The Biba Model

- Contrary to Bell-LaPadula model, in Biba model information can only flow from a higher integrity class to a lower integrity class.
- L is a linearly ordered set of integrity levels
- C is a lattice of integrity categories
- Integrity levels form a linear lattice in which each level represents the classification of integrity of information an object can contain or the clearance of a subject for modifying an object.
- Integrity categories form a subset lattice and are used to enforce the need-to-have principle.
- The lattice of security classes is $L \times C$.

The Biba Model contd…

- Security with respect to integrity in the Biba model is described by the following two axioms:

- **Simple security property:** Writing information to an object o by a subject s requires that SC(s) dominates SC(o) (“no write up”).

- **The*-property:** Reading information from an object o by a subject s requires that SC(o) dominates SC(s) (“no read down”).

Multilevel Integrity (2)

- Big potential application – control systems
- E.g. in future "smart grid"
  - Safety: highest integrity level
  - Control: next level
  - Monitoring (SCADA): third level
  - Enterprise apps (e.g. billing): fourth level
- Complexity: prefer not to operate plant if SCADA system down (though you could)
- So a worm attack on SCADA can close an asset

Multilevel Integrity (3)

- LOMAC was an experimental Linux system with system files at High, network at Low
- A program that read traffic was downgraded
- Vista adopted this – marks objects Low, Medium, High or System, and has default policy of NoWriteUp
- Critical stuff is System, most other stuff is Medium, IE is Low
- Could in theory provide good protection – in practice, UAC (User account control in Windows) trains people to override it!

Comparison of two Multilevel Models

- The Bell-LaPadula Model is concerned with information confidentiality
  - subjects reading from an object must have higher a security class than the object.
  - objects being written to by a subject must have higher security class than the subject.
- The Biba model emphasizes information integrity
  - subjects writing information to an object must have higher a security class than the object.
  - objects being read from by a subject must have higher security class than the subject.

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

•Does not deal with information flow through covert channels

Overt (Explicit) Channels vs. Covert Channels

- Security objective of MLS in general, BLP in particular
  - high-classified information cannot flow to low-cleared users
- Overt channels of information flow
  - read/write an object
- Covert channels of information flow
  - communication channel based on the use of system resources not normally intended for communication between the subjects (processes) in the system

Examples of Covert Channels

- Using file lock as a shared boolean variable
- By varying its ratio of computing to input/output or its paging rate, the service can transmit information to a concurrently running process
- Covert channels are often noisy
- However, information theory and coding theory can be used to encode and decode information through noisy channels

More on Covert Channels

- Covert channels cannot be blocked by *-property
- It is generally very difficult, if not impossible, to block all cover channels
- One can try to limit the bandwidth of covert channels
- Military requires cryptographic components be implemented in hardware
  - to avoid trojan horse leaking keys through covert channels

- Requests by a subject to access an object are controlled with respect to the access class of the subject and the object and granted only if some relationship, depending on the requested access, is satisfied
- Two principles, must be satisfied 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 dominates 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

- Satisfaction of these two principles prevents information to flow from high level subjects/objects to subjects/objects at lower (or incomparable) levels, thereby ensuring the satisfaction of the protection requirements (i.e., no process will be able to make sensitive information available to users not cleared for it)

- It is important to control both read and write operations, since both can be improperly used to leak information

- Consider the earlier example of the Trojan Horse
- Possible classifications reflecting the access restrictions to be enforced could be: Secret for Vicky and Market, and Unclassified for John and Stolen
- In the respect of the no-read-up and no-write-down principles, the Trojan Horse will never be able to complete successfully
  - If Vicky connects to the system as a Secret (or Confidential) subject, and thus the application runs with a Secret (or Confidential) access class, the write operation will be blocked
  - If Vicky invokes the application as an Unclassified subject, the read operation will be blocked instead

- Given the no-write-down principle, users are allowed to connect to the system at different access classes, so that they are able to access information at different levels (provided that they are cleared for it)
- A lower class does not mean “less” privileges in absolute terms, but only less reading privileges
- Although users can connect to the system at any level below their clearance, the strict application of the no-read-up and the no-write-down principles may result too rigid

- Real world situations often require exceptions to the mandatory restrictions
  - data may need to be downgraded
  - information released by a process may be less sensitive than the information the process has read

- To respond to situations like these, multilevel systems should then allow for exceptions, loosening or waiving restrictions, in a controlled way, to processes that are *trusted* and ensure that information is *sanitized* (meaning the sensitivity of the original information is lost)

- Note also that DAC and MAC policies are not mutually exclusive, but can be applied jointly

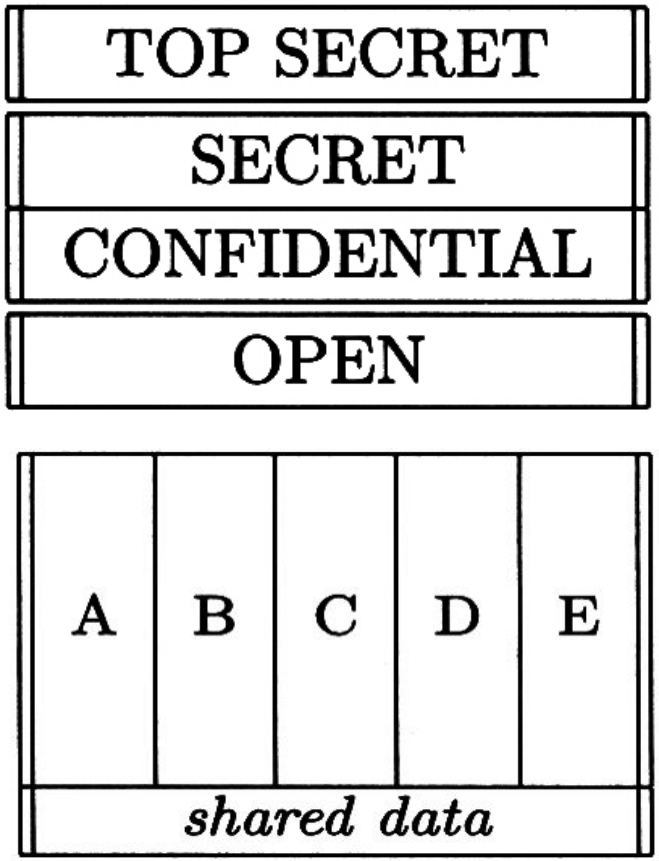
- In this case, an access to be granted needs both
  - the existence of the necessary authorization for it and
  - to satisfy the mandatory policy

- Intuitively, the discretionary policy operates within the boundaries of the mandatory policy: it can only restrict the set of accesses that would be allowed by MAC alone

Multilateral Security

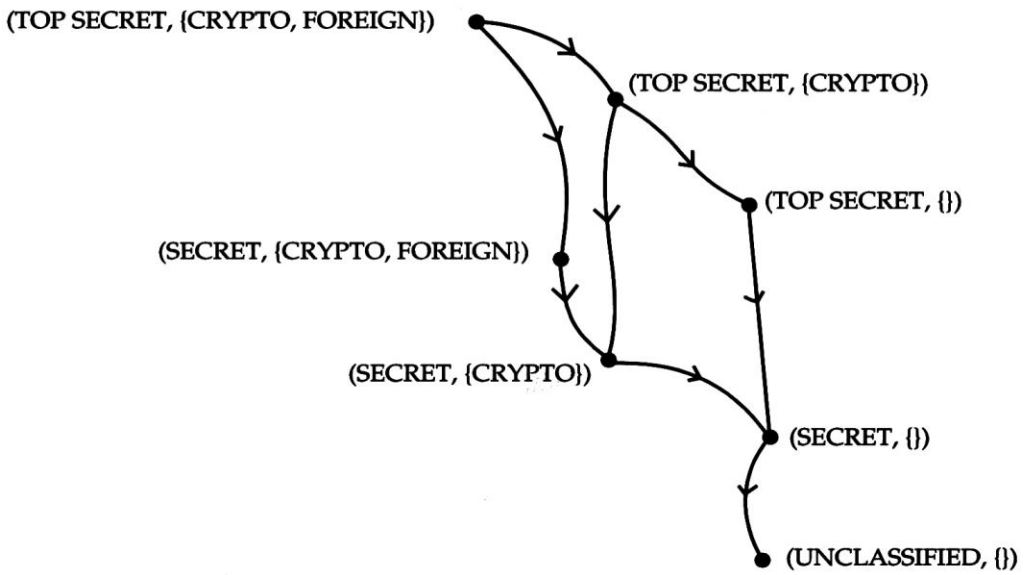
Multilateral Security

- Sometimes the aim is to stop data flowing down
- Other times, you want to stop lateral flows
- Examples:
  - Intelligence
  - Competing clients of an accounting firm
  - Medical records by practice or hospital



The Lattice Model

- This is how intelligence agencies manage ‘compartmented’ data – by adding labels
- Basic idea: BLP requires only a partial order



The Chinese Wall Model

- Industries such as investment banking, advertising and accounting have too few top firms for each big client to have its own
- So maybe you're auditing BP, and Shell too!
- Traditional control: a "Chinese Wall" rule that stops the two teams communicating
- Idea (Brewer and Nash, 1989): use a refinement of Bell-LaPadula

The Chinese Wall Model (2)

- Idea: it's not enough to stop a Shell analyst reading BP data
- Must stop a BP analyst writing data to a Barclays file that the Shell analyst can also read
- For each object O, let y(O) be the firm it relates to
- Let x(O) be that firm's conflict-of-interest class
- Let x(O) = Ø if the information has been sanitized (so anyone can see it)

The Chinese Wall Model (3): in Summary

- Then reading is allowed if the object belongs to a firm the subject has access to, or a different conflict-of-interest class

  S can read O iff for all O' to which S has access, y(O)=y(O') or x(O) $\notin$ x(O')

- Writing is allowed iff the user cannot read an object that contains unsanitised information

  S can write O iff S cannot read O' with y(O)$\neq$y(O') and x(O)$\neq \emptyset$

- Practical issues: where is the state kept? Should you automate this at all?

Chinese Wall Model

Chinese Wall Model (from Matt Bishop)

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

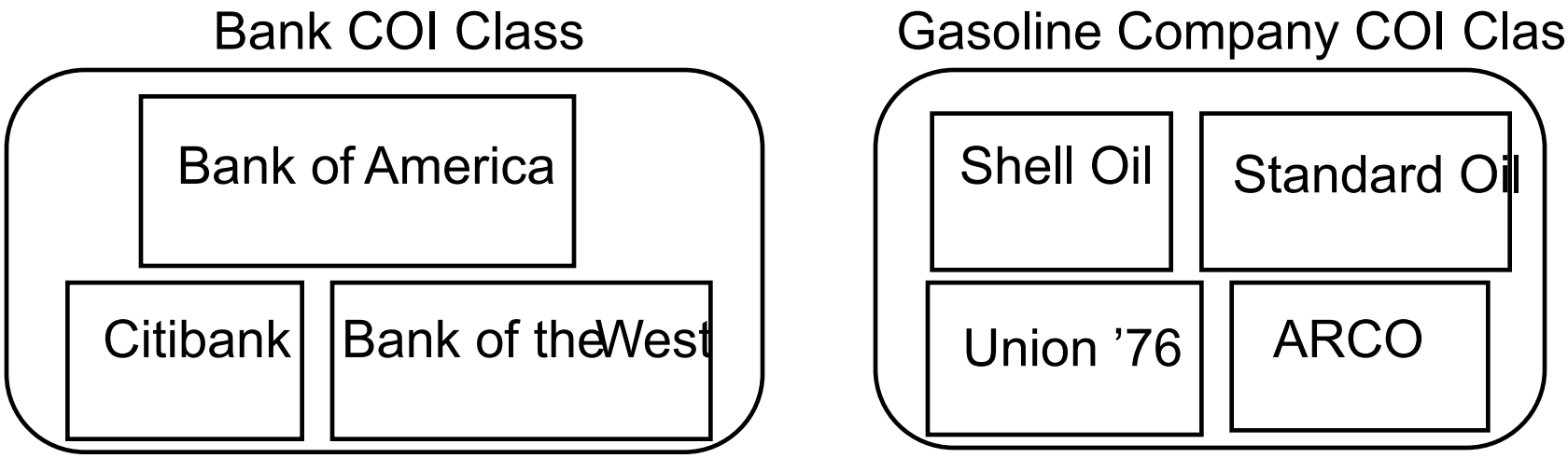
Organization

- 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 either condition holds:
  1. There is an *o′* such that *s* has accessed *o′* and *CD*(*o′*) = *CD*(*o*)
     - Meaning *s* has read something in *o*'s dataset
  2. For all *o′* ∈ *O*, *o′* ∈ *PR*(*s*) ⇒ *COI*(*o′*) ≠ *COI*(*o*)
     - Meaning *s* has not read any objects in *o*'s conflict of interest class
- Ignores sanitized data (see below)
- Initially, *PR*(*s*) = ∅, so initial read request granted

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

- *s* can write to *o* iff both of the following hold:
  1. The CW-simple security condition permits *s* to read *o*; and
  2. 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 it can read are in the same dataset

Formalism

- Goal: figure out how information flows around system
- S set of subjects, O set of objects, $L = C \times D$ set of labels
- $I_1 : O \rightarrow C$ maps objects to their COI classes
- $I_2 : O \rightarrow D$ maps objects to their CDs
- $H(s, o)$ true iff s has *or had* read access to o
- $R(s, o)$: Request from s to read o

Axioms

- Axiom 1. For all $o, o' \in O$,
  if $I_2(o) = I_2(o')$, then $I_1(o) = I_1(o')$
  - CDs do not span COIs.
- Axiom 2. $s \in S$ can read $o \in O$ iff,
  for all $o' \in O$ such that $H(s, o')$, either
  $I_1(o') \neq I_1(o)$ or $I_2(o') = I_2(o)$
  - *s* can read *o* iff *o* is either in a different COI than every other o' that *s* has read, or in the same CD as *o*.

Which Objects Can Be Read?

- Suppose $s \in S$ has read $o \in O$.
- If s can read $o' \in O$, $o' \neq o$, then $l_1(o') \neq l_1(o)$ or $l_2(o') = l_2(o)$.
  - Says s can read only the objects in a single CD within any COI

Proof

Assume false. Then

$H(s, o) \wedge H(s, o') \wedge I_1(o') = I_1(o) \wedge I_2(o') \neq I_2(o)$

Assume s read o first.

Then $H(s, o)$ when s read o, so by Axiom 2,

either $I_1(o') \neq I_1(o)$ or $I_2(o') = I_2(o)$, so

$(I_1(o') \neq I_1(o) \vee I_2(o') = I_2(o)) \wedge (I_1(o') = I_1(o) \wedge I_2(o') \neq I_2(o))$

Rearranging terms,

$(I_1(o') \neq I_1(o) \wedge I_2(o') \neq I_2(o) \wedge I_1(o') = I_1(o)) \vee$

$(I_2(o') = I_2(o) \wedge I_2(o') \neq I_2(o) \wedge I_1(o') = I_1(o))$

which is obviously false, contradiction.

Lemma

- Suppose a subject $s \in S$ can read an object $o \in O$. Then s can read no o' for which $l_1(o') = l_1(o)$ and $l_2(o') \neq l_2(o)$.
  - So a subject can access at most one CD in each COI class
  - Sketch of proof:
    - Initial case follows from Axioms 3-4.
    - If $o' \neq o$, theorem immediately gives lemma.

COIs and Subjects

- Theorem: Let $c \in C$ and $d \in D$. Suppose there are n objects $o_i \in O$, $1 \le i \le n$, such that $I_1(o_i) = d$ for $1 \le i \le n$, and $I_2(o_i) \ne I_2(o_j)$, for $1 \le i, j \le n$, $i \ne j$. Then for all such o, there is an $s \in S$ that can read o iff $n \le |S|$.
  - If a COI has n CDs, you need at least n subjects to access every object
  - Proof sketch: If s can read o, it cannot read any o' in another CD in that COI (Axiom 2). As there are n such CDs, there must be at least n subjects to meet the conditions of the theorem.

Sanitized Data

- $v(o)$: sanitized version of object o
  - For purposes of analysis, place them all in a special CD in a COI containing no other CDs
- Axiom 5. $I_1(o) = I_1(v(o))$ iff $I_2(o) = I_2(v(o))$

Which Objects Can Be Written?

- Axiom 6. $s \in S$ can write to $o \in O$ iff the following hold simultaneously
  1. $H(s, o)$
  2. There is no $o' \in O$ with $H(s, o')$, $I_2(o) \neq I_2(o')$, $I_2(o) \neq I_2(v(o))$, $I_2(o') = I_2(v(o))$.
  - Allow writing iff information cannot leak from one subject to another through a mailbox
  - Note handling for sanitized objects

How Information Flows

- Definition: Information may flow from *o* to *o′* if there is a subject such that *H*(*s*, *o*) and *H*(*s*, *o′*).
  - Intuition: if *s* can read 2 objects, it can act on that knowledge; so information flows between the objects through the nexus of the subject
  - Write the above situation as (*o*, *o′*)

Key Result

- Set of all information flows is

  $\{ (o, o') \mid o \in O \wedge o' \in O \wedge I_2(o) = I_2(o') \vee I_2(o) = I_2(v(o)) \}$

- Sketch of proof: Definition gives set of flows:

  $F = \{(o, o') \mid o \in O \wedge o' \in O \wedge \exists\ s \in S$ such that $H(s, o) \wedge H(s, o'))\}$

  Axiom 6 excludes the following flows:

  $X = \{ (o, o') \mid o \in O \wedge o' \in O \wedge I_2(o) \neq I_2(o') \wedge I_2(o) \neq I_2(v(o)) \}$

  So, letting F^* be transitive closure of F,

  $F^* - X = \{(o, o') \mid o \in O \wedge o' \in O \wedge \neg(I_2(o) \neq I_2(o') \wedge I_2(o) \neq I_2(v(o))) \}$

  which is equivalent to the claim.

Comparison with Bell-LaPadula

- Fundamentally different
  - CW has no security labels, B-LP does
  - CW has notion of past accesses, B-LP does not
- Bell-LaPadula can capture state at any time
  - Each (COI, CD) pair gets security category
  - Two clearances, *S* (sanitized) and *U* (unsanitized)
    - *S dom U*
  - Subjects assigned clearance for compartments without multiple categories corresponding to CDs in same COI class

Comparison with Bell-LaPadula

- Bell-LaPadula cannot track changes over time
  - Susan becomes ill, Anna needs to take over
    - C-W history lets Anna know if she can
    - No way for Bell-LaPadula to capture this
- Access constraints change over time
  - Initially, subjects in C-W can read any object
  - Bell-LaPadula constrains set of objects that a subject can access
    - Can’t clear all subjects for all categories, because this violates CW-simple security condition

Clinical Information Systems Security Policy

Clinical Information Systems Security Policy

- Intended for medical records
  - Conflict of interest not critical problem
  - Patient confidentiality, authentication of records and annotators, and integrity are
- Entities:
  - Patient: subject of medical records (or agent)
  - Personal health information: data about patient's health or treatment enabling identification of patient
  - Clinician: health-care professional with access to personal health information while doing job

Assumptions and Principles

- Assumes health information involves 1 person at a time
  - Not always true; OB/GYN involves father as well as mother
- Principles derived from medical ethics of various societies, and from practicing clinicians

Access

- Principle 1: Each medical record has an access control list naming the individuals or groups who may read and append information to the record. The system must restrict access to those identified on the access control list.
  - Idea is that clinicians need access, but no-one else. Auditors get access to copies, so they cannot alter records

Access

- Principle 2: One of the clinicians on the access control list must have the right to add other clinicians to the access control list.
  - Called the *responsible clinician*

Access

- Principle 3: The responsible clinician must notify the patient of the names on the access control list whenever the patient's medical record is opened. Except for situations given in statutes, or in cases of emergency, the responsible clinician must obtain the patient's consent.
  - Patient must consent to all treatment, and must know of violations of security

Access

- Principle 4: The name of the clinician, the date, and the time of the access of a medical record must be recorded. Similar information must be kept for deletions.
  - This is for auditing. Don't delete information; update it (last part is for deletion of records after death, for example, or deletion of information when required by statute). Record information about all accesses.

Creation

- Principle: A clinician may open a record, with the clinician and the patient on the access control list. If a record is opened as a result of a referral, the referring clinician may also be on the access control list.
  - Creating clinician needs access, and patient should get it. If created from a referral, referring clinician needs access to get results of referral.

Deletion

- Principle: Clinical information cannot be deleted from a medical record until the appropriate time has passed.
  - This varies with circumstances.

Confinement

- Principle: Information from one medical record may be appended to a different medical record if and only if the access control list of the second record is a subset of the access control list of the first.
  - This keeps information from leaking to unauthorized users. All users have to be on the access control list.

Aggregation

- Principle: Measures for preventing aggregation of patient data must be effective. In particular, a patient must be notified if anyone is to be added to the access control list for the patient's record and if that person has access to a large number of medical records.
  - Fear here is that a corrupt investigator may obtain access to a large number of records, correlate them, and discover private information about individuals which can then be used for nefarious purposes (such as blackmail)

Enforcement

- Principle: Any computer system that handles medical records must have a subsystem that enforces the preceding principles. The effectiveness of this enforcement must be subject to evaluation by independent auditors.
  - This policy has to be enforced, and the enforcement mechanisms must be auditable (and audited)

Compare to Bell-LaPadula

- Confinement Principle imposes lattice structure on entities in model
  - Similar to Bell-LaPadula
- CISS focuses on objects being accessed; B-LP on the subjects accessing the objects
  - May matter when looking for insiders in the medical environment

Originator Controlled Access Control (ORCON)

- Problem: organization creating document wants to control its dissemination
  - Example: Secretary of Agriculture writes a memo for distribution to her immediate subordinates, and she must give permission for it to be disseminated further. This is “originator controlled” (here, the “originator” is a person).

Requirements

- Subject $s \in S$ marks object $o \in O$ as ORCON on behalf of organization X. X allows o to be disclosed to subjects acting on behalf of organization Y with the following restrictions:
  1. o cannot be released to subjects acting on behalf of other organizations without X's permission; and
  2. Any copies of o must have the same restrictions placed on it.

DAC Fails

- Owner can set any desired permissions
  - This makes 2 unenforceable

MAC Fails

- First problem: category explosion
  - Category *C* contains *o*, *X*, *Y*, and nothing else. If a subject $y \in Y$ wants to read *o*, $x \in X$ makes a copy *o′*. Note *o′* has category *C*. If *y* wants to give $z \in Z$ a copy, *z* must be in *Y*—by definition, it's not. If *x* wants to let $w \in W$ see the document, need a new category *C′* containing *o*, *X*, *W*.
- Second problem: abstraction
  - MAC classification, categories centrally controlled, and access controlled by a centralized policy
  - ORCON controlled locally

Combine Them

- The owner of an object cannot change the access controls of the object.
- When an object is copied, the access control restrictions of that source are copied and bound to the target of the copy.
  - These are MAC (owner can’t control them)
- The creator (originator) can alter the access control restrictions on a per-subject and per-object basis.
  - This is DAC (owner can control it)