CSCI262
System Security (S3a)

Access control I:

Concepts and BLP

Overview

- What is access control?
- Access control matrices.
- Access control lists.
- Capabilities.
- Access policies.
- Multilevel policies.
- The Bell-LaPadula model.

What is access control?

- Access:
  - – Able to do something or get somewhere.
  - – Carry out an action.
- Control:
  - – To restrict or allow.

- Access Control → Being able to restrict or allow particular actions.
  - – Often access control is applied **where there is authentication**.

Here is a question…

Is there a difference between these two statements?

1. “This file can only be accessed by an authenticated user.”
2. “This file can only be accessed by an authorised user.”

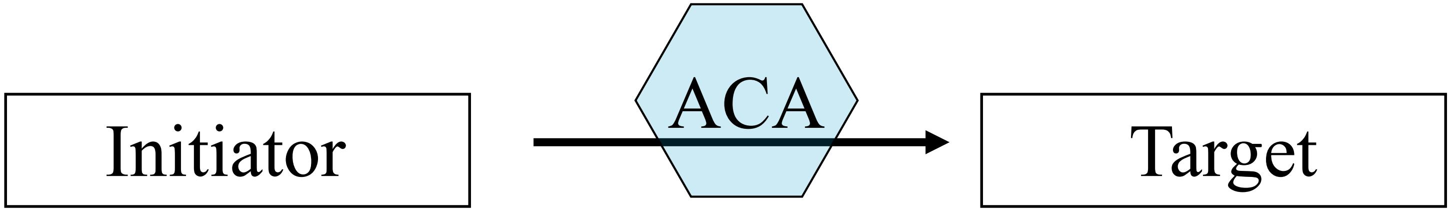
1. An authenticated user is a non-anonymous user who identity has been confirmed. Any such user could then access this system.
2. An authorized user is an user that has the required permissions to access this particular file, or is in a group/role that has the required access permissions.
   - While we know who you are, we won’t let you access this file because you are not authorised to do so.
   - An entity might be considered an authorized user once we have authenticated them and checked the access control permissions.

Another question...

- Do we always need both authentication and access control?
- Are there any circumstances where we might apply one but not the other?

Access Control Authority (or service)

- Control access to information and resources.
- Model : Initiator and Target Entities.



MORDAC, THE PREVENTER OF INFORMATION SERVICES
DO YOU KNOW HOW THE SYSTEM DISTINGUISHES BETWEEN AUTHORIZED AND ILLEGAL ACCESS?
ZAP!
IT DOESN'T.
HA HA!!!
www.dilbert.com scottadams@aol.com
4-4-08 ©2008 Scott Adams, Inc./Dist. by UFS, Inc.

Dilbert: 04-April-2008

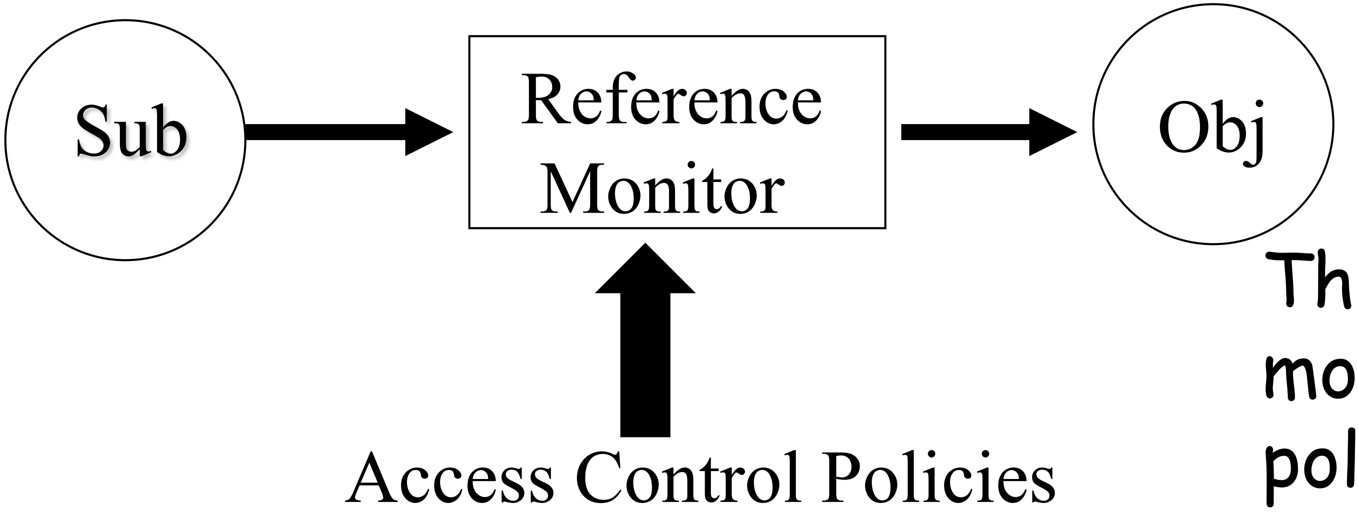
- Access Control Information:
  - Individual/Group identities of initiators and targets.
  - Security labels of initiators and targets.
    - More on this later.
  - Roles.
  - Actions or operations that can be performed.
  - Contextual information : location, time periods.
- Access Control Policy:
  - Rules that define the conditions under which initiators can access targets.
    - Who? can Access What?, When? and How?
- Access Control Authority:
  - Enforcement of access control policy.

Some formalism: Subjects and Objects

- Subjects: Entities that use:
  - Includes users and processes.
- Objects : Entities that are used:
  - Includes files, directories, memory.

- Subjects access objects:

  → Traditional Access Control Model.



Sub

Reference Monitor

Obj

Access Control Policies

The reference monitor enforces policies for each interaction

Access Control Matrices

- The concept was developed independently by researchers in operating systems and databases.
- An access control matrix (ACM) model is defined in terms of state and state transitions.
- The state of a system is defined by a triplet (S, O, A):
  - S: A set of subjects.
  - O: A set of objects.
  - A: An access control matrix, A[S, O] with entries a(s,o).
    - a(s,o) lists the access rights of s on o.
    - Access rights specify the kind of access allowed for a subject on each object.

- What are the relevant subjects and objects in the context of operating systems and databases?
- In operating systems:
  - Subjects : users, processes, domains (a protection environment in which a process executes).
  - Objects : files, memory segments, processes.
- In databases:
  - Subjects: users.
  - Objects: relations, records, fields.

Example: Access Control Matrix

Objects Subjects	**File$_1$**	**File$_2$**	**Process$_1$**	**…**	**…**
Process$_1$	Read	Read Write			
Alice		Read	Execute		
Bob	Write		Execute		
Carol					
…					

Translate please

- Process$_1$ can read File$_1$:
  - If A(Process$_1$, File$_1$) $\supseteq$ Read
- Alice can execute Process$_1$:
  - If A(Alice, Process$_1$) $\supseteq$ Execute
- …

Building states ... Dynamics

- Our system isn’t going to be static, in the sense of there always be the same actions currently place, or even the same possible objects or subjects.
- The Harrison-Ruzzo-Ullman Model is the earliest model to really describe changes to the access control matrix, effectively to both...
  - ... the collection of possible states the system can be in, that is *possible*.
  - ... the (security) policies for the system that tell us which subset of states are *allowed*.

- We can start with an initial state effectively a collection of actions taking place at time t=0.
- A series of state transitions will take us from a current state to another state.
- These state transitions are composites of commands that change...
  - ... the collection of possible states the system can be in, that is *possible*.
  - ... the (security) policies for the system that tell us which subset of states are *allowed*.

The primitive commands

- Following Bishop, Following Harrison, Ruzzo and Ullman…
  - – Construct subject s.
  - – Destroy subject s.
  - – Construct object o.
  - – Destroy object o.
  - – Enter a into a[s,o].
  - – Remove a from a[s,o].
- Bishop formalises these with preconditions and postconditions, such as, constructing a subject is possible iff the subject didn’t exist already.

- A command sequence for a process to create a file, which it will subsequently own and have read and write rights on.

```
command create.file(p,f)
    create object f;
    enter own into a[p,f];
    enter read into a[p,f];
    enter write info a[p,f];
end
```

Conditions or context…

- Conditions could be added.
- For example,

```
command grant.read.file(p,f,q)
    if own in a[p,f];
    then
      enter read into a[q,f];
end
```

You wouldn’t need to write these in the exam!

Meta-actions or meta-rights

- The creating of subjects and objects, may have their own access control rules.
- The granting of privileges typically will too, and are pretty much governed by the …
- Principle of Attenuation of Privilege:
  - A subject may not give rights it doesn’t possess to another.

A warning …

- There are many problems that occur because of the dynamics.
  - If an action is currently taking place and there is an attempt to remove permissions, what happens?
    - Consider specifically: What if someone attempts to remove an object that someone is currently reading?
  - What happens if we are asking these questions in the context of a distributed system?

Representations of Access Control

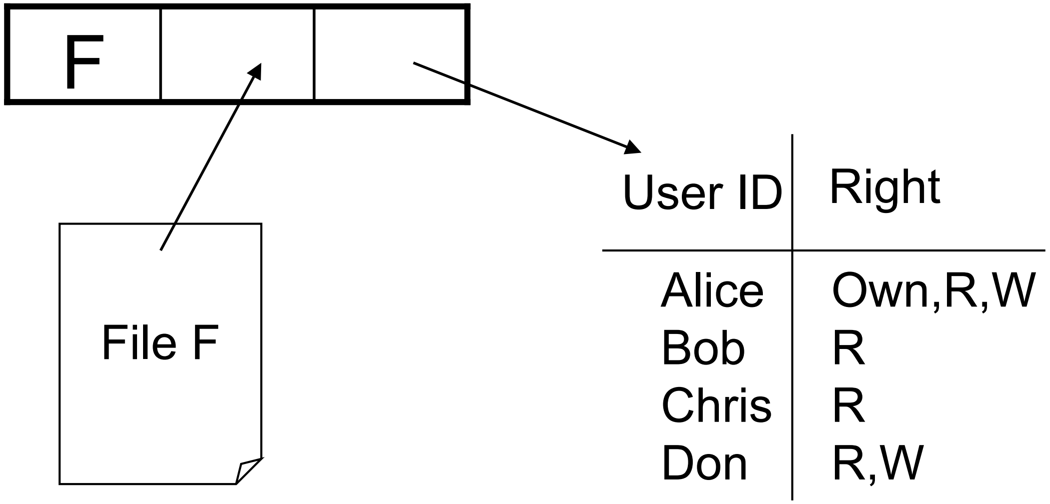
- Access Control Lists (ACL):
  - Contain access from the viewpoint of an object.
  - In other words, a column of the ACM.
  - For example:
    - File F : (A, Write), (B, Read).
- Capabilities:
  - These correspond to the viewpoint of a subject.
  - In other words, a row of the ACM.
  - For example:
    - ((Read, F1), (Write, F2)….)
  - Subject presents a capability to an object.

Access Control Lists

- A list of subjects that are authorised to access an object.
- Identity-based policies (such as individual-based and role-based policies) can be realised in a straightforward way.
- Maintenance of the list and enforcement of the access control are essentially the responsibility of the systems and environment surrounding the object.
- Access control list:

s1	s2	s3	...
r1	r2	r3	...

- What does it mean?
  - Subject s1, s2, s3 … have rights r1, r2, r3 … for this object.
  - The subjects could be individuals or roles (team leaders, managers…)



User ID	Right
Alice	Own,R,W
Bob	R
Chris	R
Don	R,W

Access control lists are used to protect owned objects. The owner can confer/invoke rights by adding, deleting or modifying the entry for a user.

Access Control Lists

- Access control lists are expensive to work with as every access to the object is checked.
  - They are best suited for situations where there are relatively few users (individuals, groups).
  - They do have the advantage of giving the owner a lot of control over modification of other user’s access.

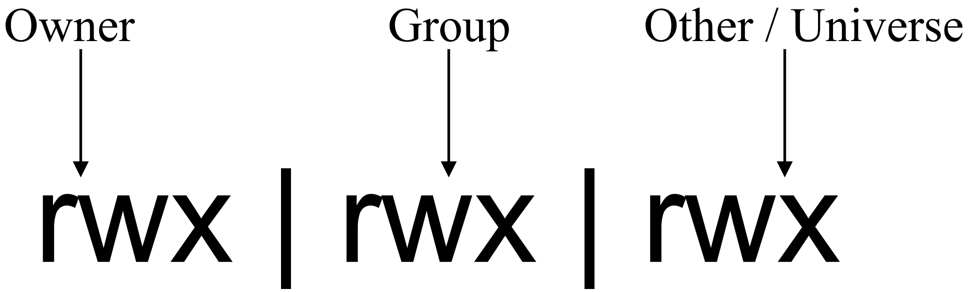
Identity	Type	Perm granted	Object
G. Smith	csci	r, w	report.tex
team-mem	admin	r, w, x	a.exe
alice	maths	r	intro.txt
. . .			

- In **Unix** a file has an access control list with three entries: owner, group, and other users.
- The type of access can be r, w, x (roughly).

```
$ ls -l | more
total 152
drwxr-xr-x   4 wsusilo  cs-uow      512 Aug 14  2007 111
drwxr-x---   5 greg     cs-uow      512 Nov  8  2004 112
drwxr-x---   2 david    csstf       512 Sep  8  2004 114.old
drwxr-x---   7 txia     other       512 Oct 18  2004 121
drwxr-x---   8 dfs      csstf       512 Dec  2  2003 121.2003
drwxr-x---   5 wsusilo  cs-uow      512 Sep  9  2003 121.old
drwxrwx---   4 koren    other       512 Apr  4  2006 124
lrwxrwxrwx   1 root     other        37 Jan  7 15:43 131 -> /web/itacs/documents/subjects/csci131
drwxr-x---   4 jrg      cs-uow      512 Jun  7  2001 1999.235
drwxrwxr-x   2 ian      csci203m   1024 May 30 11:46 203
drwxr-x---  10 greg     uowugc      512 Sep 20  1999 203.old
drwxr-sr-x   7 lukemc   csci204m    512 May 16 17:14 204
drwxr-x---   2 dfs      root        512 Jul 21  2007 204.dfs
drwxr-xr-x   7 lei      cs-uow      512 May 28 13:33 212
…
```

The permission string…

- The permission string - --- --- --- can be (if we ignore the first element) written in terms of permissions for the three classes:



Owner

Group

Other / Universe

rwx | rwx | rwx

- The ‘R’ permission means read.
- The ‘W’ permission means write.
- The ‘X’ permission means execute on a file. In the context of a directory it means the directory can be searched.

- These permissions can also be described as a 3 digit octal (base 8) number. The *read* flag contributes a 4, the *write* flag contributes a 2 and the *execute* flag contributes a 1.

- So the following would yield the octal permission 744.

-rwxr--r--	
4 (r) + 2 (w) + 1 (x) for owner	7
4 (r) for group	4
4 (r) for others/ universe	4

Capabilities

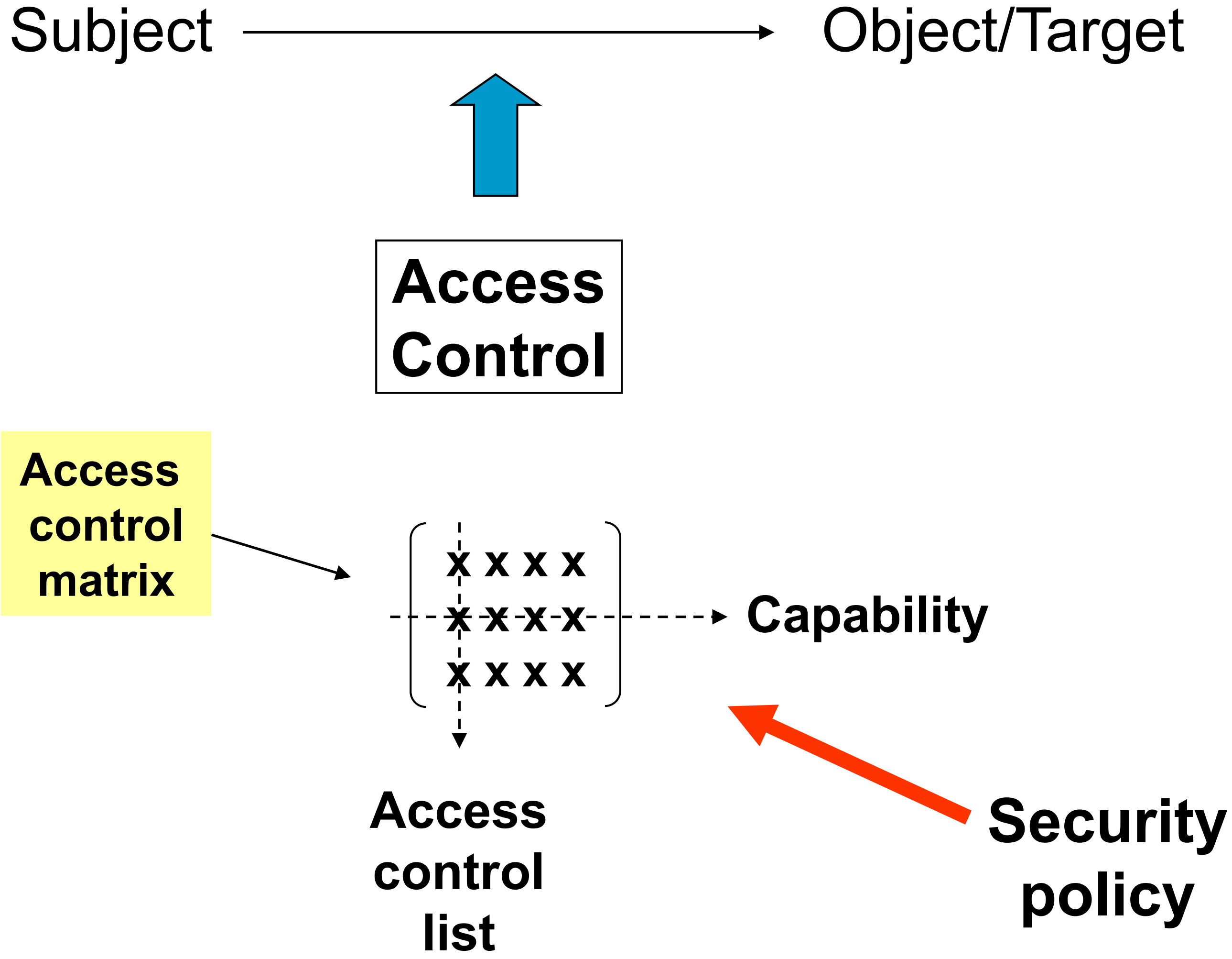
- Describes what a subject is capable of doing.
- Subject : With a list of (Object, Rights)
- Classical Capability System:
  - Capability : A triplet: <Object, Rights, Check>
  - Check = f(Object, Rights).
- We can think of a capability as a ticket that authorises the holder to access an object in a particular way.
  - The Check is there for authentication. It could be something like a message authentication code or digital signature (see CSCI361).
  - Capabilities are difficult to revoke.

- To obtain access an access request and the capability are transmitted to the appropriate server.
- Access Decision:
  - When the access request and capability arrive, the function f is applied to detect tampering.
  - If the capability passes => access is granted!

Sandhu & Samarati's authorization table

- The access control matrix may be sparse.
  - A more concise representation, but less common, is the authorization table.
- One row of the table is used for each allowed access triplet.
- The table can be sorted by Subject or Object to give equivalence to a capability list or an ACL, respectively.

A brief summary



Subject

Object/Target

Access Control

Access control matrix

Capability

Access control list

Security policy

Access Control Policies

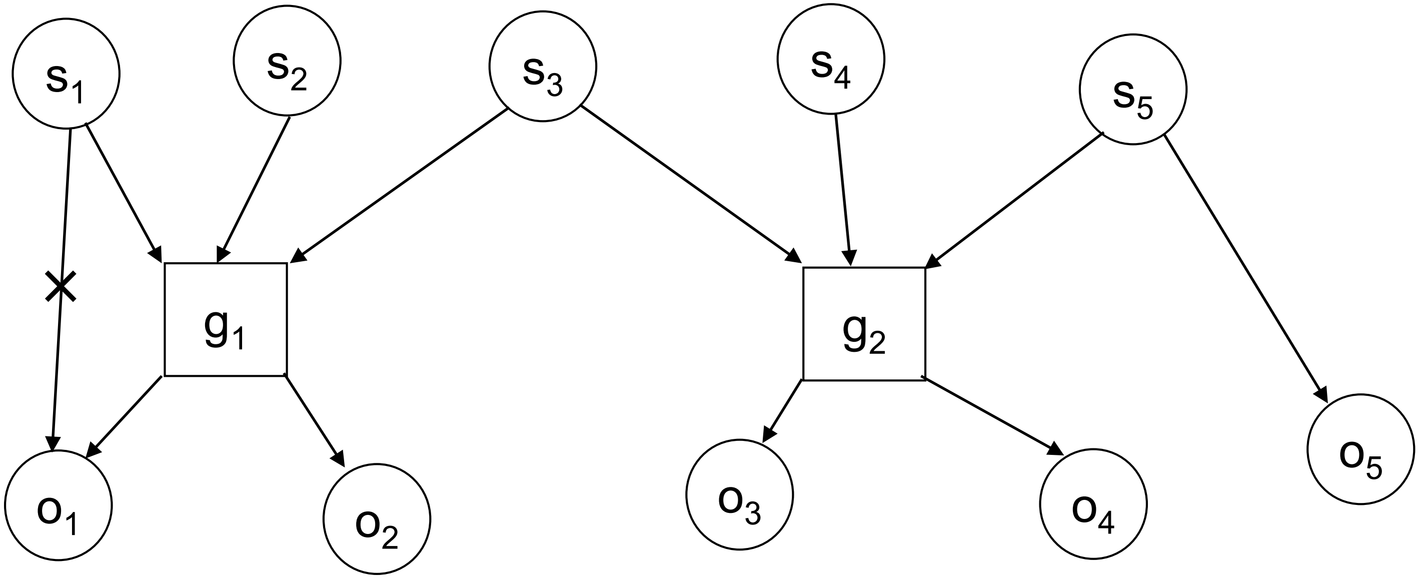
- Traditional Views:
  - Discretionary Access:
    - Users use their own discretion to specify who can access what, usually within some prescribed structure as in Unix.
      - The access control matrices capture a discretionary view.
  - Mandatory Access:
    - Subjects and Objects have fixed security attributes that are used by the System to determine access.
    - Users cannot modify security attributes.
    - System (Security Administrator) → decides.

- Identity based Policies:
  - (subject, action, object, time, location, context).

- Examples:
  - User A can read the balance of account Y.
  - User B can withdraw money from account Y.
  - User C can withdraw money from account Y, Monday to Friday.
  - User D can withdraw money from account Y on a Monday, from Bank Z, if they wear a blue hat.

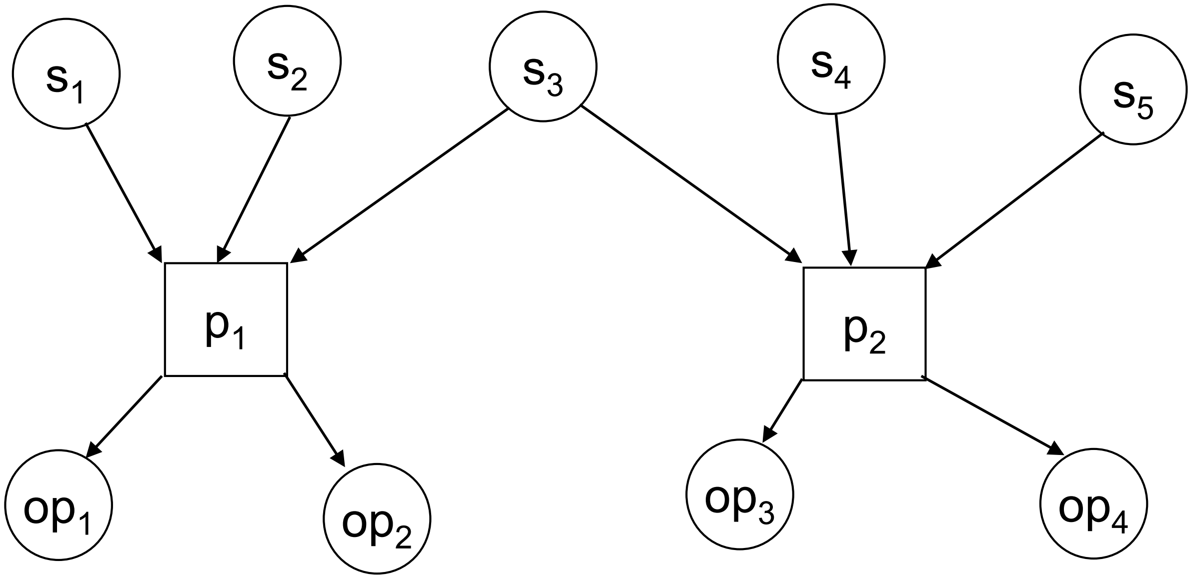
- Intermediate control layers:
  - These are structures in-between the individual subject/object/action level and global permissions.
    - Group, privilege, role, attribute, ring, level based.
  - These simplify control management.
- **Group based access control:**
  - Users are assigned to groups.
    - Depending on the system policy, a user might be allowed to be a member of one group or multiple groups.
  - Groups are given permissions to access objects.
  - Each user has the permissions assigned to the group or groups it is a member of.

- Here goes an example of group based access control.

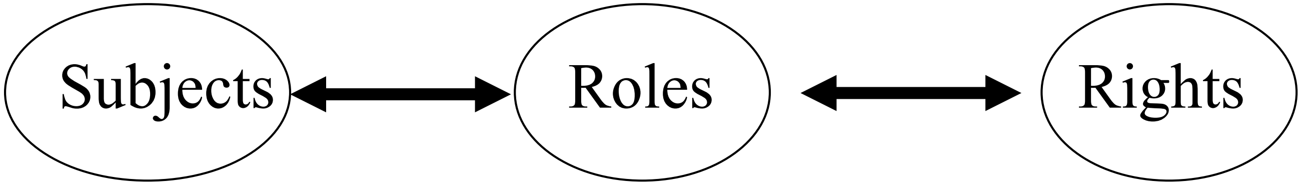


- Notice the negative permission.
- There is a conflict in the policies associated with s_1.
  - There would need to be a reference monitor policy to resolve this conflict.

- **Privileges**:
  - Privileges can be viewed as an intermediate layer between subjects and actions or operations.
  - A subject is assigned privileges that allow the subject to execute certain operations.
  - Typically, privileges are associated with OS functions and relate to activities like system administration, or network access.



- **Role based access control (RBAC):**
  - While privileges are associated an operating system, this idea can be generalized…
  - Roles, gathering together particular actions on objects, can be defined for other applications or contexts.
  - For example, an Accountant can issue cheques.



Role based access: An example

- In a banking environment there are several appropriate roles: Teller, Branch Manager, Customer, System Administrator, Auditor.
- A **Teller** has permission to modify a customer account with a deposit, carry out withdrawal transactions up to a specified limit, and query all account log entries.
- A **Branch Manager** has the same permissions as a teller but can also create and terminate accounts.

- A **Customer** is allowed to query the account log for his/her own account.
- A **System Administrator** can query all system log entries, activate/deactivate the system, but cannot read or modify customer account information.
- An **Auditor** can read any data in the system but modify nothing.

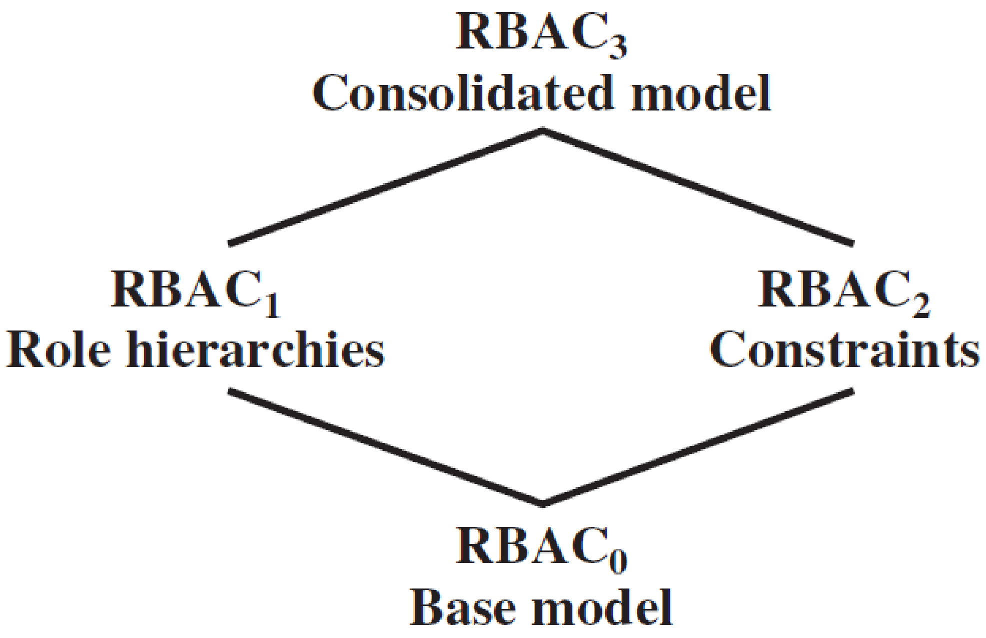
Role based access control

- Assign access rights to roles instead of individual users
- Users are assigned to different roles
  - A single user may be assigned multiple roles
  - Multiple users may be assigned to a single role

RBAC models

- The base model includes the association of the type described earlier.
- Extensions include role hierarchies and constraints.
  - Role hierarchies can capture the idea of the branch manager having all the functionality of the teller, plus something beyond that.
  - The constraints relate to role assignment, and include mutual exclusivity, cardinality and prerequisites.

$RBAC_i$



(a) Relationship among RBAC models

RBAC$_1$

- Role hierarchies
  - Reflecting the hierarchical structure of roles in an organisation
  - Make use of the concept of inheritance

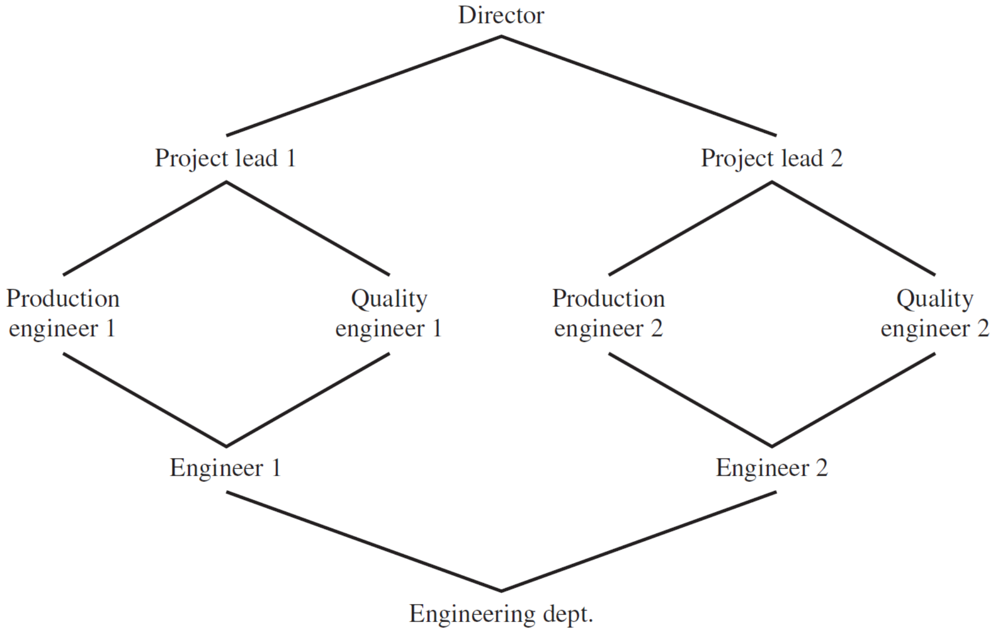


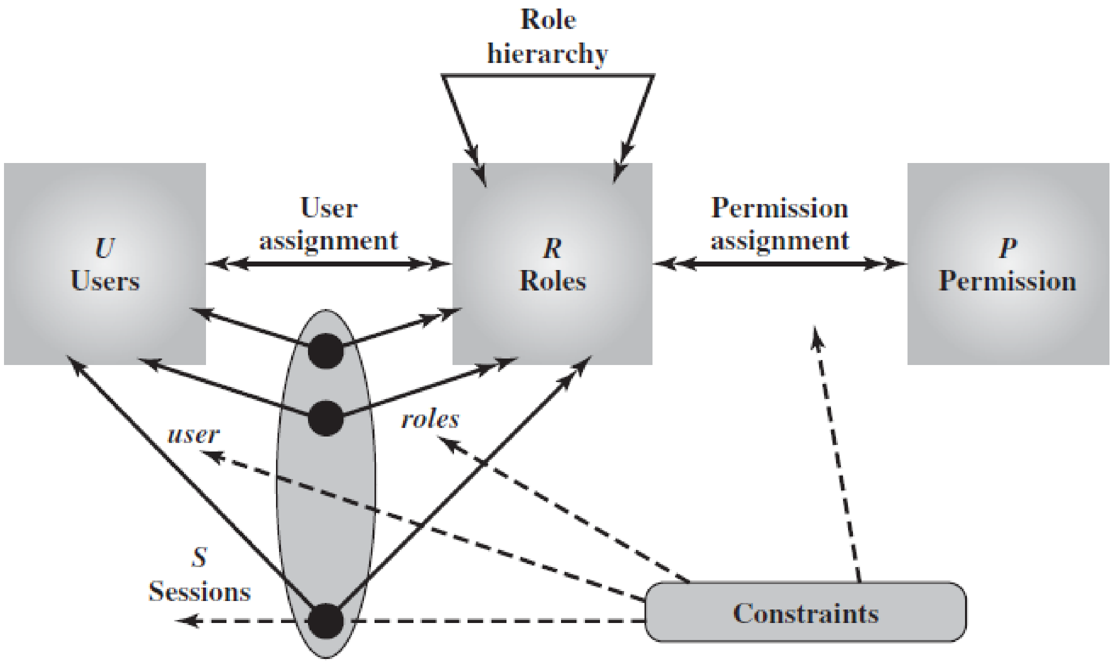
Figure 4.10 **Example of Role Hierarchy**

RBAC$_2$

- Constrains
  - Mutually exclusive roles: separation of duties
    - Purchasing manager and accounts payable manager cannot be the same user
  - Cardinality: maximum number w.r.t. roles
    - Maximum number of users for a given role
    - Maximum number of roles for a given user
    - Maximum number of roles for a permission
  - Prerequisite
    - A user can be assigned a particular role only if the user has been assigned to some other specific role(s)

$RBAC_3$

- Include $RBAC_1$ and $RBAC_2$, and by transitivity, $RBAC_0$



Attribute based access control (ABAC)

- This is a relatively new approach to access control.
- In this approach we can construct more complex authorisation statements based on attributes associated with subjects, objects, operations, and the environment.
  - It doesn't have the subject focus on RBAC.
- The flexibility is it's upside, the performance cost, checks per access, it's downside.
- The textbook says: "cooperating web services and cloud computing" already have fairly high cost per access, so ABAC is a reasonable way to go.
  - eXtensible Access Control Markup Language (XAMCL).

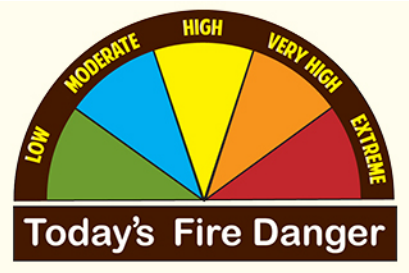
AMAC elements

- **Attributes**: Defined for entities.
- **Policy Model**: Defines the policies – so the rules and relationships for governing what’s allowed and what’s disallowed.
- **Architecture**: This is the infrastructure used to manage requests, at the policy and enforcement levels, and carry out the interactions with the sources of attributes.

Type of ABAC attributes

- Attributes are characteristics defining specific aspects.
- Subject attributes:
  - Define identity and characteristics of the subject.
  - Examples: Name, age, job title, roles, …
- Object attributes:
  - As above but for an object.
  - Examples: File name, file title, creation date, …

- Environment attributes:
  - Largely only lightly used.
  - Examples: Date, time, attacker activity,…



- ABAC supports taking inputs from a range of sources to determine the values of attributes that are needed to make access control decisions.

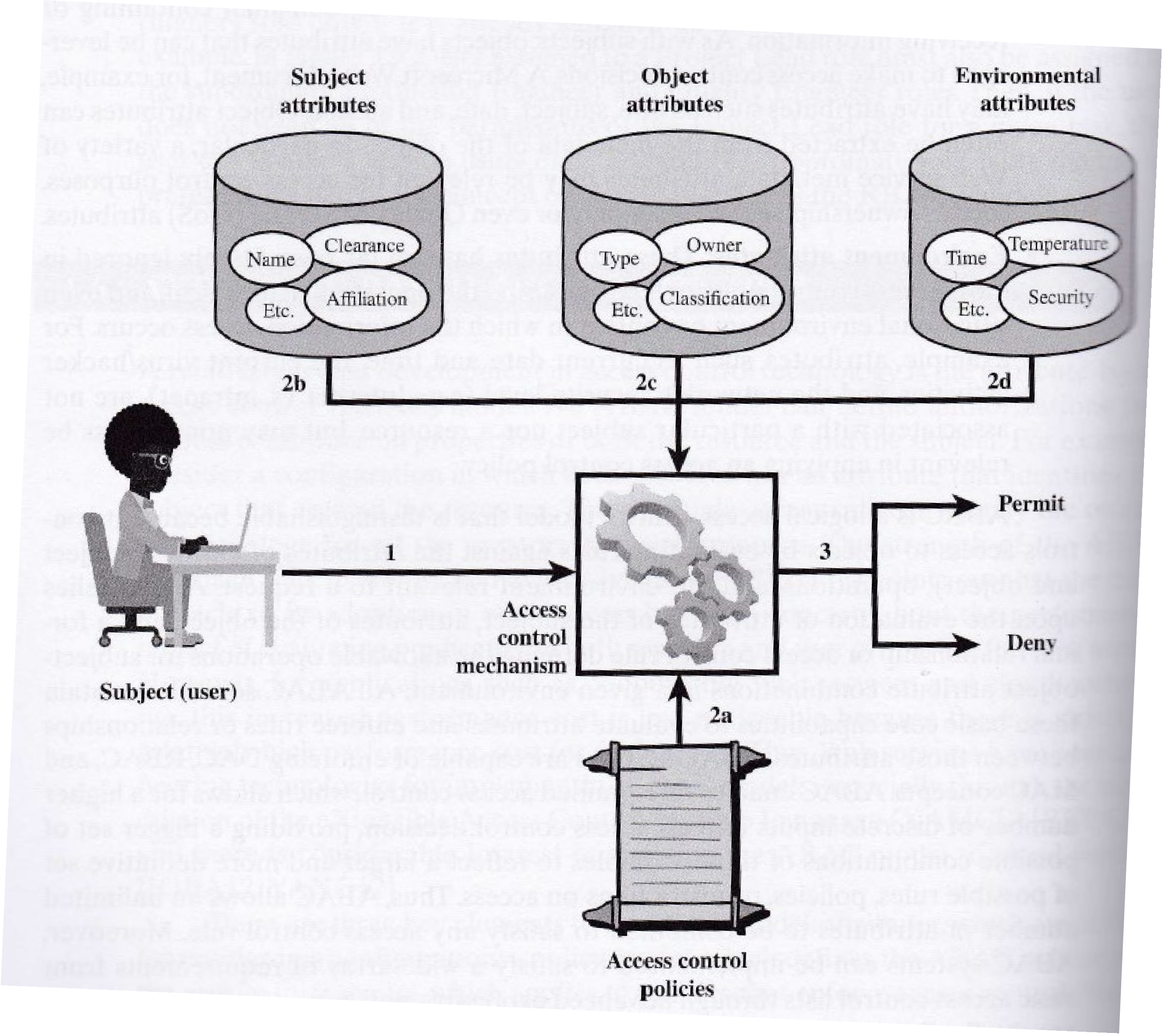


Figure 4.10: ABAC Scenario
Page 150 of the textbook.

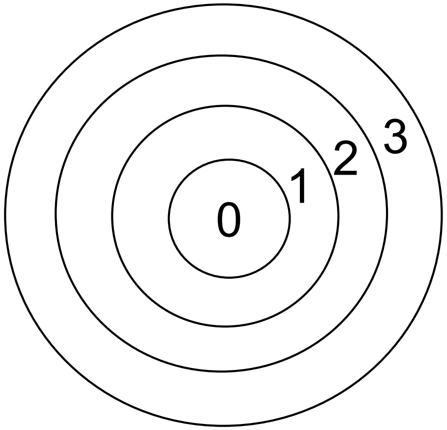
- **Protection rings**
  - Each subject or each object is assigned a number depending on its importance.
  - For an OS, it could be:

    0: OS kernel.

    1: OS.

    2: Utilities.

    3: User processes.



  - The numbers are compared to make decisions about access control.
  - Unix, Intel processors, etc. adopt this method.

Multilevel access control

- Subjects and objects both correspond to security labels.
  - But the labels are of different sorts.
- For subjects the labels correspond to clearances, that is, every user has a clearance.
- For objects the labels correspond to classifications or sensitivity, that is, every action/operation has a sensitivity rating (like top secret).
- The access relationship between security labels of the two types are governed by a series of rules.
- Multilayer models are also referred to as *data flow models*.

- A **Security label** is a set of information security attributes bound to an object or a subject.
  - The most common use is in supporting multi-level access control policies.
- When a subject makes an access request, a label is generated and attached to the request, by a trusted process.
  - Each object has a label bound to it, identifying it with a classification level.
- To process a request, a security server in the object environment compares the request label with the object label and applies policy rules (e.g. Bell-LaPadula rules) to decide whether to grant or deny access.

- Labels generated in one security domain may or may not be significant in another domain:
  - For example:

    Consider labels of identical format in two different organisations. Information classified as confidential in the first organization should generally not be disclosed to the persons with confidential clearance in the other organization.
  - Access control models taking into account such “horizontal” structures are referred to as multi-lateral.

Multilevel policies

- These are useful for organizations with various levels of sensitivity for their data:
  - Military organizations.
  - Banks.
- We will look at an example in military organisations but the same approach can be used in other cases.
- Users are given various levels of clearance.
- Objects have different levels of sensitivity.
- The access rights of a subject to an object is determined based on these two parameters.

Example:

- The sensitivity levels are:

  top secret

  secret

  confidential

  unclassified

A piece of information can be accessed by someone whose clearance level is as high as the object.

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. See 3a_Bell-LaPadula-1973.
- 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.
      - The full lattice will turn up later.

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, which we will deal with that later.
    - 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 $(S_i,O_j,\mathrm{read}) \in b$, we have that $f_o(O_j) \leq f_c(S_i)$.

- * – property:

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

  $(S_i,O_j,\text{read/write})$ 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 $(A \leq B$ AND $B \leq A)$

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

BLP properties: … and 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

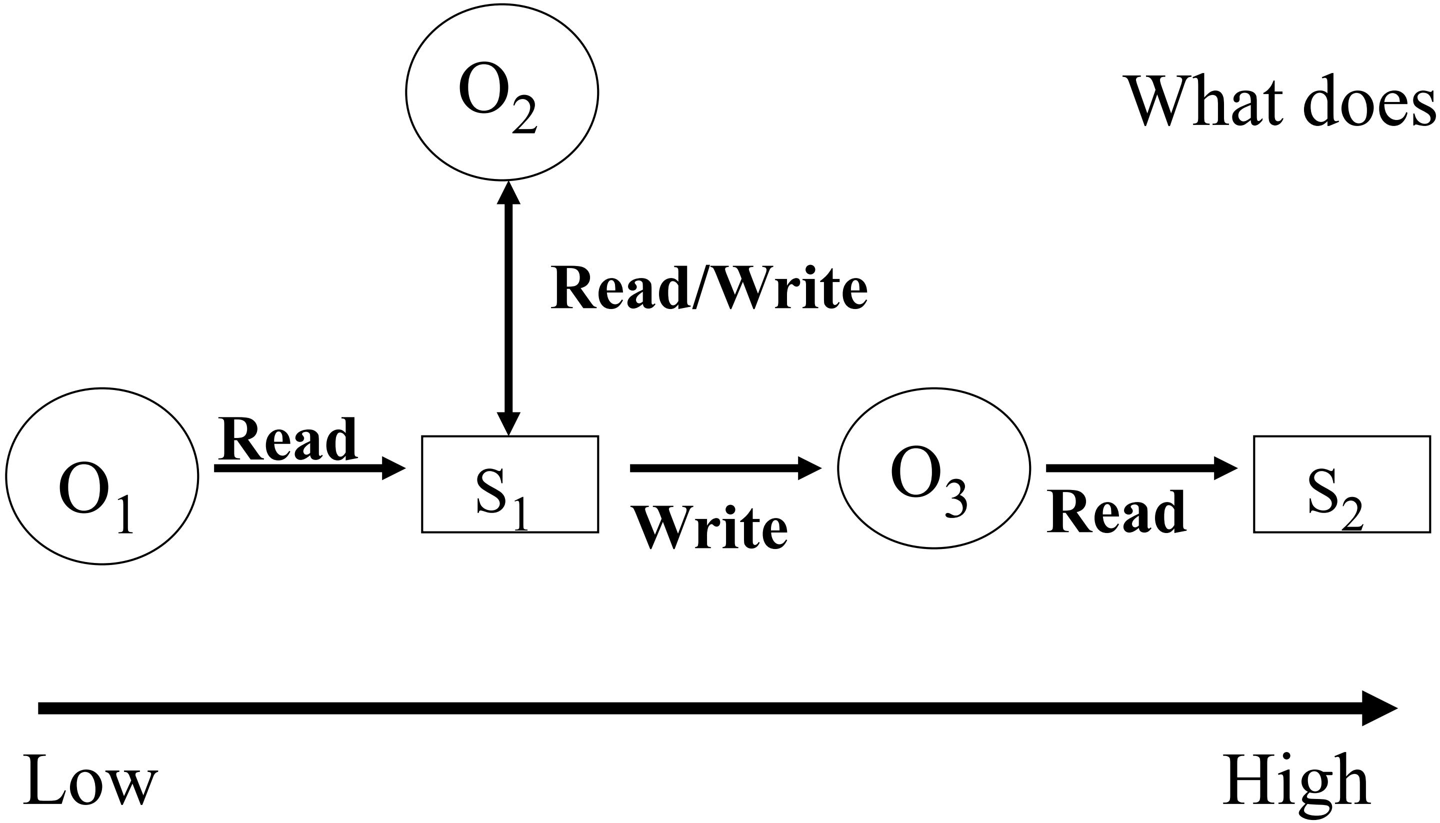
	Personnel file	Email file	Activity log	Phone list
Tam	Read, Write			
Sam		Read		
Cam				
Uma			Write	

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.

Operations in BLP

- The description we have so far is static.
- It lets us represent the state of a system at a particular time.
- We also need dynamic operations describing transitions between states.
- Remember the notes on the Harrison-Ruzzo-Ullman Model.
- These include:
  - Adding or removing elements of b, the current access set.
  - Changing object or subjects levels.
  - Changing access permissions in the access matrix M.
  - Creating or deleting objects.
- All these transitions should be carried out in accordance with the BLP rules.

Here goes a static diagram!



What does it mean?

- O_1 can be read by S_1.
- S_1 can read/write with respect to O_2.
- S_1 can write to O_3.
- O_3 can be read by S_2.

CSCI262
System Security (S3b)

Access control II:

Security models.

Beyond BLP

Overview

- Security models:
  - Traditional (BLP).
- Integrity based security models.
  - The Biba Model.
  - The Clark-Wilson Model.
- Multilateral models.
  - Lattice models.
  - Lippner’s Model.
- Other models:
  - The Chinese-Wall Model.

Security Models

- A security model is a precise representation of the security requirements (Security Policy). BLP is a security model for access control.
- 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, such as BLP, 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

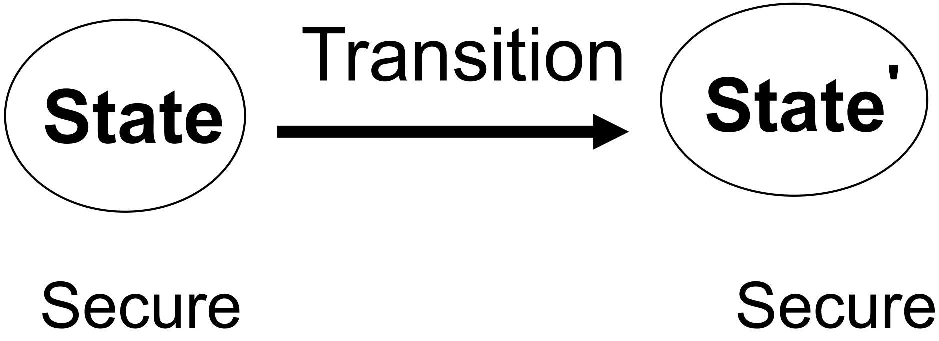
Security Relevant State Variables:
<Subjects, Objects, Security Attributes>

State Machine based Security Model

To prove security of a particular state machine based system we use the following procedure: (whether we are using BLP or not!)

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.

The start of an example

$S = \{S_1, S_2, S_3\}$
$O = \{O_1, O_2, O_3\}$
A = {**r**ead, **w**rite, **e**xecute}

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:
(a) ss-property.
(b) *-property.
(c) ds-property.

Defining transitions and an initial state requires an application domain.

Confidentiality based access control

- BLP is a confidentiality based access control model.
    - A subject cannot convey information to a subject at a lower level.
    - In other words, a confidentiality based model protects against unauthorized disclosure of information.

… 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). See 3b_Biba-1977.pdf.
- 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!”

- This policy is used in the static version of the Biba model, but not in the dynamic version which we won’t consider in detail.

- 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.

- 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.
- Note: in the dynamic models, integrity levels of subjects and objects are modified when they come into contact with other entities.

Clark-Wilson

- Clark-Wilson is another integrity based access model. See 3b_Clark-Wilson-1987.pdf.
- 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.*"
  - Stallings/Brown (Figure 13.5, page 434) reproduce a diagram from Clark-Wilson that summarises the integrity rules in Clark-Wilson.

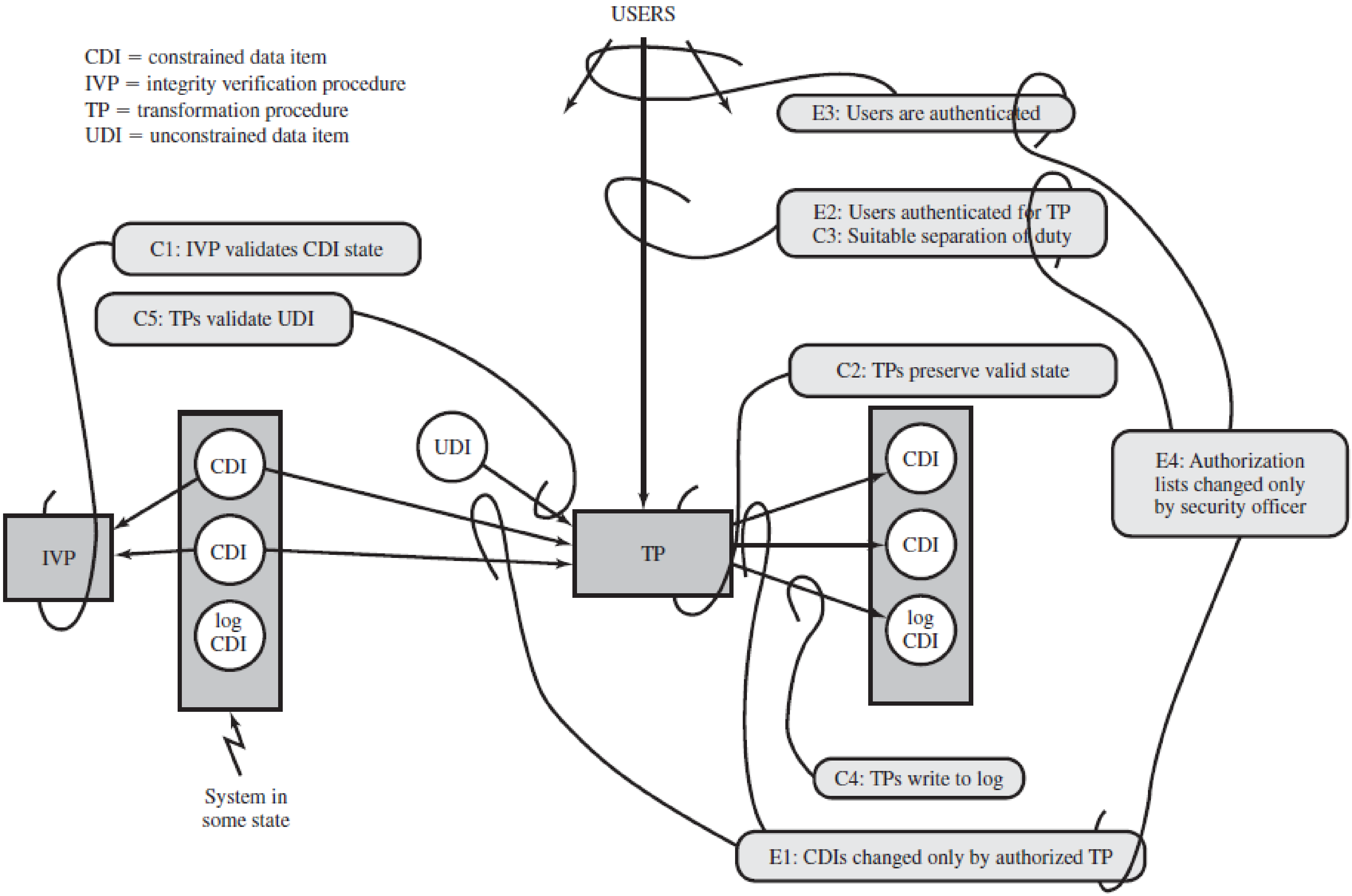
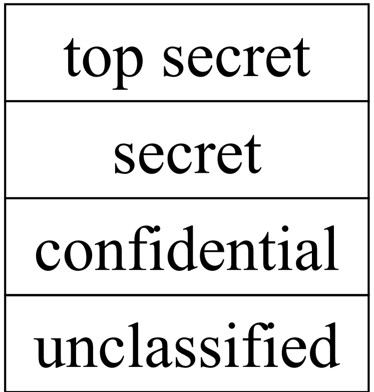
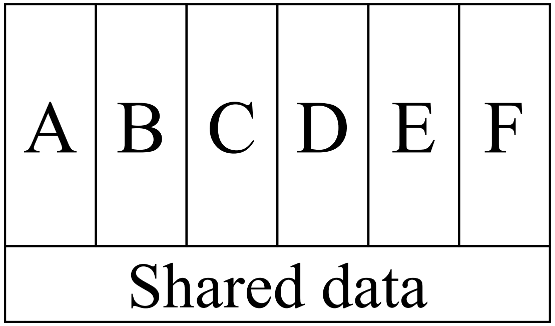

USERS
CDI = constrained data item
IVP = integrity verification procedure
TP = transformation procedure
UDI = unconstrained data item
E3: Users are authenticated
E2: Users authenticated for TP
C3: Suitable separation of duty
C1: IVP validates CDI state
C5: TPs validate UDI
C2: TPs preserve valid state
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 13.5 Summary of Clark-Wilson System Integrity Rules
Source: [CLAR87].

Multilateral models

- When we were working towards the BLP model we described multilevel models.
  - Multilevel with respect to the objects (data) and with respect to the subjects (people/processes).
  - Information flow is up/down.
- This is okay but limited with respect to modelling many types of real systems.
  - We also need multilateral models which account for information flowing sideways. We will look at these briefly.
  - They are, for example, particularly important in the context of database security.





The lattice models

- The proto-multilateral system is the lattice model, which was developed around the same time as BLP.
- It is actually quite similar to the BLP model, which is itself based on a lattice.
  - We looked at a somewhat simplified version.
- The lattice model wasn’t “popularised” until Denning, but there was input from many and some of that also feed into BLP. See 3b_Denning-1976.pdf.
- The lattice model is, like BLP, a confidentiality model.

Lattices…

- Multilevel is fine when we have a clear and complete ordering, but we don't always.
- We can use lattices to represent the relationship between security categories.
- They provide:
  - A mathematical structure under a relational operator.
  - A partial ordering with respect to “ ≤ “.

- Questions:
  1. Given two objects at different security levels, what is the lowest security level a subject must have to be allowed to read both objects?
  2. Given two subjects at different levels, what is the highest security level an object can have so that it still can be read by both subjects?
- Why are these questions important?
  – Principle of least privilege.
- How can we answer them?

- **Definition**: 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:

This is AND

$$a \leq u,\ b \leq u, \text{ and for all } v \in L : (a \leq v \wedge b \leq v) \rightarrow (u \leq v)$$

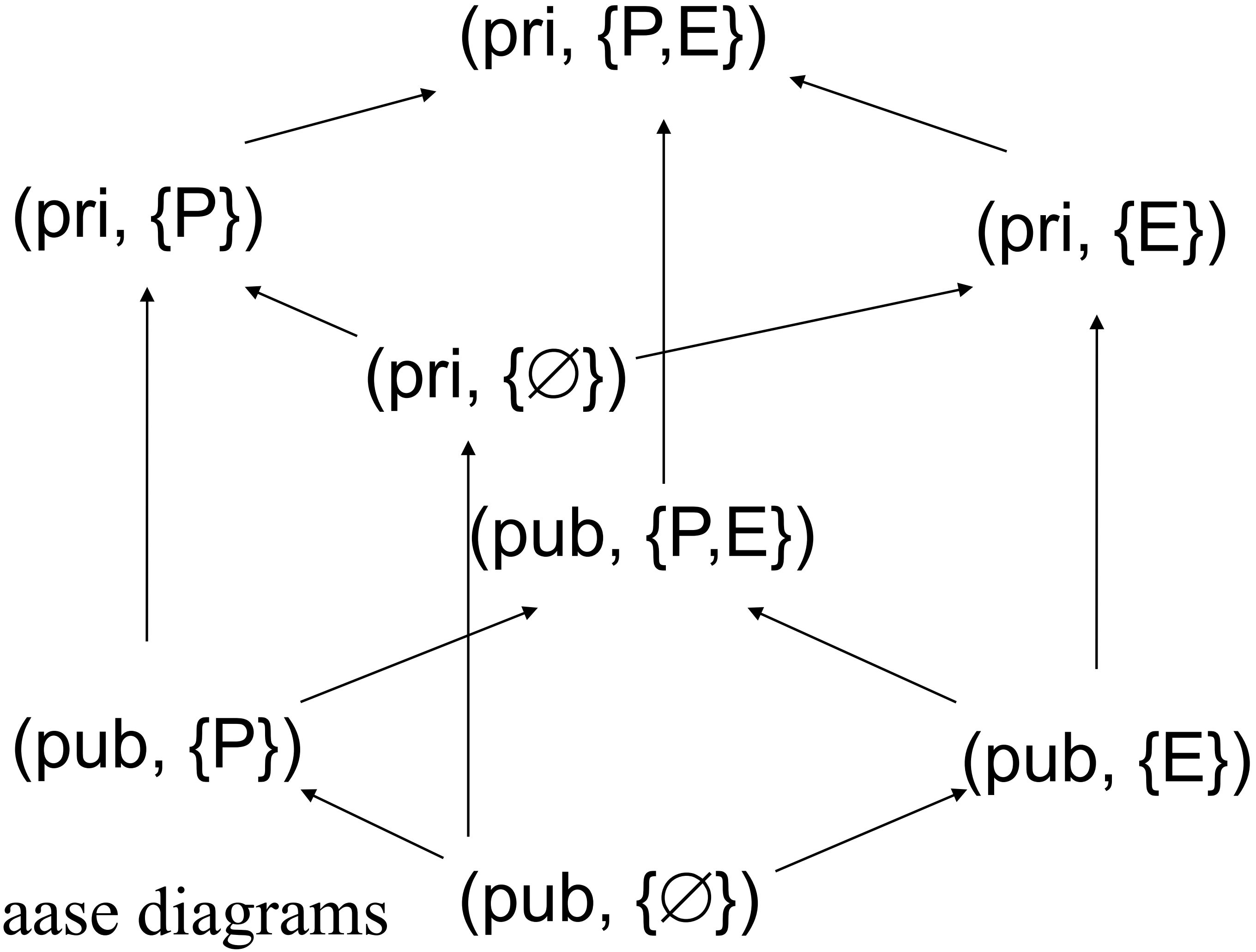
$$l \leq a,\ l \leq b, \text{ and for all } k \in L : (k \leq a \wedge k \leq b) \rightarrow (k \leq l)$$

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.

A lattice we could work with…



(pri, {P,E})

(pri, {P})

(pri, {E})

(pri, {∅})

(pub, {P,E})

(pub, {P})

(pub, {E})

(pub, {∅})

See Haase diagrams

The questions again ...

1. Given two objects at different security levels, what is the lowest security level a subject must have to be allowed to read both objects?
   - Least upper bound.
2. Given two subjects at different levels, what is the highest security level an object can have so that it still can be read by both subjects?
   - Greatest lower bound.

More on partial ordering…

- It seems students struggle with this concept, so here goes an informal illustration that might help.
- Three people: Alice, Alex and Bob.
- Order them alphabetically:
  - Alex, Alice and Bob.
  - This is, for this dataset, a complete ordering.
- Order them by initial only:
  - {Alice, Alex}, Bob
  - This is a partial ordering.
- Truncation or rounding in ordering will give a partial order.

Properties for confidentiality: ss-property

- **Simple Security Property**:
- For subject S, object O, and authorization or ability A, with A = read, the subject S *dominates* the classification of the object O.
- In other words, the read ability is only possible for subjects with a level higher or equal to that of the object level.
- **No read-up security policy**:
  - No subject can read data at a higher level.
- This is simple but can make it difficult to handle exceptions.
- **WARNING**: Here security means confidentiality, but confidentiality is usually only part of something being secure!
  - "Secure" needs a context.

* – property

The * doesn't mean anything. It was just a searchable string that never got replaced! ☺

- **Star-property**:
- For subject S, object O, and authorization or ability A, where A = writing, the subject S is *dominated by* the object O.
- In other words, the writing ability is only possible when subjects are writing into objects of greater or equal security level.
- **No write-down security policy**:
  - You cannot give higher level information to systems with a lower authorisation.

Problems with the * – property

- How can high ranking subjects pass any information to lower level subjects?
  - One way is to allow subjects to operate at lower ranks.
  - Another is to identify trusted subjects which are allowed to violate the * – property.

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.

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

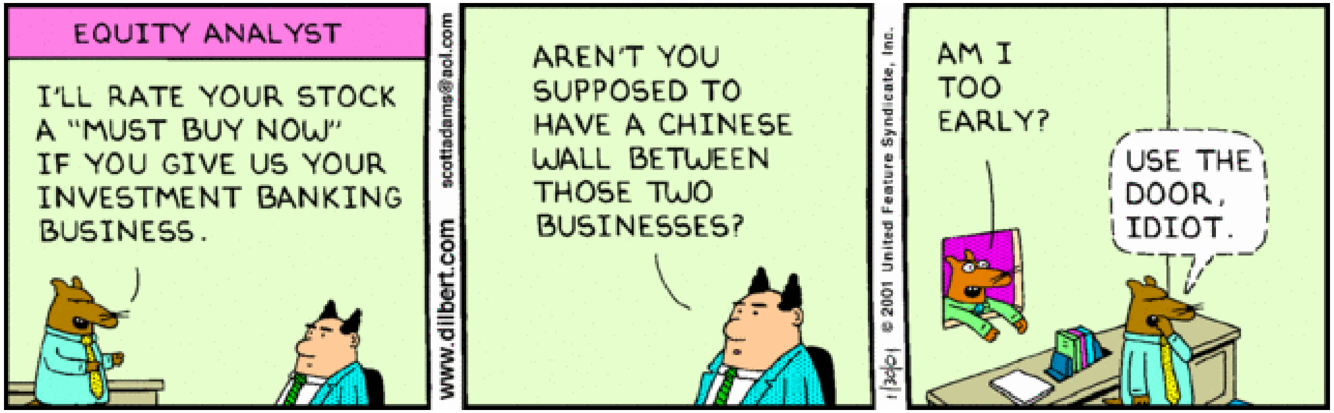
Objects	Confidentiality Clearance	Integrity Clearance / Trust
Development code/test data	(CSL, {CD})	(ISL, {ID})
Production code	(CSL, {CP})	(IO, {IP})
Production data	(CSL, {CP})	(ISL, {IP})
Software Tools	(CSL, ∅)	(IO, {ID})
System programs	(CSL, ∅)	(ISL, {IP, ID})
System programs in modification	(CSL, {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.

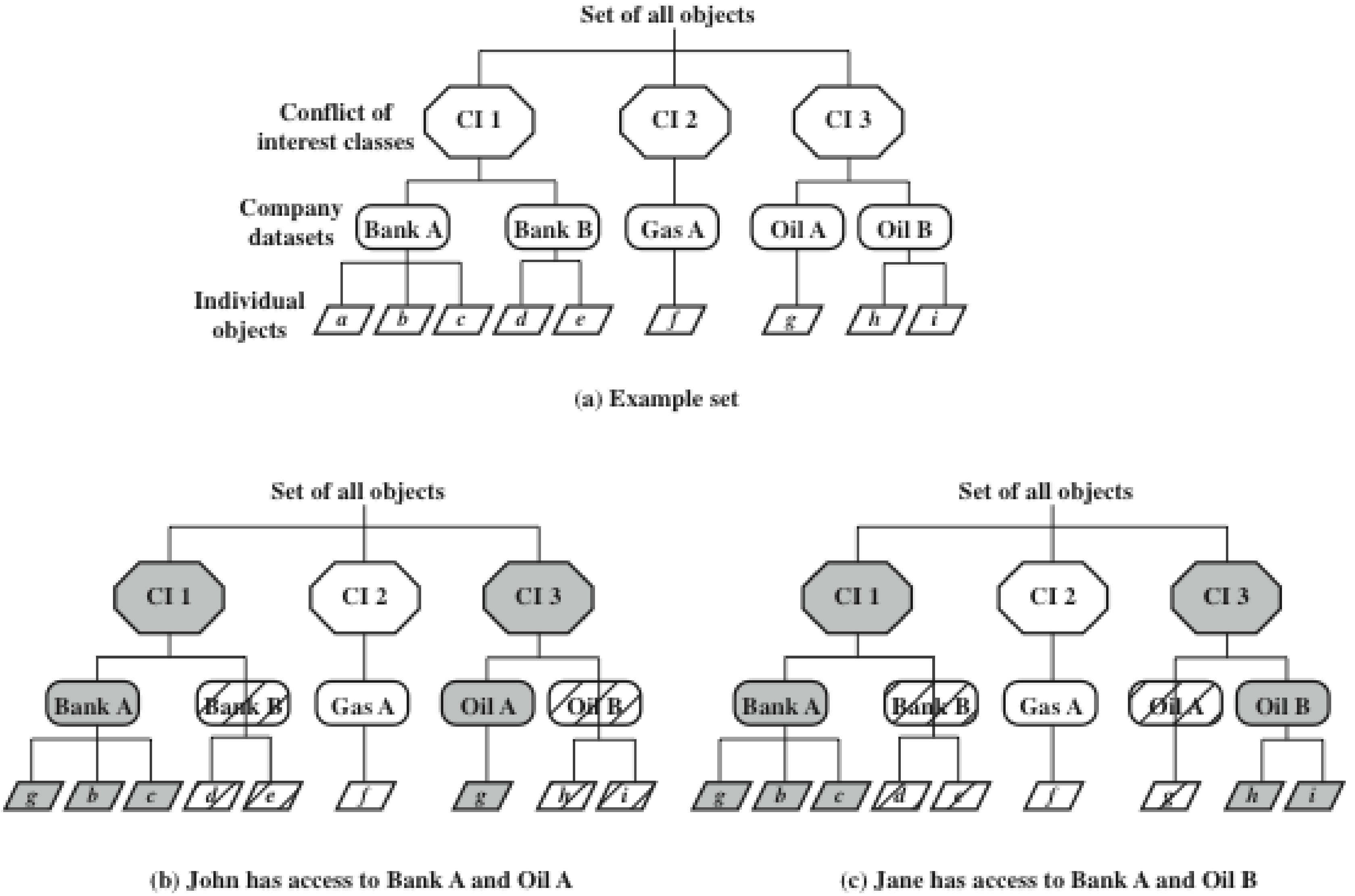
The Chinese-Wall model

- 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.
www.dilbert.com
scottadams@aol.com
AREN'T YOU SUPPOSED TO HAVE A CHINESE WALL BETWEEN THOSE TWO BUSINESSES?
1/30/01
© 2001 United Feature Syndicate, Inc.
AM I TOO EARLY?
USE THE DOOR, IDIOT.

Dilbert: 30-Jan-2001

- In the Chinese-Wall model we have conflicts of interest classes defined.
  - One might, for example, contain banks.
  - Another might, for example, contain energy industry companies.
- This model is dynamic, depending on the activities of a subject, subsequent actions will be disallowed.
- For example, if Alice consults for Bank A she won’t be allowed to consult for Bank B.



(a) Example set

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

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

Figure 13.6 Potential Flow of Information Between Two CIs

A warning … CWM Access control

- Be careful:
  - – Chinese-Wall model.
  - – Clark-Wilson model.