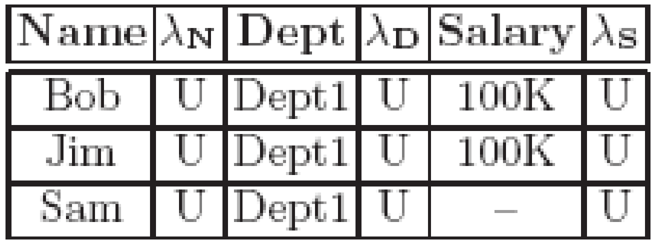
Applying Mandatory Policies to Databases

- Commercial DBMSs Oracle, Sybase, and TruData have MLS versions of their DBMS
- Because of Bell-LaPadula restrictions, subjects having different clearances see different versions of a multilevel relation



Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S

(a)



Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Sam	U	Dept1	U	–	U

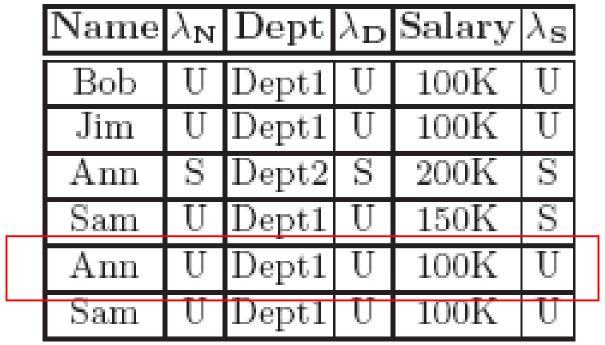
(b)

Visible to a user with secret level.

Visible to a user with unclassified level.

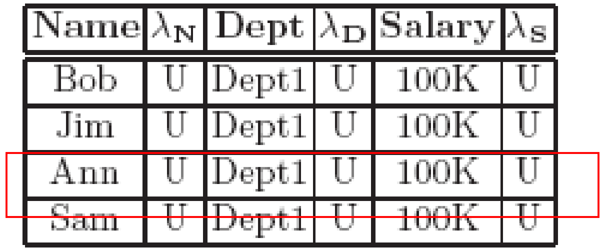
Polyinstantiation

- Request by low level subject
  - An unclassified subject request insert of <Ann, Dept1, 100K>
- If this update is rejected, then the user would be able to infer something about Ann
- MLS would allow the secret channel to permit data update and protect data integrity



Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S
Ann	U	Dept1	U	100K	U
Sam	U	Dept1	U	100K	U

(a)



Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Ann	U	Dept1	U	100K	U
Sam	U	Dept1	U	100K	U

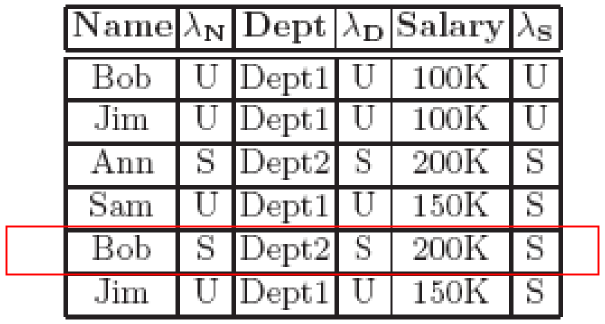
(b)

Visible to a user with secret level.

Visible to a user with unclassified level.

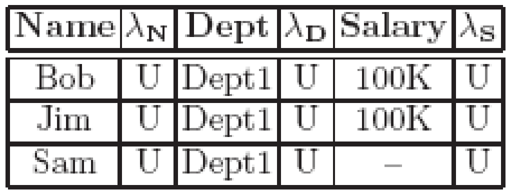
Polyinstantiation

- Request by high level subjects
  - A secret subject request to insert <Bob, Dept2, 200K>
  - Inform the subject of the conflict and refuse the insertion (no)
  - Overwrite the existing tuple (no)



Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S
Bob	S	Dept2	S	200K	S
Jim	U	Dept1	U	150K	S

(a)



Name	λ_N	Dept	λ_D	Salary	λ_S
Bob	U	Dept1	U	100K	U
Jim	U	Dept1	U	100K	U
Sam	U	Dept1	U	–	U

(b)

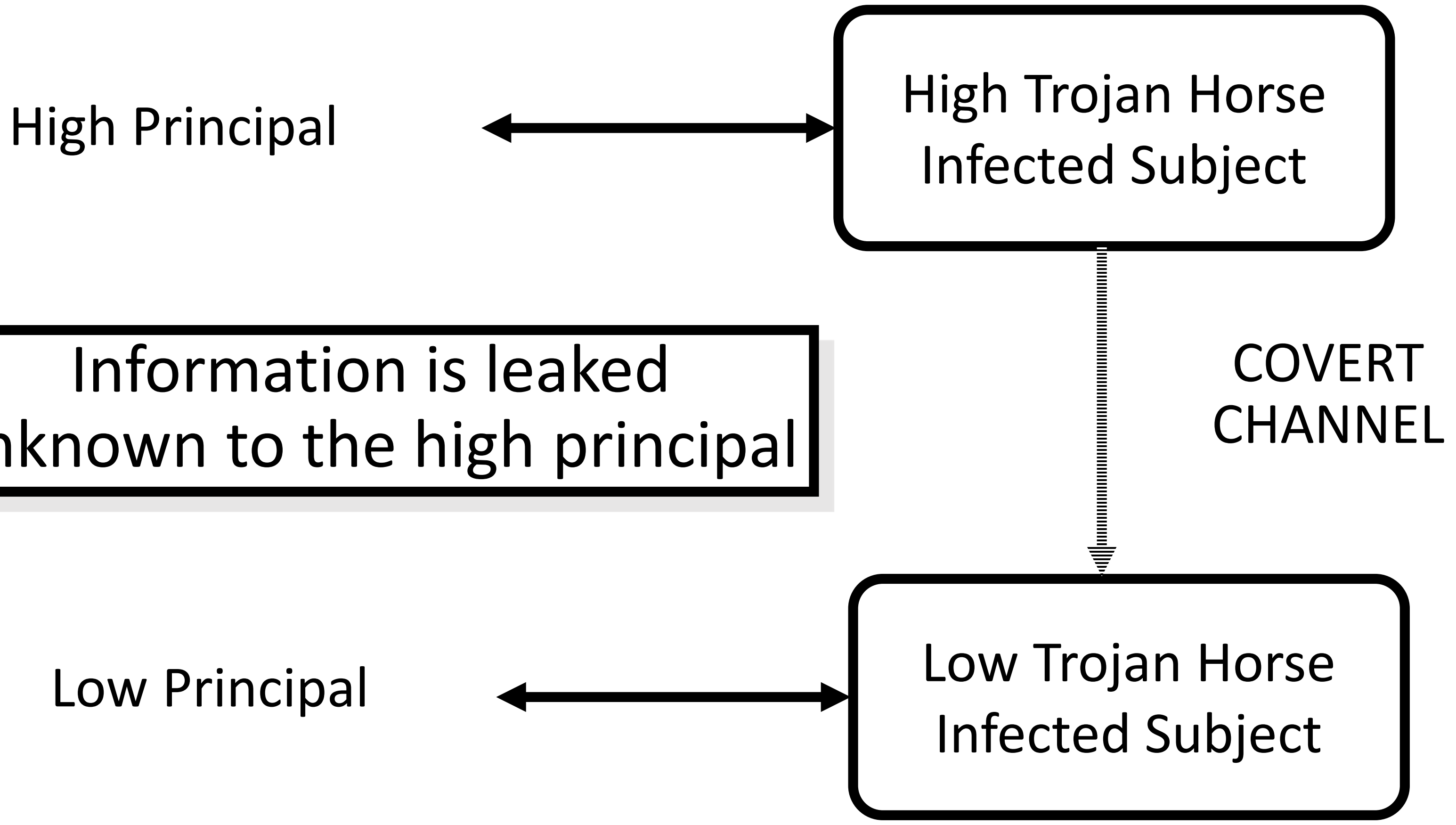
Challenges

- Cover Stories
    - Non-true data to hide the existence of the actual value
    - Not released is a cause of information leakage
- Fine-grained is not easy
    - Aggregation, association
    - Block inference channels

COVERT CHANNELS

- A covert channel is a communication channel based on the use of system resources not normally intended for communication between the subjects (processes) in the system
- 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
- star-property prevents overt leakage of information and does not address the covert channel problem

COVERT CHANNELS


High Principal
High Trojan Horse Infected Subject
COVERT CHANNEL
Low Principal
Low Trojan Horse Infected Subject

Information is leaked unknown to the high principal

RESOURCE EXHAUSTION CHANNEL

Given 5MB pool of dynamically allocated memory

HIGH PROCESS

- bit = 1 $\Rightarrow$ request 5MB of memory
- bit = 0 $\Rightarrow$ request 0MB of memory

LOW PROCESS

- request 5MB of memory
- if allocated then bit = 0 otherwise bit = 1

LOAD SENSING CHANNEL

HIGH PROCESS

bit = 1 $\Rightarrow$ enter computation intensive loop

bit = 0 $\Rightarrow$ go to sleep

LOW PROCESS

perform a task with known computational requirements

if completed quickly then bit = 0 otherwise bit = 1

COPING WITH COVERT CHANNELS

- Identification
  - close the channel or slow it down
  - detect attempts to use the channel
  - tolerate its existence

Covert Channels

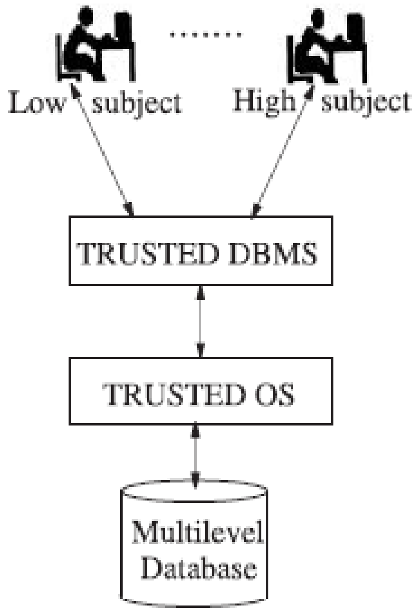
- A <u>covert channel</u> is an information flow that is not controlled by a security mechanism.
- In BLP, you could use the access control mechanism itself to construct a covert channel.
  - A low level subject makes an object *“dummy.obj”* at its own level.
  - Its high level accomplice either <u>upgrades</u> the security level of *dummy.obj* to high or <u>leaves it unchanged</u>.
  - Later, the low level subject tries to read *dummy.obj*. Success or failure of this request disclose the action of the high-level subject.
    - <u>One bit of information has flown from high to low.</u>
    - Failure means *dummy.obj* has be upgraded; success means *dummy.obj* has not been changed

Covert Channels (cont’d)

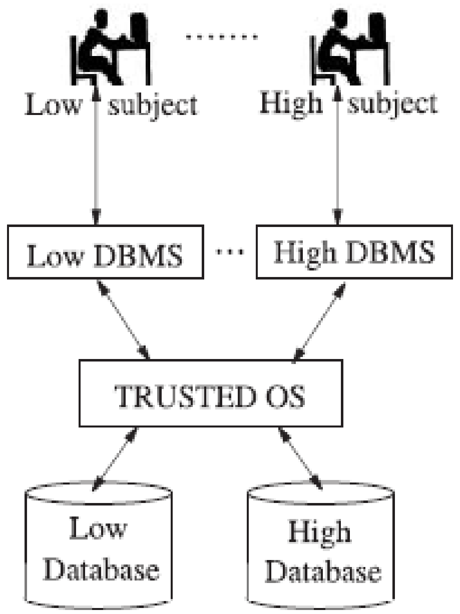
- Other Examples for Covert Channels:
  - Timing Channels
  - Resource State
  - Hidden Information in downgraded documents
- Commonly used techniques for reducing covert channels:
  - Reduce abusable functionality
  - High level processes get lowest resource allocation priority and can be preempted by low level processes.
  - Random delays, clock noise, randomized resource availability.
  - Auditing the use of known channels
  - Polyinstantiation

Multilevel DBMSs Architecture

- Trusted subject. The DBMS itself must be trusted to ensure mandatory policy
- Trusted Computing Base: Data are partitioned in different databases, one for each level



(a) Trusted subject



(b) Trusted computing base

Reference

- Sushil Jajodia and Ravi S. Sandhu, Toward a Multilevel Secure Relational Model, essay 20

Excercise

- Customer order scenario
    - Identify the subject, actions, objects
    - Design the MAC

Chinese Wall Model

Problem:

- Tony advises American Bank about investments
- He is asked to advise Toyland Bank about investments

- Conflict of interest to accept, because his advice for either bank would affect his advice to the other bank

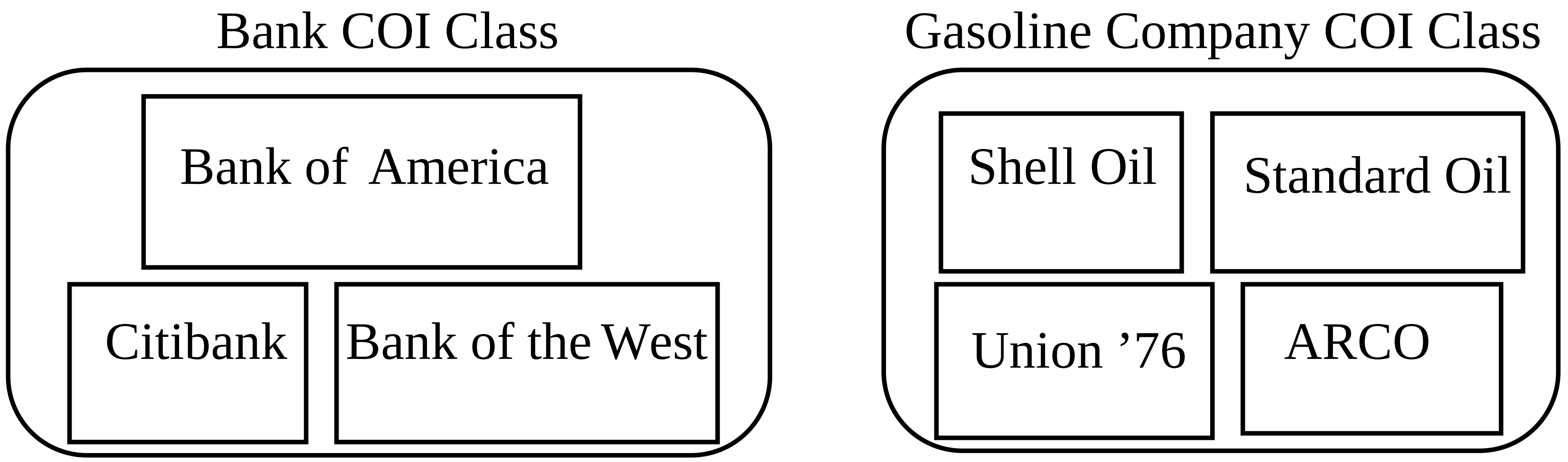
Organization

- Organize entities into “conflict of interest” classes
- Control subject accesses to each class
- Control writing to all classes to ensure information is not passed along in violation of rules
- Allow sanitized data to be viewed by everyone

Definitions

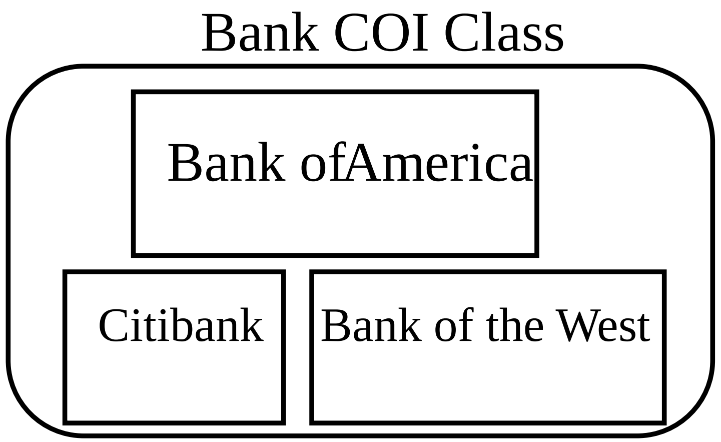
- *Objects*: items of information related to a company
- *Company dataset* (CD): contains objects related to a single company
  - Written *CD*(*O*)
- *Conflict of interest class* (COI): contains datasets of companies in competition
  - Written *COI*(*O*)
  - Assume: each object belongs to exactly one *COI* class

Example


Bank COI Class
Bank of America
Citibank
Bank of the West
Gasoline Company COI Class
Shell Oil
Standard Oil
Union '76
ARCO

Temporal Element

- If Anthony reads any CD in a COI, he can *never* read another CD in that COI
  - Possible that information learned earlier may allow him to make decisions later
  - Let *PR*(*S*) be set of objects that *S* has already read



CW-Simple Security Condition

- *s* can read *o* iff :
  1. s has read something in o’s dataset, and object o is in the same company datasets as the objects already access by s, that is “within the Wall”, or
  2. s has not read any objects in o’s conflict of interest class, what s has read belongs to an entirely different conflict of interest class
- Ignores sanitized data (see below)

Sanitization

- Public information may belong to a CD
    - As is publicly available, no conflicts of interest arise
    - So, should not affect ability of analysts to read
    - Typically, all sensitive data removed from such information before it is released publicly (called *sanitization*)
- Add third condition to CW-Simple Security Condition:
    - 3. *o* is a sanitized object

Writing

- Anthony, Susan work in same trading house
- Anthony can read Bank 1’s CD, Gas’ CD
- Susan can read Bank 2’s CD, Gas’ CD
- If Anthony could write to Gas’ CD, Susan can read it
  - Hence, indirectly, she can read information from Bank 1’s CD, a clear conflict of interest

CW-*-Property

- Write access is only permitted if
  - Access is permitted by the CW-simple security rule, and
  - For all unsanitized objects o’, if s can read   o’, then CD(o’) = CD(o)
- Says that s can write to an object if all the (unsanitized) objects he/she can read are in the same dataset

Exercise : Oracle Label security (Tutorial)

- Test user **hr** with password **hr** is the owner of table **locations** after installation.
  - **connect hr/hr**
  - **select * from locations**

What you need?

- One user who owns a database LOCATIONS, and grants privileges to created users -- hr
- One user to create policy – LBACSYS
- One security policy – ACCESS_LOCATIONS
- One column appended to table LOCATIONS and hold security labels -- OLS_COLUMN
- One user creates security levels and labels -- sec_admin
- One user creates Users, Roles and binds with security labels -- hr_sec

Major steps

- create users (sec_admin, hr_sec, SKING, KPARTNER, and LDORAN)
- create a policy
  - create a policy ‘ACCESS_LOCATIONS’ by lbacsys
  - lbacsys grants some executive rights (ACCESS_LOCATIONS_DBA) to sec_admin (SA_COMPONENT) and hr_sec (SA_USER_ADMIN), so they can change the security policy.
  - sec_admin create security level and labels: ‘SENS’, ‘CONF’, ‘PUB’
- setting user authorization
  - HR_sec binds the labels to the users, defining their clearance.
  - Give owner HR the FULL access to the table
- Applying a policy to a table, only HR can read the data, no label set yet.
- Adding labels to the data by HR.
- revoking Access from Admin Users (sec_admin, hr_sec), revoke ACCESS_LOCATIONS_DBA
- Testing the Policy implementation by connecting to database from different user accounts.

Users

User	**Password and role**
sysdba	Create, alter user, grant CREATE SESSION privilege
system	password: oracle
	create users: *sec_admin* and *hr_sec* with password *welcome1*
	grant *connect* to *emp_role*
LBACSYS	password: LBACSYS
	default Oracle DBA for Oracle Label Security (OLS)
	After creating a policy
	He has a role <policy_name>_DBA with ‘ADMIN’ option,
	which allow him to grant *execute on SA_COMPONENTS* and *execute on SA_USER_ADMIN* to other users such as *sec_admin* and *HR_sec*.
sec_admin	password: welcome1
	Create levels and labels

Users

hr_sec	password: welcome1
	Maintain user-related part of the OLS, create database roles, users and grants clearance to them. **Bind labels to the users.**
	create role emp_role
	create user SKING identified by welcome1
	grant role emp_role to SKING
	create user KPARTNER identified by welcome1
	grant role emp_role to KPARTNER
	grant user LDORAN identified by welcome1
	grant role emp_role to KPARTNER
hr	Password: hr
	Owner of table locations, who determines the sensitivity of his data and who will get access to which level of sensitivity.
	Grant *select* to emp_role
	Adding labels to the data

Users

SKING	Password: welcome1
	Labeled as ‘SENS’ by hr, owner of table locations
KPARTNER	Password: welcome1
	Labeled as ‘CONF’ by hr, owner of table locations
LDORAN	Password: welcome1
	Labeled as ‘PUB’ by hr, owner of table locations

Tables, Policy, and Colum of Labels

Table name	Owner
LOCATIONS	A table owned by hr
Policy name	**Creator and objective**
ACCESS_LOCATIO NS	Creator is LBACSYS
	Control access to hr.LOCATIONS table
Colum name	**objective**
OLS_COLUMN	Name of the hidden column, will be appended to the hr.LOCATIONS table
	Holds the data label.

Classified Users and Labeled Users

Labeled table:

City	country_id,	label_to_char (OLS_COLUMN)
Venice	IT	PUB
Hiroshima	JP	PUB
Southlake	US	PUB
South San Francisco	US	PUB
South Brunswick	US	PUB
Seattle	US	PUB
Toronto	CA	PUB
Whitehorse	CA	PUB
Bombay	IN	PUB
Sydney	AU	PUB
London	UK	PUB
Stratford	UK	PUB
Sao Paulo	BR	PUB
Geneva	CH	PUB
Bern	CH	PUB
Utrecht	NL	PUB
Mexico city	MX	PUB
Roma	IT	CONF
Oxford	UK	CONF
Munich	DE	CONF
Tokyo	JP	SENS
Beijing	CN	SENS
Singapore	SG	SENS

Classified Users:

SKING (SENS)
KPARTNER (CONF)
LDORAN (PUB)

Inference and Aggregation

Introduction

- Inferring prohibited information from results of queries is known as the inference problem.
- Inference problem uses an inference channel.
- Inference channel in a database provides a facility to infer data with a higher classification from a data with a lower classification.
- Goal of inference problem is to detect and remove inference channels

Inference Problems

- Inference involves indirect access
- Example: User has privilege to view data X but not data Y. Both these data are in table T. If the query

  SELECT X FROM T WHERE Y = value

  produces any result, then user has inferred something about Y
- If user attempts an insert and it is denied, then it leads to inference
- Inferences of this type are easy to eliminate.
  - The system can either modify the user query such that the query involves only the authorized data or simply abort the query.
- If the user is cleared to see all data involved in the query, then the result can be returned to him, but it must be labeled at the least upper bound of all labels involved in the query.

Inference Problems

- Inference involves indirect access
- Example
  - Unclassified relation EP(EMPLOYEE-NAME, PROJECT-NAME)
  - Secret relation PT(PROJECT-NAME, PROJECT-TYPE)
  - The existence of the relation scheme PT is unclassified.

Uncleared user made the SQL query

```
SELECT        EP.EMPLOYEE-NAME
FROM          EP, PT
WHERE         EP.PROJECT-NAME = PT.PROJECT-NAME
```

Although the output of this query is unclassified, it reveals Secret information in PT relation. We have an inference channel.

Inference Problems

- Inference could also result from correlated data, meaning that visible data is related to invisible data
- Knowing the values t and k can to guess an unknown value $z = t * k$ is inference
- Estimating value of z requires reducing the degree of uncertainty for z. Reducing the uncertainty degree using results of authorized queries is also inference

Inference Problems

- Inference could also result from missing data
- A channel of missing data is an inference channel
- Missing data usually comes from having null values for fields such as salary when an employee has a name and department identified

Specific Inference Problems--Inference from data combined with metadata

- Key integrity requires that every tuple in a relation must have a unique key.
- Functional and multivalued dependencies are constraints over the attributes of a relation.
- Value constraints is a constraint on data values that can involve one or more items of data
- Classification constraints is a rule describing the criteria according to which data is classified.

Specific Inference Problems--Inference from data combined with metadata

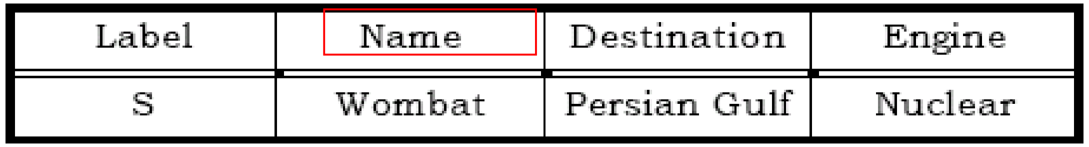
- **Key integrity**: A user at a low security class can use the low security class data and the constraint (if it is made available to the user) to **infer** information about high security class data also affected by the constraint.
- This constraint does **not** cause a problem when data is classified at the relation or column level, since in that case all keys in a relation are at the same security class.

Specific Inference Problems--Inference from data combined with metadata

- If a low security class user who wants to enter a tuple in a relation in which data is classified at either the tuple or the element level.
- If a tuple with the same key at a higher security class already exists, then to maintain key integrity, the DBMS must <u>either delete the existing tuple</u> or <u>inform the user that a tuple with that key already exists</u>.
- Problems
  - In the first case, the actions of a low user can cause data inserted by a high user to be deleted, which is unacceptable.
  - In the second case, we have an inference channel: The existence of high data is allowed to affect the existence of low data.

Specific Inference Problems--Inference from data combined with metadata

- To illustrate, consider the following instance (where “Name” is the key for the relation):
- Suppose an unclassified user wants to insert the tuple (Wombat, Norfolk, Nuclear).
- We have an integrity problem if we delete the secret tuple (since it is possible that the entry “Norfolk” in the unclassified tuple is merely a cover story for the real, classified entry “Persian Gulf”).
- If we reject the insertion, then the low user can derive an inference.
- this problem can be eliminated using polyinstantiation, in which case both tuples are allowed to exist.



Label	Name	Destination	Engine
S	Wombat	Persian Gulf	Nuclear

Specific Inference Problems--Inference from data combined with metadata

- In **Functional and multivalued dependencies**, inference channels can arise if certain functional dependencies are known to low users.
- *Example 2*. Assume that a company database consists of the relation scheme EMP-SAL(NAME, RANK, SALARY). The attributes NAME and RANK are nonsensitive, while the attribute SALARY is sensitive.
- Suppose every employee is aware of the constraint that all employees having identical ranks have the same salaries. Given this scenario, an employee who is not permitted to have access to sensitive data can easily determine employee salaries, which are sensitive.

Specific Inference Problems--Inference from data combined with metadata

- Reason: functional dependency RANK → SALARY is not properly reflected in the classification levels of attributes RANK and SALARY.
- If the rank of an employee is known to a user, then the employee’s salary is also known to that user.
- Solution: Raise the classification of the attribute RANK from nonsensitive to sensitive.
- If attributes are assigned security labels in a manner consistent with the functional dependencies, then these inference threats can be eliminated.

Specific Inference Problems--Inference from data combined with metadata

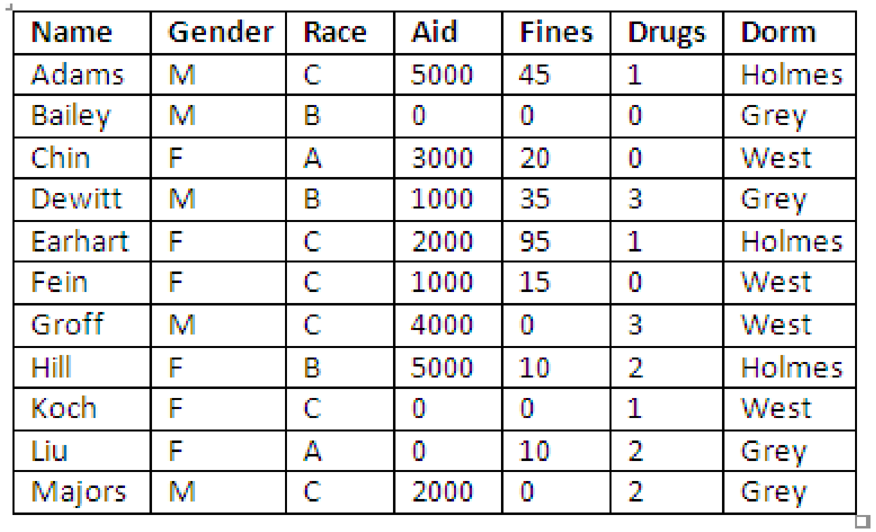
- More solutions: Su and Ozsoyoglu give several algorithms for raising the classification labels of attributes based on functional and multivalued dependencies among them.
- One of their algorithms takes as **input**
  - a list of attributes,
  - the proposed classification labels of the attributes, and
  - a set of functional dependencies that cause inferences.
- The algorithm produces as **output** another list of attributes together with their classification labels such that the list is free of inference channels arising from functional dependencies.

Specific Inference Problems--Inference from data combined with metadata

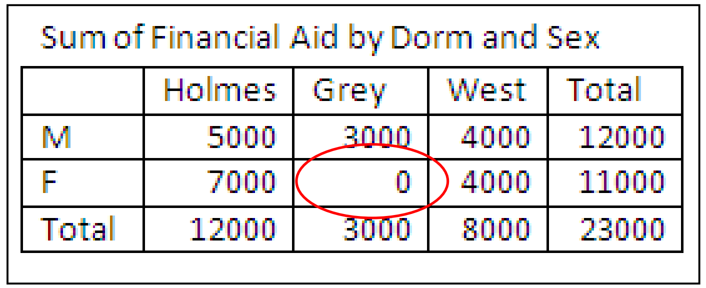
- In **value constraints**, a constraint defined over data at different security levels, availability may lead to inference channels.
- Example 3. Suppose that an attribute A is Unclassified while attribute B is Secret.
- Suppose the database enforces the constraint $A + B \leq 20$, which is made available to Unclassified users.
- The value of B does not affect the value of A directly, but it does determine the set of possible values A can take.

Inference Problem

- The **inference problem is a way to infer or derive** sensitive data from non-sensitive data.
- **Sum: An attack by sum tries to infer a value from** reported sum. Often helps us determine a negative result.
  - This report reveals that no female living in Grey is receiving financial aid.



Name	Gender	Race	Aid	Fines	Drugs	Dorm
Adams	M	C	5000	45	1	Holmes
Bailey	M	B	0	0	0	Grey
Chin	F	A	3000	20	0	West
Dewitt	M	B	1000	35	3	Grey
Earhart	F	C	2000	95	1	Holmes
Fein	F	C	1000	15	0	West
Groff	M	C	4000	0	3	West
Hill	F	B	5000	10	2	Holmes
Koch	F	C	0	0	1	West
Liu	F	A	0	10	2	Grey
Majors	M	C	2000	0	2	Grey



Sum of Financial Aid by Dorm and Sex

	Holmes	Grey	West	Total
M	5000	3000	4000	12000
F	7000	0	4000	11000
Total	12000	3000	8000	23000

Inference Problem

- **Count: count + sum → average; average + count → sum**
  - This report reveals that two males in Holmes and West are receiving financial aid in the amount of $5000 and $4000, respectively.
    - Holmes → Adams
    - West → Groff

Name	Gender	Race	Aid	Fines	Drugs	Dorm
Adams	M	C	5000	45	1	Holmes
Bailey	M	B	0	0	0	Grey
Chin	F	A	3000	20	0	West
Dewitt	M	B	1000	35	3	Grey
Earhart	F	C	2000	95	1	Holmes
Fein	F	C	1000	15	0	West
Groff	M	C	4000	0	3	West
Hill	F	B	5000	10	2	Holmes
Koch	F	C	0	0	1	West
Liu	F	A	0	10	2	Grey
Majors	M	C	2000	0	2	Grey

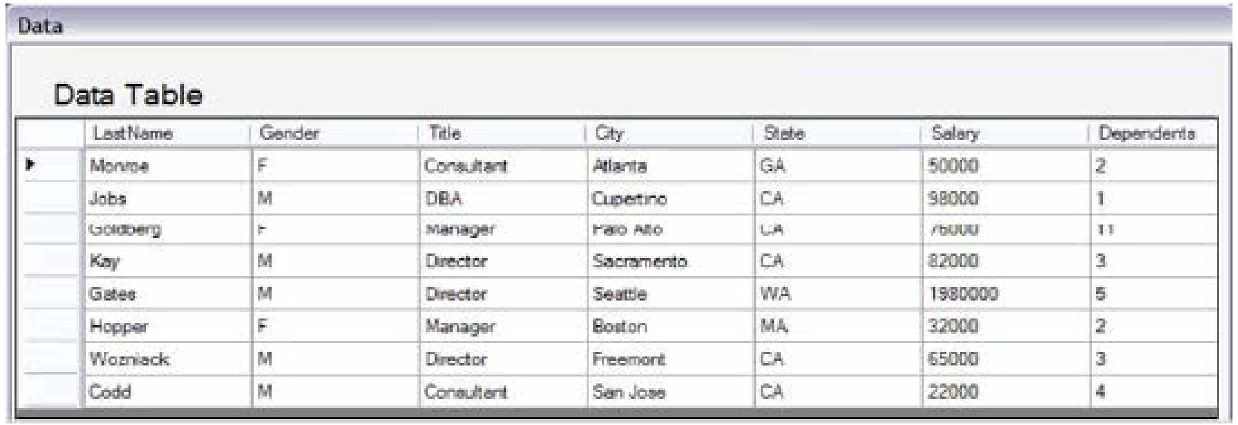
Count of students by Dorm and Sex

	Holmes	Grey	West	Total
M	1	3	1	5
F	2	1	3	6
Total	3	4	4	11

Sum of Financial Aid by Dorm and Sex

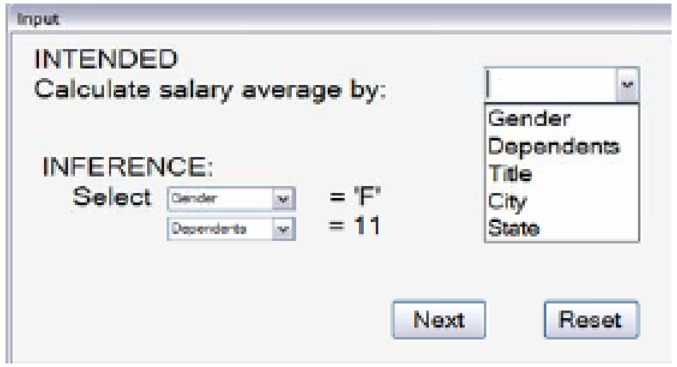
	Holmes	Grey	West	Total
M	5000	3000	4000	12000
F	7000	0	4000	11000
Total	12000	3000	8000	23000

Inference Problem


Data

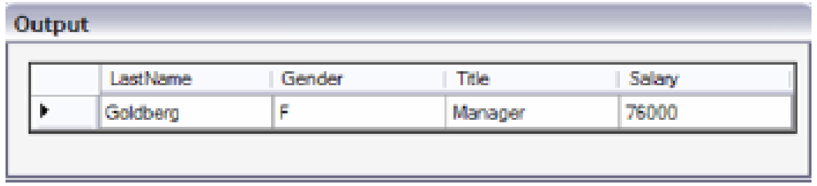
Data Table

	LastName	Gender	Title	City	State	Salary	Dependents
▸	Monroe	F	Consultant	Atlanta	GA	50000	2
	Jobs	M	DBA	Cupertino	CA	98000	1
	Goldberg	F	Manager	Palo Alto	CA	76000	11
	Kay	M	Director	Sacramento	CA	82000	3
	Gates	M	Director	Seattle	WA	1980000	5
	Hopper	F	Manager	Boston	MA	32000	2
	Wozniack	M	Director	Freemont	CA	65000	3
	Codd	M	Consultant	San Jose	CA	22000	4

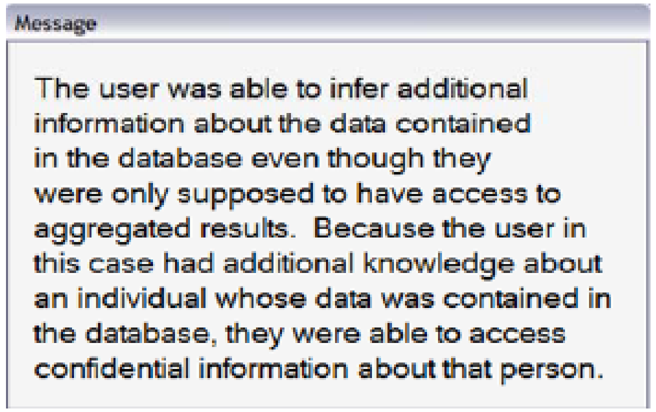

Input

INTENDED
Calculate salary average by:

INFERENCE:
Select Gender = 'F'
Dependents = 11


Output

	LastName	Gender	Title	Salary
▸	Goldberg	F	Manager	76000


Message

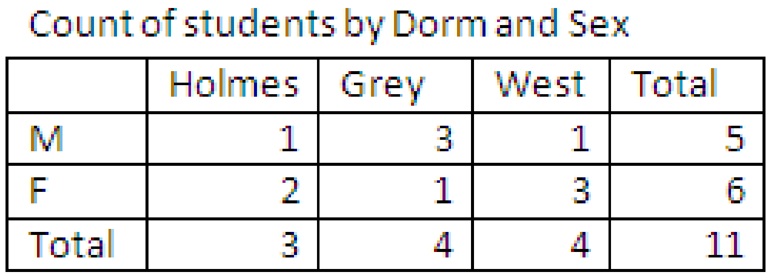
The user was able to infer additional information about the data contained in the database even though they were only supposed to have access to aggregated results. Because the user in this case had additional knowledge about an individual whose data was contained in the database, they were able to access confidential information about that person.

Controls for Statistical Inference Attacks

- Controls are applied to queries
  - Difficult to determine if query discloses sensitive data
- Controls are applied to individual items within the database (security vs. precision)
  - **Suppression: sensitive data values are not** provided; query is rejected without response
    - Many results suppressed; precision high
  - **Concealing: answer provided is close to by not** exactly the actual value
    - More results provided; precision low

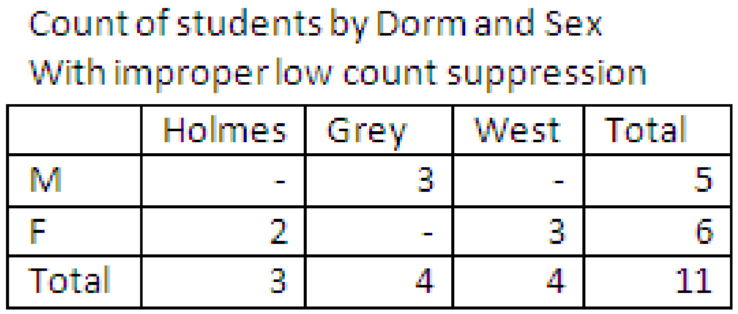
Limited Response Suppression

- The n-item k-percent rule eliminates certain low-frequency elements from being displayed
- When one cell is suppressed in a table with totals for rows and columns, must suppress at least one additional cell on the row and one on the column to provide some confusion.



Count of students by Dorm and Sex

	Holmes	Grey	West	Total
M	1	3	1	5
F	2	1	3	6
Total	3	4	4	11



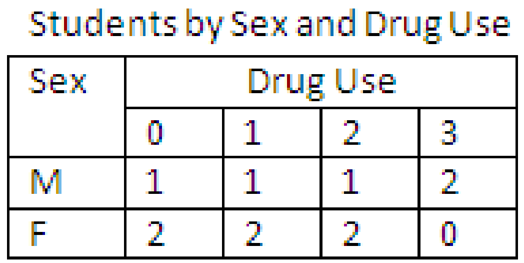
Count of students by Dorm and Sex
With improper low count suppression

	Holmes	Grey	West	Total
M	-	3	-	5
F	2	-	3	6
Total	3	4	4	11

… Can only provide totals

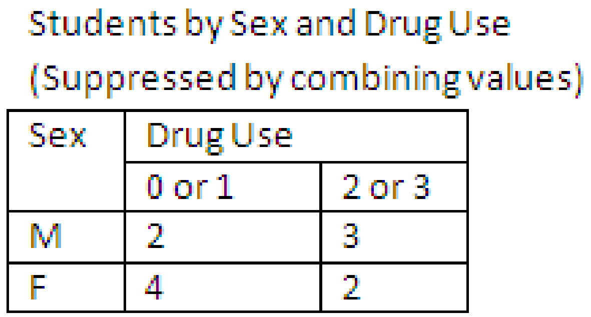
Other suppression and concealing

- <u>Combine rows or columns to protect sensitive values</u>



Students by Sex and Drug Use

Sex	Drug Use			
	0	1	2	3
M	1	1	1	2
F	2	2	2	0



Students by Sex and Drug Use
(Suppressed by combining values)

Sex	Drug Use	
	0 or 1	2 or 3
M	2	3
F	4	2

- Take a random sample (sample must be large enough to be valid)
  - Same sample set would be repeated for equivalent queries
- Query analysis
  - Query and its implications are analyzed
  - Can be difficult
  - Maintain query history for each user
- … no perfect solution to inference problem
- … recognizing the problem leads to being defensive

Aggregation Problems

- An aggregation problem exists when the aggregate of two or more data items is classified at a level higher than the least upper bound of the classification of the individual items.
- The most commonly cited example is the SGA (Secretive Government Agency) phone book [SCHA83]: The entire phone book is classified but individual numbers are not.

Aggregation Problems

- Aggregation policies
    - Use the aggregation policy as a guide for downgrading. That is, begin by classifying all members of the aggregate at the level of the aggregate, and then downgrade as many as is consistent with the aggregation policy.
    - Use the aggregation policy as a guide for relaxing security requirements. In one example, the members of the aggregate were made available only to individuals who were cleared to the level of the aggregate, but they were allowed to follow less strict policies for handling individual aggregate members. Thus, an Unclassified member of a Confidential aggregate could be stored on an Unclassified PC.
    - Release individual members of an aggregate to individuals cleared at the lower level, but do not release more than a certain fixed number to any one individual. This was the policy followed in the SGA phone book example. Any individual could be given as many as *N* phone numbers, where *N* was some fixed number, but no more.

Aggregation Problems

- Another possible way of handling an aggregation problem can be used when inferences may be formed by watching the ways in which data changes over time. In this case, one could prevent inferences by limiting not the amount of data an individual sees, but the amount of time during which he has access to the data.

Aggregation Problems

- Mechanisms to implement aggregation policies. These usually involve keeping some sort of history of each user’s access, and granting or denying access to a member of the aggregate based on that history:
  - In the SeaView system [LUNT89a], data is stored high and selectively downgraded according to the requester’s past access history.

Aggregation Problems

- In the LDV system [STAC90], data is stored low and access to it is selectively restricted based on its access by low users.
- In the Brewer-Nash model [BREW89] and its generalization by Meadows [MEAD90a], data is stored at different levels and access is granted to levels based on the past access history of the user or of a set of users. In Meadows' model, histories may also be kept of devices and other environments to which the data may be exported.

Aggregation Problems

- A problem closely related to aggregation and often confused with it is one commonly known as the "data association problem."
- This occurs when two data items may not be sensitive, but their association is.
- Example: names and salaries are considered nonsensitive, but the association between a name and a salary is.
- Solution: treat it as an aggregation problem; that is, to give a user access to names or to salaries, but not to both.
- what is really sensitive in this case is not the combination of a list of names and a list of salaries, but the association between individual names and individual salaries.