Chapter 4

Mandatory-Access-Control Model

Introduction

Mandatory-access control (MAC) stands as a well-established model in computing security. Despite the fact that it lends itself well to military environments, it represents clearly distinguishing aspects in controlling information flow. Such information flow is foremost characterized as being deterministic. We begin with an introductory to the foundations of information flow. We describe the mathematical elements underpinning MAC as a lattice-based information-flow model. Subsequently, we discuss the details of the Bell-LaPadula and the Biba models. The first one is based on the need to preserve confidentiality of information flow, while the second is concerned with maintaining integrity. We compare the two models and describe scenarios in which they can be combined. Finally, we introduce the Chinese-wall policy as an instance of the lattice-based information-flow policy applicable in commercial environments.

Mandatory-Access-Control Theory

In a system governed by the mandatory-access-control model, user privileges are not resource-owner centric. In fact, no concept of ownership does exist in MAC, which is rather based on a policy that is driven by the sensitivity of the protected information. To access a MAC-protected object, one must hold the proper security clearance required by that object. The security label of a resource is matched up against the clearance of an attempting accessor. MAC policies fall under what is known as *lattice-based access-control system.* Information flow in these systems is formally determined by the mathematical structure of the underlying lattice that reflects it. We begin by reviewing the foundations behind the MAC model.

Partial Orders

A set S is said to be *partially ordered* along a binary relationship R between S and itself if and only if the following conditions are satisfied:

- R is reflexive: $a\ R\ a$ for every element a in S.
- R is transitive: if $a\ R\ b$ and $b\ R\ c$, then $a\ R\ c$.
- R is antisymmetric: if $a\ R\ b$ and $b\ R\ a$, then $a = b$.

A partially ordered set is sometimes referred to in the literature as a *poset* for short. Note that it is not required that every pair of elements in a partially ordered set to be related, and hence the use of the term *partial ordering*. When every pair of elements x and y of a partially ordered set S can be compared with each other (i.e., $x\ R\ y$ or $y\ R\ x$) the set S becomes a *totally ordered* set also referred to as a *linearly ordered* set or simply an *ordered set*.

Example: Partial Orders

Consider the elements of set S to be the subsets of $\{a,b,c\}$ and R to be the containment relationship denoted by $\subseteq$. The set:
$S = \{\Phi,\{a\},\{b\},\{c\},\ \{a,b\},\{a,c\},\{b,\ c\},\{a,b,c\}\}$ forms a partial order along the relationship $\subseteq$ because

- $\subseteq$ is reflexive: for every element x in S, $x \subseteq x$.
- $\subseteq$ is transitive for every x, y, and z in S, $x \subseteq y$ and $y \subseteq z \Rightarrow x \subseteq z$.
- $\subseteq$ is antisymmetric: for every pair of elements x and y in S, $x \subseteq y$ and $y \subseteq x \Rightarrow x = y$.

Similarly, $(Z, \leq)$ is a total order, where Z is the set of negative and non-negative integers.

Lattices

A *lattice* is a partially ordered set in which all nonempty finite subsets have a *least upper bound* and a *greatest lower bound*. If $\leq$ denotes a partial order over S, then the *least upper bound* and the *greatest lower bound* of a subset V of S are, respectively, defined as follows:

- A least upper bound of V, denoted by *lub*, is an element u in S such that $x \leq u$ for all x in V, and
  For any y in S such that $x \leq y$ for all x in V, it holds that $u \leq y$
- A greatest lower bound of V, denoted by *gub*, is an element l in S such that
  $l \leq x$ for all x *in* V, and
  for any y in S such that $y \leq x$ for all x in V, it holds that $y \leq l$.

In particular, every two elements of a lattice have a least upper bound and a greatest lower bound. It can be easily shown that the least upper bound and greatest lower bound of any set are always unique: if x and y are both a least upper bound of V, then it follows that $x \leq y$ and $y \leq x$, and since $\leq$ is antisymmetric, it follows that $x = y$.

Example: Lattices

The poset $(P(S), \subseteq)$, where $P(S)$ is the power set (all possible subsets of a three-element set S), forms a lattice. Every pair of elements x and y in $P(S)$ has a unique least upper bound given by $x \cup y$ and a unique greatest lower bound given by $x \cap y$. Both of these bounds are computed based on the $\subseteq$ relationship. By definition, for every x and y in $P(S)$, if $u = gub(x, y) = x \cup y$, then x and y are necessarily contained in u, and for every other subset of S (say, s) containing both x and y, it implies that u is contained in s. Similarly, if $l = lub(x, y) = x \cap y \Rightarrow l \subseteq x$ and $l \subseteq y$ and for every s in $P(S)$ if $s \subseteq x$ and $s \subseteq y \Rightarrow s \subseteq l = x \cap y$. Figure 4.1 depicts a poset constructed from $S = \{a,b,c\}$.

Lattice-Based Access-Control Models

Predicting the paths of information flow is central to maintaining confidentiality and integrity of data. When information access in a protected system is modeled along a lattice structure, any policies dealing with control of information flow are directly reflected by the lattice. Lattice-based access control is an essential aspect of computing security in environments requiring stringent information-flow controls.

In lattice-based protection systems, information-flow policies bind system objects and subjects to security classes. Flow of information from one object to another is thereafter governed by this binding. Denning [DENN76b] formally defines an information-flow model denoted by FM as

$$FM = < N, P, SC, \oplus, \rightarrow >,$$

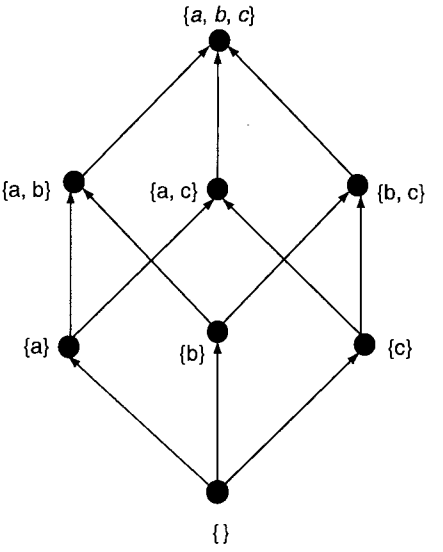

FIGURE 4.1 A depiction of the lattice corresponding to the poset $(P(\{a,b,c\}), \subseteq)$

where $N = \{a,b,...\}$ is a finite set of system resources (objects) that includes users that are, in effect, active objects of the system. $P = \{p,q,...\}$ is the set of system processes running on behalf of users. $SC = \{A,B,...\}$ is a finite set of *security classes* corresponding to disjoint classes of information containers. An example of SC corresponds to the classification:

$$SC = \{TOPSECRET, SECRET, CONFIDENTIAL, UNCLASSIFIED\}.$$

Each object $o \in N$ is statically or dynamically bound to a security class $O \in SC$. As a result, each process $p \in P$ is also bound to a security class from SC. We adopt the notation of using upper-case characters to indicate a security class while a corresponding lower-case character represents an object bound to that security class.

The class combining binary operator defined within $SC \times SC$ to SC, $\oplus$ is associative—that is,

$$A \oplus B \oplus C = A \oplus (B \oplus C) = (A \oplus B) \oplus C \text{ for all } A, B, C \in SC$$

and is commutative—that is,

$$A \oplus B = B \oplus A \text{ for all } A, B \in SC.$$

Applying the $\oplus$ operator to any pair of security classes A and B yields the security class to which information derived from security classes A and/or B belongs. The security class corresponding to any function that operates on objects from classes A and B is thus $A \oplus B$. By an intuitive extension, the class of a transformation by an n-ary function $f(a_1, ..., a_n)$ is $A_1 \oplus A_2 \oplus ... A_n$.

The flow relationship of $\rightarrow$ is defined over the elements of $SC \times SC$ and is essentially what defines an information-flow policy. The notation $A \rightarrow B$ is used to indicate the fact that information contained in an object whose security class is A may flow to an object that has security class of B. Simply stated $A \rightarrow B$ if and only if information from class A is permitted to flow into class B through some kind of transfer. The information-flow model as such is said to be secure if and only if any execution of a sequence of operations in the system yields a state of information flow that is consistent with a predefined flow policy expressed in terms of the $\rightarrow$ relationship. If a data value resulting from a series of operations denoted by function $f(a_1, ..., a_n)$ flows to an object b that is statically bound to security class B, then $A_1 \oplus A_2 \oplus ... A_n \rightarrow B$ must hold as part of the stated flow policy.

The Lattice Structure of the Information Flow Model

Denning's observation in her landmark paper [DENN76b] established a set of axioms for which $< SC, \rightarrow, \oplus >$ forms a universally bounded lattice. Such a lattice consists of a finite partially ordered set that has a least-upper bound operator and a lower upper-bound operator with respect to the flow

relationship $\rightarrow$. These axioms or rather assumptions are implied by the intuitive semantics of information flow and are stated as follows:

1. $< SC, \rightarrow >$ is a partially ordered set.
2. SC is a finite set.
3. SC has a lower bound L with respect to the $\rightarrow$ relationship.
4. The join operator $\oplus$ is a least upper bound that is totally defined over SC.

The rationale behind these intuitive assumptions is discussed in the following:

- *First axiom of Denning's information flow* SC along with the binary relationship $\rightarrow$ yields a partially ordered set. This result is evidenced by the nature of information flow.

1. The relationship $\rightarrow$ is reflexive (i.e., $A \rightarrow A$ for every $A \in SC$). The source containing information and the receptacle destination of information are the same object. It is evident that information flow is permitted from object a to itself. Otherwise, an inconsistency in the definition of the $\rightarrow$ relationship arises.
2. The relationship $\rightarrow$ is transitive (i.e., $A \rightarrow B$ and $B \rightarrow C \Rightarrow A \rightarrow C$). $A \rightarrow B$ implies that information contained in object a of class A is permitted to flow to object b of class B. Similarly, $B \rightarrow C$ implies that information contained in object b is permitted to flow to object c in class C. This basically means that one can transfer information from object a to object c through a two-step process and thus information might as well be permitted to directly flow from objects of class A to the objects in class C. Otherwise, an inconsistency arises in the semantics of $\rightarrow$.
3. The relationship $\rightarrow$ is antisymmetric (i.e., $A \rightarrow B$ and $B \rightarrow A \Rightarrow A = B$). If information is allowed to flow from all objects of class A to objects in class B and similarly information is allowed to flow from all objects in class B to objects in class A then we are simply dealing with two redundant security classes. Thus, classes A and B are the same.

- *Second axiom of Denning's information flow* Assuming that SC is a finite set reflects a property of every practical system. One can always adopt finitely as many security classes as needed. Note that the number of objects associated with each security class can be unbounded.
- *Third axiom of Denning's information flow* This assumes the existence of a lower bound class $L \in SC$ which means $L \rightarrow A$ for all $A \in SC$. First, this property can be assumed without loss of generality. Second, it allows the modeling of publicly available information, which is a useful property in many information systems. Theoretically, this class can be represented by an empty set as the availability of public information in a system does not necessarily hold all the time.
- *Fourth axiom of Denning's information flow* To show that the class-joining operator $\oplus$ combines two security classes into their least upper

bound, Denning shows that the following two properties hold for all $A,B,C \in SC$:

1. $A \rightarrow A \oplus B$ and $A \oplus B$.
2. $A \rightarrow C$ and $B \rightarrow C \Rightarrow A \oplus B \rightarrow C$.

Property 1 is intuitively arrived at. If $A \oplus B$ is the security class resulting from information obtained collectively from objects in classes A and B, then information from objects in class A as well as from objects in class B is permitted to directly flow into objects from class $C = A \oplus B$.

Property 2 states that if information can flow individually from classes A and B to class C, then information combined from A and B should also be permitted to flow to C. For clarity, we refer to the example given by Denning [DENN76b]. Consider five objects containing numeric values a, b, c, c_1, and c_2, and corresponding to security classes A, B, C, C_1, and C_2, respectively. Assume that we have $A \rightarrow C$, $B \rightarrow C$, and $C = C_1 = C_2$. Now consider the following transformation affecting values a, b, c, c_1, and c_2:

$$c_1 := a;$$
$$c_2 := b;$$
$$c := c_1 * c_2.$$

Execution of this sequence of instructions assigns to c information derived from a and b, and thus $A \oplus B \rightarrow C$. Generalizing this fact for all types of transformations combining values from objects in classes A, B, and C, it follows that $A \oplus B$ yields the least upper bound of A and B.

The four axioms of Denning's information flow imply the existence of a greatest lower-bound operator over SC, denoted by $\otimes$. This, in turn, implies the existence of a unique upper bound for SC, denoted by H, therefore leading to the structure $<SC, \rightarrow, \oplus, \otimes>$ being a lattice. The greatest lower-bound operator, $\otimes$, is shown by Denning to be defined as

$$A \otimes B = \oplus L(A,B), \text{where } L(A,B) = \{C \mid C \rightarrow A \text{ and } C \rightarrow B\}.$$

Applying the $\oplus$ operator to $L(A,B)$ yields the greatest lower bound of A and B. As with the least upper-bound operator $\oplus$, the greatest lower-bound operator $\otimes$ is also operable on subsets of SC. It follows that for a subset $S = \{S_1, ..., S_n\} \subseteq SC$, $\otimes S = S_1 \otimes ... \otimes S_n$. Information contained in object a with a security class A can flow into an object whose security class is a member of the subset S if and only if $A \rightarrow S_1 \otimes ... \otimes S_n$.

The totality of the operator $\oplus$ means that it should be defined for every pair of security classes (i.e., $A \oplus B \in SC$ for every $A, B \in SC$). An information-flow policy in which the class-combining operator is not initially totally defined can incrementally add security classes as dictated by the $\oplus$ operator until it is totally defined. In fulfilling this theoretical aspect one might end up defining security classes that are not bound to any system resources.

Implications of the Lattice-Based Flow Model on Access Control

Access-control systems that are based on policies drawn from a lattice structure as in Denning's flow model are automatically *safe*. The safety property of such systems is due to the fact that an information flow taking place from, say, object a to object b cannot occur without the policy stating that $A \rightarrow B$ directly or indirectly through the transitivity of the $\rightarrow$ relationship. Considering that a lattice structure maps directly to a directed graph, the safety property of lattice-based access-control models reduces to deciding whether a directed path exists between any two nodes in the graph. Although both end nodes of this path would generally represent two passive objects, it can also be illustrated using active entities. In this case the origin node of the path represents the security class associated with an active entity such as an end user, a host system, or some programming agent. The end node represents the security class of an object in the system. This determination is a straightforward process. Furthermore, the transitive closure of the graph can be computed, and hence all access decisions become known a-priori. A process p is capable of transferring information from object a to object b if and only if $A \rightarrow P \rightarrow B$.

This flow property is further generalized to $A_1 \oplus \ldots \oplus A_n \rightarrow P \rightarrow B_1 \oplus \ldots \oplus B_m$ to indicate that process p can transfer information from objects $a_1,\ldots,a_n$ to any of the objects $b_1,\ldots,b_m$.

Examples of Lattice-Based Information-Flow Models

A basic lattice information-flow policy is one in which there are only two security classes one is system low denoted by L and the other is system high denoted by H. For instance, all resources with nonconfidential information are bound to L, while those containing confidential information are assigned to class H. In this case, $SC = \{L, H\}$. Besides reflexivity, the policy mainly consists of a single rule $L \rightarrow H$ as shown in Figure 4.2A, where the lattice is derived from a linear ordering of the security classes L and H. A generalization of this policy to a set of n linearly ordered classes is depicted in Figure 4.2B. A richer policy based on partial ordering is illustrated in Figure 4.2C. Figure 4.3 shows a policy derived from a poset of $\{A, B\}$.

Since the Cartesian product $\times$ of two lattices is a lattice, a richer lattice structure of an information-flow policy can be generated from the product of two lattices. An example of such structures is to combine one lattice from a linearly ordered set and one from a partially ordered set. In practice, the linear ordering is drawn from a set of authority levels referred to as *security levels*. An instance of such a linear ordering consists of

SC = {*unclassified*, *confidential*, *secret*, *TopSecret*}. The partial ordering is derived from the poset of a *set* of properties known as *categories*.

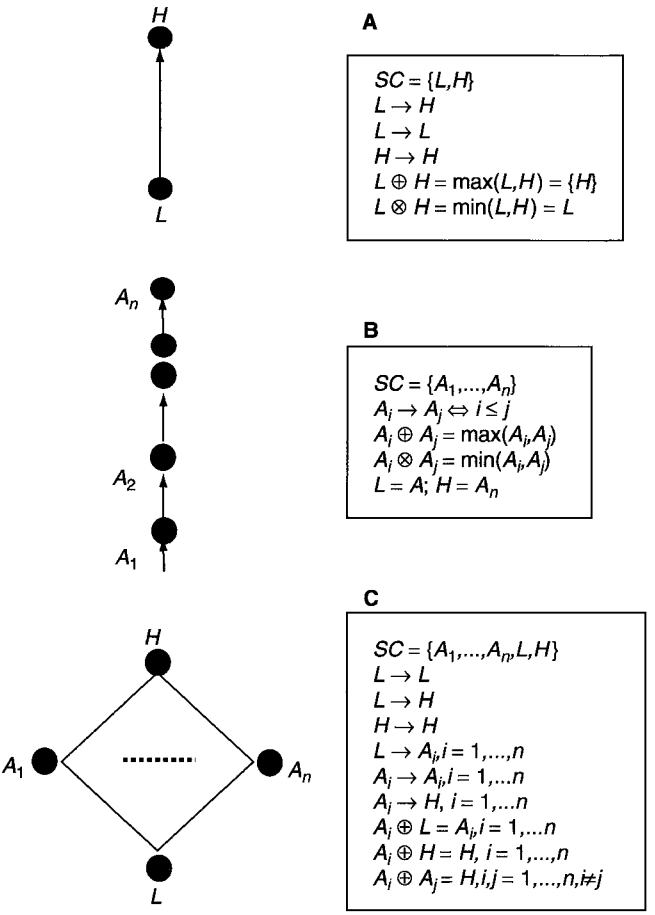

FIGURE 4.2 Basic examples of lattice-based information flow policies

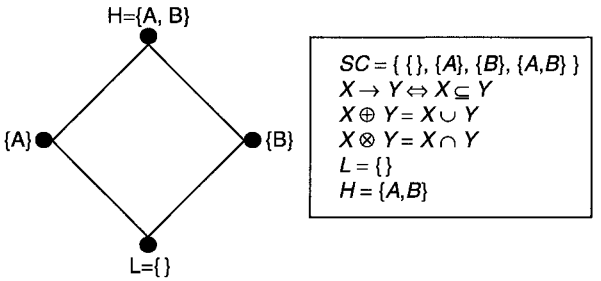

FIGURE 4.3 A simple lattice-based policy derived from poset of {A,B}

An example of categories is the set of departments of an organization in which a resource can be accessible. Security labels assigned to active system entities such as users and processes are said to be bound to security *clearances* and system resources are assigned security *labels*.

The derivation of a lattice structure for an information-flow model can be extended to a Cartesian product of n lattices. The resulting flow relationship $\rightarrow$ is determined by $\rightarrow \equiv \wedge \rightarrow_i, i = 1,...n$.

This means the flow relationship is computed as a logical AND over the flows in all of the participating lattices. The flow relationship therefore must hold in each of the lattices for it to hold in the lattice represented by their Cartesian product. For instance, when combining a linear ordering of security levels with a partial ordering as represented by the poset, the flow relationship is expressed as

$$A \geq B \Leftrightarrow B \rightarrow A, (B \rightarrow A) \Leftrightarrow A_{level} \geq B_{level} \text{ and } A_{categories} \supseteq B_{categories}.$$

The Bell-LaPadula Flow Model

Bell and Lapadula [BELL75, MCLE88] developed and formalized the concept of mandatory-access models, which falls in line with the information-flow model of Denning. It is worth noting that the model of Bell-Lapadula (BLP) preceded Denning's work on the information-flow model. The mandatory access-control policy as defined in BLP consists of assigning *security labels* (*classes*) to system subjects and objects. Labels assigned to objects are dubbed as *security classifications,* while those assigned to subjects are referred to as *security clearances*. BLP is stated in terms of two rules: the *simple security policy* and the **-property* (read as star property), both of which are mainly concerned with the flow of confidential information:

- Simple security rule This is also known as the *read-down* property. It states that information can be read only downward in the lattice structure representing the MAC policy. Subject s can read object o only if $S \geq O$ where S is the security label (class in Denning's formalism) of subject s, while O is the security label of object o. The security clearance of a subject has to dominate the security classification of an object so it can be read.
- *-property This rule is also known as the *write-up* policy. It states that subject s can write object o only if $O \geq S$. This prevents leaking confidential information in that a subject can write only objects whose security classifications dominate the security clearance of the subject. Writing objects takes place in an upward fashion within the lattice structure of the BLP policy, while reading is performed downward, as illustrated in Figure 4.4.

As has been indicated the flow model in BLP is motivated by the confidentiality of information. Consequently, the ability to read objects upward in

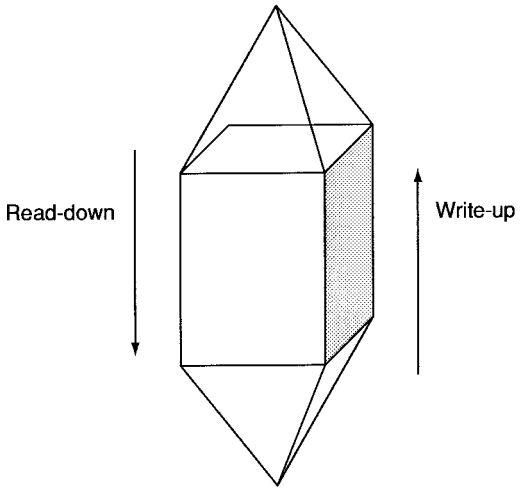

FIGURE 4.4 Information-flow direction in the BLP model as abstracted by a lattice structure

the lattice structure is not permitted. Similarly, the ability to write objects downward in the lattice structure is prohibited as both of these operations lead to transferring confidential information from higher-level entities to those having access to only lower-level objects.

The write-up property of BLP alone is not sufficient for preventing a subject from corrupting information at levels dominating those of the subject. Confidential information can be corrupted by subjects having lower security labels even when the read-down property prevents reading the information. To address this integrity problem, MAC policies have adopted a modified *-property that allows subject s to write object o only if the subject and the object are both bound to the same security class (i.e., $S = O$).

The integrity issue associated with the write-up property can in fact be addressed by the second component of the BLP model, which enforces a discretionary policy of resource-access control. In BLP the dominance relationship as stated by the MAC policy is augmented with a discretionary-access policy. An access decision therefore depends on both policies, MAC and DAC, being enforced at the same time. With this approach, corruption of confidential information by processes at lower security classes is prevented by specifically exposing resources that are intended to be receptacles of information from lower processes and disallowing access to the ones that contain confidential information through proper DAC policies. Similarly, the read-down property may also be controlled in this manner, although generally enforcing DAC controls around the write-up property is the main concern of many MAC policies.

The Biba Model

As has been noted, the goal of the BLP model is to prevent downgrading confidential information. The Biba model, on the other hand, is concerned

with the integrity of information [BIBA77]. This model follows along the same ideas of the BLP model and as such does not present a fundamental departure from the concepts introduced by BLP. The underlying concept in Biba is that security classes are organized along a lattice structure in which each class corresponds to some integrity level with the highest integrity at the top of the structure and the lowest at the bottom. Information is allowed to flow from high-integrity objects to low-integrity objects only. In a similar way to BLP, Biba states its information flow policy using two rules: the *simple-integrity property* and the *integrity *-property*:

- *Simple-integrity property* This property states that subject s can read object o only if the security class of o dominates that of s (i.e., $O \geq S$).
- *Integrity *-property* This property states that subject s can write object o only if the security class of s dominates that of o (i.e., $S \geq O$).

Recall that a security class in Biba corresponds to an integrity label. A curious aspect of the Biba properties is that they are duals of their counterpart in BLP. For instance, while the policy in BLP is about read-down of information, the simple-integrity property of Biba states a read-up of information. Similarly, the integrity *-property of Biba states a write-down type of information flow as opposed to the write-up of the *-property in BLP.

Comparing Information Flow in BLP and Biba Models

The direction in which information flows in the BLP and the Biba models is driven by the nature of protections sought in each model. The BLP is motivated by confidentiality of information, and hence information in objects at higher levels is not allowed for read access by lower-level processes. Similarly, information at lower levels is allowed to flow to objects from higher security classes in the lattice structure. The write-up property of BLP represents an interesting aspect of information flow. It can be used to upgrade the classification of information from the bottom of the lattice all the way to its top as illustrated in Figure 4.5A. Once this information is copied to higher-level objects, there is no rule that enforces its deletion from lower-level objects where the information originates so that it can no longer be read by processes at those levels. Recall that the BLP as well as the Biba properties allow a process to simultaneously read and write objects at the same level in the lattice.

A process p_1 as depicted in Figure 4.5A reads object o_1 situated at its immediate lower level, writes it to object o_2 at the same level as p_1, then writes it to object o_3 located immediately above the level of p_1. Similarly, p_1 may also read o_1 and write it directly to o_3. Thus the flow of information between a lower level and any higher level may be achieved through a sequence of operations or simply in by a single sequence of read and write operations.

The direction of information flow in the Biba model is the opposite of that in the BLP model. As illustrated in Figure 4.5B information is allowed to flow from the top of the lattice all the way to its bottom in accordance with

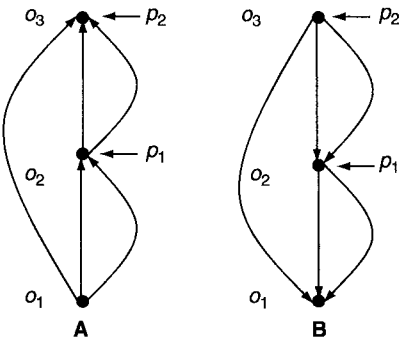

FIGURE 4.5 Scenarios of information flow in the BLP and the Biba models

the Biba properties. Although this flow does not imply modifying the security classes of objects involved, it somehow represents a downgrade of information as it yields a transfer of information from higher to lower security classes.

A curious reader may ask the question of why we need to enforce the read-up property in the Biba model as it does not seem to interfere with the integrity goal of Biba. Let us assume that in addition to the read-up capability, processes are also able to read-down objects in the lattice structure of a Biba integrity policy. As shown in Figure 4.6, process p_1 reads down an object o and writes it to object o_1 located at the same security label as p_1 (read and write at the same level are permissible due to the equality in the dominance relationship $\geq$). Now an upper level process p_2 reads down o_1 and writes it to object o_2 at the same level as that of p_2. Performing these steps in a bottom-up fashion along the lattice structure results in the flow of information upward, therefore conflicting with the intent of the Biba model.

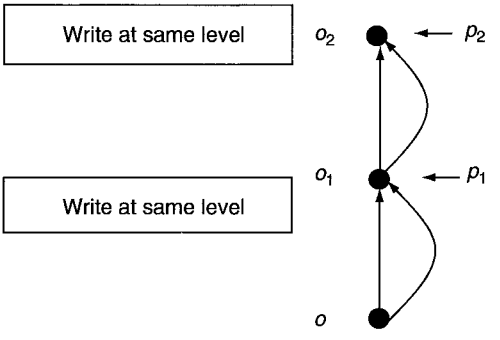

FIGURE 4.6 The need for read-up only in the Biba integrity model

Implementation Considerations for the BLP and the Biba Models

One implementation aspect that is worthy of mention for the BLP and the Biba models is the need to provide safety of concurrency. At any level in the BLP or the Biba policy lattice, objects have to be protected from concurrent writes by processes of that level. In the BLP model, objects situated at level l need to be further protected against concurrent writes by processes at levels $\leq l$ (Figure 4.7A).

It is also desirable to prevent against a simultaneous read and write of the same object. In the Biba model, objects situated at level l should be protected against concurrent writes by processes at levels $\geq l$ as illustrated in Figure 4.7B. Like in the BLP case, it is also desirable to prevent against simultaneous read and write of the same object.

Combining the BLP and the Biba Models

Protected entities of a computing system (resources, subjects, and programming agents or processes) can be subjected simultaneously to the BLP and Biba policies. We distinguish two ways in which such coexistence may take shape. In the first scenario we draw the security classes for the combined confidentiality and integrity lattices from a single set SC in which every security

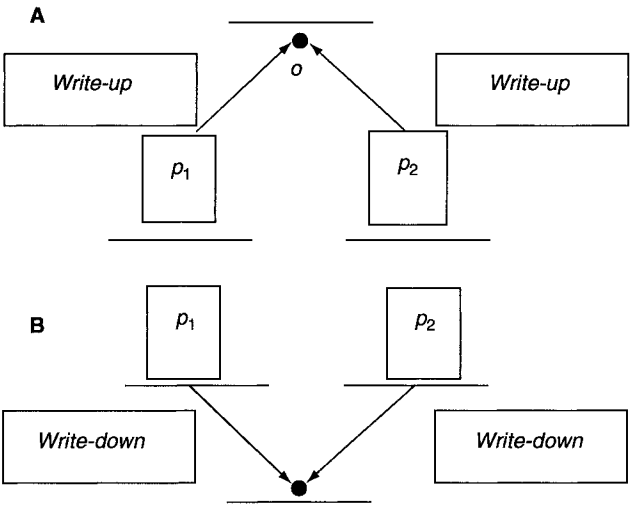

FIGURE 4.7 Synchronization requirement for concurrent reads and writes in the BLP and Biba models

class applies as a confidentiality and an integrity label simultaneously. The write-up in BLP requires the security class of the writing subject to be dominated by that of the receptacle object, while the write-up property of Biba requires the opposite. Hence writing an object in this scenario is confined to processes that are all at the same level as that of the object to be written. This amounts to the trivial isolationist policy where no information flows across security levels of a lattice. From the standpoint of information flow analysis, this model is equivalent to using a single security class. The isolated classes scenario is depicted in Figure 4.8.

The second and a more useful scenario of combining the BLP and the Biba models results from adopting independent confidentiality and integrity classes as shown by Sandhu [SAND93]. A composite model as such is the product of two lattices, which is in turn a lattice. Let $C = \{c_1,...,c_n\}$ be a lattice of confidentiality corresponding to the BLP model, and let $I = \{i_1,...,i_m\}$ be a lattice of integrity representing a policy based on the Biba model. Let α be a function that maps a system entity (subject or object) onto its confidentiality class (label), and let β be the function that maps an entity onto its integrity class. The composite BLP and Biba lattice is defined by the following constraints:

- Subject s can read object o only if $\alpha(s) \geq \alpha(o)$ and $\beta(s) \leq \beta(o)$.
- Subject s can write object o only if $\alpha(s) \leq \alpha(o)$ and $\beta(s) \leq \beta(o)$.

As has been noted, the composite model is the product of two lattices which reduces to one lattice. Figure 4.9 illustrates an instance of this lattice for $C = \{\alpha_L, \alpha_H\}$ with $\alpha_H \geq \alpha_L$ and $I = \{\beta_L, \beta_H\}$ with $\beta_H \geq \beta_L$, where L and H denote system Low and High, respectively. Note that while information in the BLP and Biba models flows in opposite directions, in the combined lattice (Figure 4.9) information flows upward.

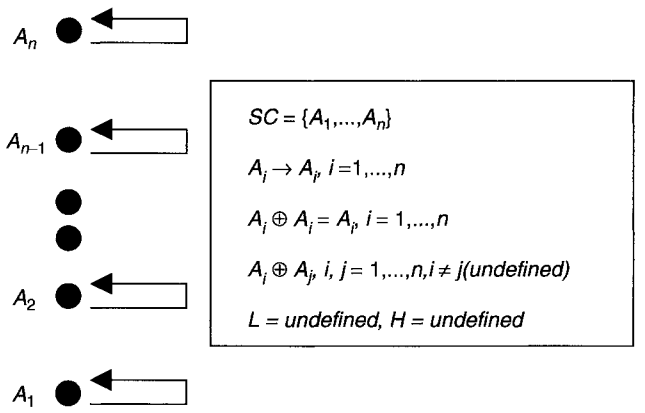

FIGURE 4.8 Combining BLP and the Biba models: The case of security classes that are used for both confidentiality and integrity

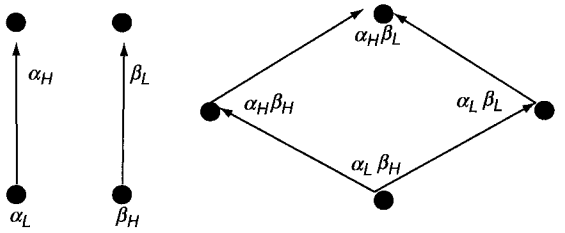

FIGURE 4.9 An example of combining BLP and the Biba models in the case of independent confidentiality and integrity classes

Figure 4.10 illustrates an access-control matrix representing the access policy of the product lattice of Figure 4.9. Rows of this matrix represent subjects, and the columns correspond to resources. Each row of the table specifies exactly the type of access a subject with a given label can have to a resource on the column. For example, a subject with label $\alpha_H \beta_H$ can read (r) information contained in resources with label $\alpha_L \beta_H$, and write (w) objects with labels $\alpha_H \beta_L$ but cannot (ϕ) read or write resources with labels $\alpha_L \beta_L$. The diagonal of this matrix represents access modes that subjects can have to the resources that are associated with the same levels as those of the subjects. Read and write accesses are thus shown along the diagonal.

One characterizing aspect of the composite BLP and Biba model is the fact that if information in the confidentiality-based model flows from one class (say, C_i) to another class C_j, then information in the composite model flows from classes $C_i I_k$ to classes $C_j I_k$ for all $k = 1,...,m$ (m being the cardinality of set I). Similarly, if information separately in the integrity model flows from

	$\alpha_L\beta_L$	$\alpha_L\beta_H$	$\alpha_H\beta_L$	$\alpha_H\beta_H$
$\alpha_L\beta_L$	rw	r	w	ϕ
$\alpha_L\beta_H$	w	rw	w	w
$\alpha_H\beta_L$	r	r	rw	r
$\alpha_H\beta_H$	ϕ	r	w	rw

FIGURE 4.10 An access-control table corresponding to the subjects and objects of the example of Figure 4.9

one class (say, I_j) to another class I_k, it follows that information in the resulting composite model flows from classes I_j C_i to classes I_k C_i for all $i = 1,...,n$ (n being the cardinality of set C). These properties are an immediate result of the fact that in either of the models information is always allowed to flow from and to the same security class.

On the Mandatory-Access-Control Paradigm

As has been noted, the development of the mandatory-access-control model was motivated mainly by the control policies found in military environments, specifically, in the United States Department of Defense (DoD). Within the DoD an information security policy assigns each system entity a linearly ordered classification level L and a set of categories C. The categories generally form a partial ordering along the poset relationship. The hierarchy of entities and resources as imposed by military policies is certainly amenable to the adoption of mandatory-access controls. In the commercial world, however, this is not generally the case, even when the categories are designed to reflect the organizational structure of an enterprise.

The authoritative policies of mandatory controls are inflexible and not amenable to sharing resources as warranted by the needs for information sharing. MAC policies are static in nature. They cannot be changed dynamically and without the intervention of an administrative authority whose immediate availability can be an issue. Resources of the same security class are undistinguishable with respect to the access controls applied at their level. For instance, all of the resources assigned the same confidentiality label in the BLP model can be read by every subject with a security label that dominates those resources. MAC policies do not support the concept of resource ownership and hence the inability to discern access rights to the resource in a discretionary fashion. Identification of resource ownership is a fundamental aspect of building access-control systems in modern commercial operating environments. With all these issues, Lipner [LIPN82] addressed optimum ways in which mandatory controls can be applied in the commercial nonmilitary world. He gave a detailed example in which confidentiality and integrity labels are simultaneously used as in the composite BLP and Biba models to achieve commercial uses.

Finally, it is worth noting that despite of the fact that BLP and Biba models are based on the confidentiality and integrity of information, respectively, they can be applied to any other types of information access. The semantics of access rights in the lattice-based models therefore can take various forms.

The Chinese-Wall Policy

The Chinese-wall policy (CWP) was developed by Brewer and Nash [BREW89] as an instance of lattice-based security models with applications

in the commercial world. The intent of CWP is to enforce a conflict of interest policy in which a single user is prevented from having to simultaneously access information that represents a conflict of interest. Specifically, CWP was formulated to address a situation in which a financial institution provides market analysis as part of its consulting services to other businesses. Each analyst must not be able to advise a particular institution when he or she has knowledge of business information about a competitor of that institution. The analyst, however, is capable of advising any companies that are not in competition with each other. Thus, every subject that is affiliated with this consulting service must be confined to accessing information on businesses that are not competing with one another. For example, information about bank B should not be accessible to a subject that already has access to information about bank A. Unlike in BLP, where access to information is based on a static relationships between subjects and objects, in CWP access is constrained by what information the subject already has access to.

The elements of CWP are illustrated in Figure 4.11. A company maintains information about other businesses that is hierarchically divided along a set of conflict of interest classes. Within each class the company groups all information about a particular business in a dataset. In turn, each dataset consists of a number of individual objects containing data related to that business.

In a way similar to the BLP model, CWP is stated in terms of its own formulation of the simple security and the *-Property rules. It is also worth noting that Sandhu developed a scheme in which he shows how CWP is mapped to a lattice-based access-control model [SAND92a, SAND93].

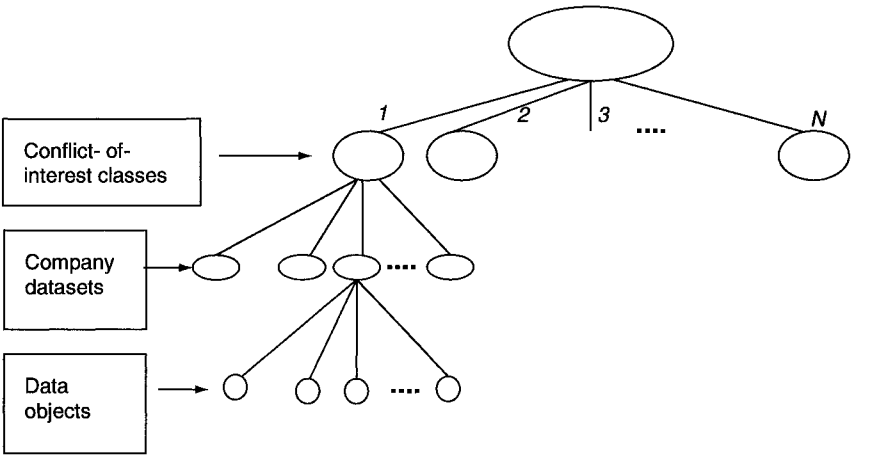

FIGURE 4.11 Dividing information along a Chinese-wall policy

Simple Security

This represents the basis of the CWP enforcing the fact that a user is allowed only access to information that is not in conflict with any information already accessible to that user. Access by a subject to an object is therefore granted only if

- The object is in the same company dataset that is already accessed by that subject (i.e., the object is within the wall), or
- The object belongs to an entirely different conflict of interest class.

As a result, Brewer and Nash establish the following theorems:

Theorem 1: Once a subject has accessed an object the only other objects accessible by that subject reside within the same company dataset or within a different conflict of interest class.

Theorem 2: A subject can at most have access to one company dataset in each conflict of interest.

Theorem 3: If for some conflict-of-interest class X there are X_Y company datasets, then the minimum number of subjects that will allow every object to be accessed by at least one subject is X_Y.

*-Property

This rule states that write access is permitted only if

- Access is permitted by the simple security rule, and
- Any object that is in a different company dataset with respect to the one for which write access is requested cannot be read.

The *-Property is used to prevent the writing of information that results in violating the simple security rule. An example of such scenario is the case of two subjects s_1 and s_2 that have access to three companies as follows: s_1 has access to bank 1 and computer company 1, while s_2 has access to bank 1 and computer company 2. If s_1 reads information about computer company 1 and writes it to objects containing information about bank-1, then s_2 can read computer company 1 information and thus yield a conflict of interest.