

# The Algebra of Security

*John McLean*

Computer Science and Systems Branch  
Naval Research Laboratory  
Washington, D.C. 20375

## ABSTRACT

We develop a general framework in which various mandatory access control security models that allow changes in security levels can be formalized. These models form a Boolean Algebra. We expand the framework to include models that allow n-person rules necessary for discretionary access controls in an industrial security setting. The resulting framework is a distributive lattice.

## 1. Introduction

General security models provide a formal framework that supports theoretical reasoning about the concept of security embodied in a wide class of systems. Because of their general nature, however, they have little to say about changing security levels [10]. This is understandable since a general policy on changing security levels does not seem forthcoming. Different systems require different tradeoffs between security and flexibility. Application-specific models can say more about a system's security policy, but they lack an overall theory. Model builders repeatedly start from scratch when developing a model for their particular needs. What clearly is needed is a single framework within which application-specific models can be developed and compared. Such a framework would give us the ability to perform general reasoning about models which are specific enough to capture real-world security policies.

This paper presents a general framework for application-specific models and shows that the models that instantiate this framework form a Boolean Algebra. The framework is extended to allow models that contain n-person rules. Such rules form the heart of discretionary security as practiced in industry [2]. The resulting framework is a distributive lattice. Both frameworks support the construction of new models from old ones and the comparison of models, allowing model developers to build on the work of their predecessors.

## 2. A Framework for MAC Models

In this section we develop a framework for Mandatory Access Control (MAC) models that allow changes in security levels. The models of this framework will be based on the access-matrix approach epitomized by the model of Bell and LaPadula (BLP) [1]. We take this approach though we realize that such models have disadvantages over models based on information flow [5] or noninterference [6]. In the first place, specification methodologies that mention the internal states of a state machine make it hard to glean behavioral essentials from specification artifact [9,12]. Access-matrix models, in particular, suggest the implementation concept of a reference monitor. Further, their rather course-grained approach causes access-matrix models to rule out accesses that do not violate security in any intuitive sense. They fail to make the distinction between having the ability to view a file and actually doing it. Hence, they fail to distinguish between a confidential user requesting read access to a secret file to copy it to another secret file and the same user requesting read access to the file to view it [7]. Finally, access-matrix models do not prohibit all non-secure flows of information [11].

Nevertheless, those who build secure systems regard all these disadvantages as being overridden by the fact that, in their opinion, access-matrix models can be implemented in a way that makes it relatively easy to verify that they are being enforced correctly. It is easier to show that a program has not violated access restrictions than to show that it has violated restrictions involving potential modification and contributing factors as defined, *e. g.*, by the SMMS model [8]. Once we have determined which types of access each system operation involves, static examination of program text shows when a file is being accessed in a particular way.<sup>1</sup> This is not true for information flow; in fact, detecting all and only nonsecure information flow is undecidable [3].

A framework is determined by a finite set of subjects,

1. Of course, making this determination is not trivial.

$S$ , a finite set of objects,  $O$ , a finite lattice of security levels,  $L$ , and a finite set of access modes,  $A$ . Intuitively,  $S$  consists of system users and perhaps, programs;  $O$  consists of system files (possibly including programs) and possibly, terminals, printers, etc.;  $L$  is the standard set of security levels under their usual ordering; and  $A$  is  $\{read, write, execute\}$ , the set of modes in which an element of  $S$  can access an element of  $O$ . Subjects who have *read* access to an object can read it; those who have *write* access can modify it; and those who have *execute* access can execute it. To edit an object requires both *read* and *write* access. Given a framework, a model is determined by an ordered pair  $C=(c_s, c_o)$  such that  $c_s$  is a total function from  $S$  to  $P(S)$  and  $c_o$  is a total function from  $O$  to  $P(S)$ . Intuitively,  $c_s(s)$  and  $c_o(o)$  are the set of subjects who can change the security levels of  $s$  and  $o$ , respectively.

A *system* is an abstract state machine constructed from the elements of a model. A system state  $v=(b, f)$  is an element of  $V \subseteq (B \times F)$ , where

- $B$  is  $P(S \times O \times A)$ , the set of all possible current access sets,<sup>2</sup> and
- $F$  is  $L^S \times L^O$ , the set of all possible ordered pairs of functions  $(f_s, f_o)$  such that  $f_s$  gives the security level (clearance) associated with each subject, and  $f_o$  gives the security level (classification) associated with each object.

The set of system requests (to change an element of a system state) is denoted by  $R$ . The system's state transition function,  $T: S \times R \times V \rightarrow V$  moves the system from one state to another: a subject  $s$  issues a request  $r$  in state  $v$  that moves the system from state  $v$  to a new state. The system  $\Sigma(V, R, T, v_{init})$  is the state machine consisting of states  $V$ , requests  $R$ , and transition function  $T$  that starts in initial state  $v_{init} \in V$ . A state  $v$  is *reachable* in a system  $\Sigma(V, R, T, v_{init})$  if and only if  $v=v_{init}$  or there is a sequence  $\langle (s_0, r_0, v_0), \dots, (s_n, r_n, v_n) \rangle$  such that  $v_0=v_{init}$ ,  $T(s_n, r_n, v_n)=v$ , and for all  $0 \leq i < n$ ,  $T(s_i, r_i, v_i)=v_{i+1}$ .

Given this abstract machine, we define a security policy on the machine by appropriately restricting the machine's reachable states and state transition function. A state  $s=(b, f)$  is *simple secure* if for each element of  $b$  of the form  $(x, y, read)$ ,  $f_s(x) \geq f_o(y)$ . A state  $s=(b, f)$  is *\*-secure* if for any subject  $w$  and objects  $x, y$ , if  $(w, x, read) \in b$  and  $(w, y, write) \in b$ , then  $f_o(y) \geq f_o(x)$ .<sup>3</sup> A

transition function is *transition secure* if each transition  $T(s, r, v)=v^*$ , where  $v=(b, f)$  and  $v^*=(b^*, f^*)$ , is such that (1) for all  $x \in S$  if  $f_s^*(x) \neq f_s(x)$ , then  $s \in c_s(x)$ , and (2) for all  $y \in O$  if  $f_o^*(y) \neq f_o(y)$ , then  $s \in c_o(y)$ .<sup>4</sup> A system is *secure* only if (1) all its reachable states are *simple secure* and *\*-secure*, and (2) its transition function is *transition secure*. All models in a framework share this security policy, though the consequences of the policy differ from model to model depending on the particular model's function pair  $C$ .

A *model* is the set of systems that share  $S, O, L, A$ , and  $C$  and do not violate our security policy. A *framework* is the set of models that share  $S, O, L$ , and  $A$ . Note that our security policy gives necessary, but not sufficient, conditions for a system to be *secure*. Hence, different models within a framework do not contradict one another. This would not be the case if our policy gave sufficient conditions for *security* as well.

A given security model corresponds to a model in a framework if and only if the set of systems that is nonsecure by the criteria of the former is the set of systems that is nonsecure by the criteria of the latter. Given this sense of correspondence and sets  $S, O, L$ , and  $A$ , BLP corresponds to that element of the framework for which  $c_s(x) = c_o(y) = S$  for all  $x \in S$  and  $y \in O$ . It is the least stringent of the framework's models. The most stringent is BLP supplemented by tranquility, the restriction that security levels cannot change. Such a model is embodied in the SeaView project [4] and corresponds to one where  $ran(c_s) = ran(c_o) = \{\emptyset\}$ . An intermediate position is held by the SMMS model [8] where a subject  $ss_o \in S$  is designated system security officer, and for all  $x \in S$  and  $y \in O$ ,  $c_s(x) = c_o(y) = \{ss_o\}$ .

The three models are intuitively ordered, and more generally, there is a partial ordering of models within a framework. Given any two models  $M1$  and  $M2$ , we say that  $M1 \leq M2$  if and only if there are one-to-one functions  $I_s$  from  $S$  onto  $S$  and  $I_o$  from  $O$  onto  $O$  such that for all  $x, y \in S$  and  $z \in O$ ,  $c1_s(x) \subseteq \{I_s^{-1}(y) : y \in c2_s(I_s(x))\}$  and  $c1_o(z) \subseteq \{I_o^{-1}(x) : x \in c2_o(I_o(z))\}$ . We say that  $M1 \equiv M2$  if and only if  $c1_s(x) = \{I_s^{-1}(y) : y \in c2_s(I_s(x))\}$  and  $c1_o(z) = \{I_o^{-1}(x) : x \in c2_o(I_o(z))\}$ .

It is straightforward to show that  $\leq$  as defined on models is transitive and that  $\equiv$  is an equivalence relation.

2. We use  $P(\phi)$  to denote the power set of  $\phi$ .

3. Though BLP's formulation of the *\*-property* in terms of a current security level,  $f_c$ , may seem more stringent than ours, our formulation is equivalent to BLP's since there is nothing in the latter or its Multics interpretation that prohibits a subject from changing its own current security level [1]. Both formulations require the assumption that processes that give up *read* access to an object are "sanitized" before gaining *write* access to an object at a lower security level [10]. Our version makes the formulation of n-person rules a bit easier. See below.

4. Two points are worth making about transition restrictions. First, our policy shares with BLP *implicit* transition restrictions, e. g., that an object does not change from state to state unless someone has *write* access to it. Second, we could state our transition restriction as a state restriction, e. g., as the restriction that  $s$  can have *write* access to the security level of a subject  $x$  (object  $y$ ) only if  $s \in c_s(x)$  ( $s \in c_o(y)$ ). However, such a formulation brings with it the disadvantages of the access-matrix approach without buying us much, since detecting changes in security levels is straightforward.

In fact, as the following theorem and its corollary show,  $\equiv$  is a congruence relation for  $\leq$  on the set of models in a framework. This justifies our ignoring differences that can be eliminated by renaming when comparing models, and hence, for example, assuming that  $M1 \equiv M2$  if and only if for all  $x \in S$  and  $y \in O$ ,  $c1_s(x) = c2_s(x)$  and  $c1_o(y) = c2_o(y)$ .

**Theorem:**  $M1 \equiv M2$  if and only if  $M1 \leq M2$  and  $M2 \leq M1$ .

**Proof:** We must prove that there are one-to-one functions  $I1_s$  and  $I2_s$  from  $S$  onto  $S$  and  $I1_o$  and  $I2_o$  from  $O$  onto  $O$  such that for all  $x, y \in S$  and  $z \in O$ ,  $c1_s(x) \subseteq \{I1_s^{-1}(y) : y \in c2_s(I1_s(x))\}$ ,  $c2_s(x) \subseteq \{I2_s^{-1}(y) : y \in c1_s(I2_s(x))\}$ ,  $c1_o(z) \subseteq \{I1_o^{-1}(x) : x \in c2_o(I1_o(z))\}$ , and  $c2_o(z) \subseteq \{I2_o^{-1}(x) : x \in c1_o(I2_o(z))\}$  if and only if there are one-to-one functions  $I3_s$  from  $S$  onto  $S$  and  $I3_o$  from  $O$  onto  $O$  such that for all  $x, y \in S$  and  $z \in O$ ,  $c1_s(x) = \{I3_s^{-1}(y) : y \in c2_s(I3_s(x))\}$  and  $c1_o(z) = \{I3_o^{-1}(x) : x \in c2_o(I3_o(z))\}$ . We claim for any finite set  $S$  with binary relations  $R1$  and  $R2$  on  $S$  and one-to-one functions  $I_s$  and  $I_s^*$  from  $S$  onto  $S$ , that for all  $x, y \in S$ ,  $R1(x, y) \rightarrow R2(I_s(x), I_s(y))$  and  $R2(x, y) \rightarrow R1(I_s^*(x), I_s^*(y))$  if and only if  $R1(x, y) \leftrightarrow R2(I_s(x), I_s(y))$ . We also claim for  $S$ ,  $I_s$ , and  $I_s^*$  as above, and any finite set  $O$  with binary relations  $R3$  and  $R4$  on  $S \times O$  and one-to-one functions  $I_o$  and  $I_o^*$  from  $O$  onto  $O$ , that for any  $x \in S$  and  $z \in O$ ,  $R3(x, z) \rightarrow R4(I_s(x), I_o(z))$  and  $R4(x, z) \rightarrow R3(I_s^*(x), I_o^*(z))$  if and only if  $R3(x, z) \leftrightarrow R4(I_s(x), I_o(z))$ . Our theorem follows *a fortiori* by letting  $R1(x, y)$ ,  $R2(x, y)$ ,  $R3(x, z)$  and  $R4(x, z)$  be the relations  $x \in c1_s(y)$ ,  $x \in c2_s(y)$ ,  $x \in c1_o(z)$ , and  $x \in c2_o(z)$ , respectively.

We prove only the first claim. The second claim follows by an analogous argument. The "if" part of the claim is obvious since we can let  $I_s$  and  $I_s^*$  be  $I_s^{-1}$ . To prove the "only if" part, note that since  $I_s$  is one-to-one and  $R1(x, y) \rightarrow R2(I_s(x), I_s(y))$ , the cardinality of the relation  $R1$  is less than or equal to the cardinality of the relation  $R2$ . Similarly, since  $I_s^*$  is one-to-one and  $R2(x, y) \rightarrow R1(I_s^*(x), I_s^*(y))$ , the cardinality of the relation  $R2$  is less than or equal to the cardinality of the relation  $R1$ . Hence,  $R1$  and  $R2$  are the same size, and since  $S$  is finite, the mapping  $I_s \times I_s$  from  $R1$  to  $R2$  is both one-to-one and onto. This implies that  $(x, y) \in R1$  if and only if  $(I_s(x), I_s(y)) \in R2$ , i. e., that  $R1(x, y) \leftrightarrow R2(I_s(x), I_s(y))$ , which is the desired result.  $\square$

**Corollary:**  $\equiv$  is a congruence relation for  $\leq$  i. e. if  $M1 \equiv M2$ ,  $M3 \equiv M4$ , and  $M1 \leq M3$ , then  $M2 \leq M4$ .

**Proof:** By our theorem,  $M1 \equiv M2$  implies that  $M2 \leq M1$  and  $M3 \equiv M4$  implies that  $M3 \leq M4$ . By the transitivity of  $\leq$   $M1 \leq M3$  implies that  $M2 \leq M4$ .  $\square$

Not only can we compare models within a framework, we can construct new models from given models within the

framework. For example, if  $M1$  embodies the principle that certain subjects control certain objects and such subjects can change the classification of objects in their control, and  $M2$  assumes the existence of a system security officer  $sso \in S$  who is the only subject allowed to change the classification of objects, we may want to form a model  $M3$  where a subject can change an object's classification if the subject controls the object or is  $sso$ . We define such a model as  $M3 = M1 \cup M2$ . In this case  $M3$  is  $M1$  except that for all  $x \in S$ ,  $c3_s(x) = c1_s(x) \cup c2_s(x)$ , and for all  $y \in O$ ,  $c3_o(y) = c1_o(y) \cup c2_o(y)$ .  $M1 \cap M2$  is defined analogously and is the model that meets the restrictions on changing security levels implied by both  $M1$  and  $M2$  together. Hence,  $M1 \cap M2$  allows only  $sso$  to change an object's classification and only if  $sso$  controls the object.  $M'$  is the same as  $M$  except that for all  $x \in S$ ,  $c_s(x) = S - c_s(x)$ , and for all  $y \in O$ ,  $c_o(y) = S - c_o(y)$ . Hence,  $M2'$  allows any user except  $sso$  to change the security level of an object. If we assume that  $sso$  monitors such changes, such a policy enforces separation of powers. These operations for model construction form a Boolean Algebra.

**Theorem:** Given a framework the set of models that are instances of the framework form a Boolean Algebra under the operations  $\cap$ ,  $\cup$ , and  $'$ .

**Proof:** Assume that  $S$  and  $O$  are ordered where  $s_i$  and  $o_i$  represent the  $i$ th element in the ordering of  $S$  and  $O$ , respectively. Each model is uniquely determined by its ordered pair  $C$ . Each  $c_s$  can be represented by a tuple  $\langle n_1, n_2, \dots, n_m \rangle$  where  $n_i$  is  $c_s(s_i)$ . Since for each  $c_s$ ,  $ran(c_s) = P(S)$ , for any  $i$  the set  $\{x : x = n_i \text{ for some } c_s\}$  forms a Boolean Algebra under  $\cap$ ,  $\cup$ , and  $'$ . Hence, the set of  $c_s$ 's form a Boolean Algebra, as well, viz. the product algebra of the algebras for the individual  $n_i$ 's. Similar observations apply to the set of  $c_o$ 's. Hence, the set of  $C$ 's (and models) is the product algebra of the algebras of the component  $c_s$ 's and  $c_o$ 's, and we are done.  $\square$

Earlier, we considered the partial order  $\leq$  on models and the models that correspond to BLP and to BLP with tranquility. It is easy to show that if all models follow the same naming convention for subjects and objects, then  $\leq$  is the ordering relation induced by the Boolean Algebra, i. e., that  $M \leq M2$  if and only if  $M1 \cap M2 \equiv M1$ , and BLP and BLP with tranquility are the top and bottom elements, respectively, of the Boolean Algebra. Hence, as we move toward the top of the algebra, the models become less stringent, and in general,  $M1 \leq M2$  if any system that satisfies  $M1$  satisfies  $M2$  as well.

Given that each framework is an algebra, it is only natural to consider an extension to a given framework in the sense that the extension may have more subjects or objects than the original. It is more natural to consider systems.

Assume we have sets  $L$  and  $A$ . Given the sets  $S1, S2, O1, O2, C1$ , and  $C2$ , the system  $\Sigma1(V1, R1, T1, v1_{init})$  is an *extension* of  $\Sigma2(V2, R2, T2, v2_{init})$  if and only if  $S2 \subseteq S1$ ,  $O2 \subseteq O1$ ,  $R1$  restricted to  $S2$  and  $O2$  is identical to  $R2$ ,  $f1_s$  restricted to  $S2$  is identical to  $f2_s$ ,  $f1_o$  restricted to  $O2$  is identical to  $f2_o$ ,  $T1$  restricted to the domain of  $T2$  is identical to  $T2$ ,  $b1_{init}$  restricted to  $S2$  and  $O2$  is identical to  $b2_{init}$ ,  $f1_{s_{init}}$  restricted to  $S2$  is identical to  $f2_{s_{init}}$ , and  $f1_{o_{init}}$  restricted to  $O2$  is identical to  $f2_{o_{init}}$ . If  $\Sigma1$  extends  $\Sigma2$ , we also say that  $\Sigma2$  is a *subsystem* of  $\Sigma1$ . This notion allows us to include systems that create and delete objects and add and remove users. For example, if we want to create an object at a certain security level while in state  $v_n$  of system  $\Sigma(V, R, T, v_{init})$ , the system from  $v_{n+1}$  onwards is the extension of system  $\Sigma(V, R, T, v_n)$  that consists of adding the new object to  $O$  and extending the functions  $f_s$ ,  $f_o$ ,  $c_s$ , and  $c_o$  to reflect the corresponding properties of the new object.

### 3. A Framework for N-Person Rules

Since this way of modeling the ability to change security levels places all the theory of Boolean Algebra at our disposal for comparing and constructing models, we should be reluctant to forsake it. Nevertheless, it has the limitation that it does not provide us the ability to formulate n-person rules on our system. Such a rule may be, for example, that the classification of an object can only be changed with the approval of both the owner of the object and the system security officer. This requirement may be cashed out on a lower level by stating that the classification of an object can be changed only if the owner of the object executes a program that requests a change and the system security officer then executes a program that grants the request. On a more abstract level, however, it is worthwhile having the ability to represent the concept of an action being executed by several people jointly.

To this end we consider frameworks whose subject set has a particular structure. We replace our set of subjects,  $S$ , by  $P(S) - \{\emptyset\}$ , the set of nonempty subsets of  $S$ , which we denote by  $\mathbf{S}$  and whose elements we denote by  $\mathbf{s}$ .<sup>5</sup> In our definition of system we replace  $B$  by  $P(\mathbf{S} \times O \times A)$ , and  $f_s$  by a function from  $\mathbf{S}$  to  $L$  such that for any  $\mathbf{s} \in \mathbf{S}$ ,  $f_s(\mathbf{s})$  is the greatest lower bound (in  $L$ ) of  $\{f_s(\{s\}) : s \in \mathbf{s}\}$ . If, for example,  $\{(\{w\}, o, write), (\{x\}, o, write), (\{y, z\}, o, write)\} \subseteq b$ , then  $w$  and  $x$  each have *write* access to  $o$ , and  $y$  and  $z$  have joint *write* access to  $o$ . The third access may signify that  $y$  and  $z$  can change  $o$ , but only if they both concur on the change.

Given these changes to the framework presented in the previous section, our definition of *simple security* remains

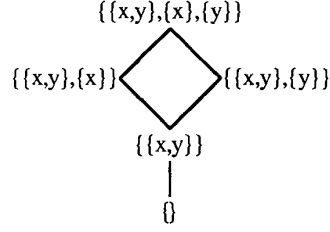
5. What follows can be applied to the set of objects in an analogous manner if we wish to capture the notion of being able to operate on an object only conjointly with operations on another object, as for example, in double-entry bookkeeping.

unchanged. Our definition of the *\*-property*, however, requires some slight modification. We now say that a state is *\*-secure* if for any subjects  $\mathbf{s}_1, \mathbf{s}_2$  and objects  $x, y$  if  $(\mathbf{s}_1, x, read) \in b$  and  $(\mathbf{s}_2, y, write) \in b$  and the classification of  $x$  dominates that of  $y$ , then  $\mathbf{s}_1 \cap \mathbf{s}_2 = \emptyset$ .

Each model in the sense of the previous section yields a unique model in the current sense since the security levels of all joint subjects is determined by the levels of the individual member subjects. It is easy to verify that if an instance of our new framework only allows singleton subjects, then the restrictions it places on a subject  $\{s\}$  is identical to the restrictions placed on  $s$  by our previous framework. Further, the same restrictions that applied to a subject  $s$  in the previous framework, now apply to any subject that contains  $s$  in the current framework. Our current model is more restrictive, however, in the sense that, for example, a subject may be denied joint *write* access to an object  $o$  which he could have single *write* access to because the person he wishes to share the access with may have *read* access to an object of higher classification.

Although the concept of having joint *read* or *write* access to an object is useful, joint access restrictions are most intuitive for the concept of changing security levels. To this end we redefine our notion of a state transition so that  $T$  is now a function  $T: \mathbf{S} \times R \times V \rightarrow V$ , and our notion of  $c_s$  and  $c_o$  so that they now are functions to  $P(\mathbf{S})$ . Our restriction on secure transitions can then be brought over intact to our new model.

Though our treatment of  $C$  is straightforward and has the advantage that the models of a framework form a Boolean Algebra, it has the drawback that it is not clear that all the instances of a framework make sense. For example, consider the system which contains two users  $x$  and  $y$ . The set of subjects in our new model will be  $\{\{x, y\}, \{x\}, \{y\}\}$ . Hence, it is possible to have  $c_o(w) = \{\{x\}\}$  for some object  $w$ . Some may regard as nonsensical a policy where  $x$  can change  $w$ 's security level, but is unable to change it operating in conjunction with  $y$ . If we wish to rule out such systems as bizarre, we can either treat such models as merely theoretical elements necessary to round out our framework or we can exclude them. To exclude them, we require that that range of each of  $C$ 's component functions is not  $P(\mathbf{S})$ , but rather  $\bar{P}(\mathbf{S})$  where  $\bar{P}(\mathbf{S})$  is the set  $\{X: X \subseteq \mathbf{S} \text{ and if } y \in X \text{ and } y \subseteq z \in \mathbf{S} \text{ then } z \in X\}$ . Intuitively, a member of  $\bar{P}(\mathbf{S})$  is generated from a set of elements of  $\mathbf{S}$  by gathering those elements and any subset of  $S$  that is a superset of any of those elements. Hence, for example, given our set  $S = \{x, y\}$ ,  $\mathbf{S} = \{\{x, y\}, \{x\}, \{y\}\}$ , and  $\bar{P}(\mathbf{S})$  contains the following five members ordered by  $\subseteq$ :



The top policy says that either  $x$  or  $y$  can perform the operation in question (and hence, can perform it jointly as well); the leftmost policy in the next row says that  $x$  can perform the operation (and hence, can perform it jointly with  $y$  as well), but that  $y$  cannot perform the operation without  $x$ ; that rightmost policy of that row is the analogous policy for  $y$ ; the policy on the next row says that  $x$  and  $y$  can perform the operation only when operating jointly; and the bottom policy says that no subject can perform the operation.

The advantage of so restricting  $C$  is that we have no policies that are nonsensical. The disadvantage is that the domain of the components of  $C$  no longer form a Boolean Algebra. However, it is straightforward to prove that the possible domains for each component does form a distributive lattice under the set theoretic operations  $\cap$  and  $\cup$ .

**Theorem:** For any set  $S$ ,  $\bar{P}(S)$  is a distributive lattice under  $\cap$  and  $\cup$ .

**Proof:** If  $\bar{P}(S)$  is a lattice, then it is distributive since  $\cap$  and  $\cup$  are distributive. Hence, we only need to prove that  $\bar{P}(S)$  is closed under  $\cap$  and  $\cup$ . Assume that  $a \in \bar{P}(S)$ ,  $b \in \bar{P}(S)$ , and  $a \cap b = X$ . Since  $a \subseteq S$  and  $b \subseteq S$ ,  $X \subseteq S$  so the first condition of  $X$  being a member of  $\bar{P}(S)$  is satisfied. Assume that  $y \in X$  and  $y \subseteq z \in S$ . Since  $y \in X$ ,  $y \in a$  and  $y \in b$ . Since  $y \subseteq z \in S$ ,  $z \in a$  and  $z \in b$  since  $a \in \bar{P}(S)$  and  $b \in \bar{P}(S)$ . Hence,  $z \in X$  so the second condition is satisfied. The argument for  $\cup$  is analogous.  $\square$

It is clear that if all models follow the same naming convention for subject and objects, then  $\subseteq$  is the partial ordering induced by our lattice and the top and bottom elements of our lattice are  $P(S)$  and  $\{\emptyset\}$ , respectively. Further, since the domains of each component of  $C$  form a lattice, the product domain is also a lattice. Hence, the set of models within a framework form a lattice. This means that the set of models is still partially ordered and that we can combine models by  $\cap$  and  $\cup$ , but we cannot, in general, compliment a model and remain in our framework.

#### 4. Discretionary Security

Though we have primarily been considering the notion of MAC, there is also the concept of discretionary access

control (DAC), which captures the notion of limiting access in a way not required by federal law. Since such a concept is a useful addition to government security and serves as one pillar of industrial security, it is worthwhile examining it briefly here.

In BLP, discretionary security is captured by a discretionary access matrix which, for any subject-object pair, lists the type of accesses that the subject may have to the object. This approach has the disadvantage of being extremely coarse grained, especially in the industrial world, where one does not want to give a subject blanket rights to alter an object in any way, but only the right to alter an object in a set of specified ways [2]. Discretionary access is better viewed as the right to execute certain programs on a specified object.<sup>6</sup> Ignoring the order of program arguments, it can be viewed as a function  $D: S \times O_1 \times O_2 \rightarrow \text{Boolean}$  where  $D(s, o_1, o_2) = \text{true}$  if and only if  $s$  has the right to execute program  $o_1$  on  $o_2$ .

The advantage of the present framework for such access restrictions is that it allows us to capture the notion that a user can access an object via a program, but only in conjunction with other users. Such restrictions form the second pillar of security for industrial purposes [2]. Hence, one may wish to say, at an abstract level, that the action of receiving an order can only be performed by a warehouse agent and an accountant operating jointly to make sure that nothing is paid for which is not received. The details of modeling such policies are beyond the scope of this paper, but it is plain to see that the resulting framework will allow for the comparison and construction of DAC policies in a way similar to our comparison and construction of MAC policies.

#### 5. Conclusion

We have developed a framework for traditional MAC security policies and a framework for security policies that contain n-person rules. The former framework forms a Boolean Algebra; the latter a distributive lattice. Both allow for the rigorous comparison of policies and the systematic creation of new policies from existing ones, though the former framework is more powerful for creating new policies from existing ones.

The next step is to consider models that permit the components of  $C$  to change as well. For example, who grants or revokes permission for a user to change an object's security level? In the mean time, both frameworks should support future research in security modeling by suggesting new models to be considered and the relations between these models and existing ones. For example, if we discover

<sup>6</sup> This is the approach taken, for example, in the SMMS [8] and SeaView [4].

a model  $M$  is too lax for our purposes, we can conclude that any model  $M^*$  such that  $M^* \geq M$  will also be too lax. Similarly if  $M$  is too restrictive, no model  $M^* \leq M$  need be considered. By creating and experimenting with such models, we will create a tool box of models that can serve a variety of purposes.

### Acknowledgments

I am grateful to Catherine Meadows for her careful reading of this paper and for her help in establishing the claim made in the proof of the theorem relating  $\equiv$  and  $\leq$ . I am also grateful to Thor Bestul, Dick Kemmerer, Carl Landwehr, and Teresa Lunt for their comments on a previous, extended version of this paper.

### References

1. D. E. Bell and L. J. LaPadula, "Secure Computer System: Unified Exposition and Multics Interpretation," MTR-2997, MITRE Corp., Bedford, MA (March, 1976). Available as NTIS AD A023 588
2. D. D. Clark and D. R. Wilson, "A Comparison of Commercial and Military Security Policies," pp. 184-194 in *Proc. 1987 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press (April, 1987).
3. D. E. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading (1982).
4. D. E. Denning, T. F. Lunt, R. R. Schell, M. Heckman, and W. Shockley, "The SeaView Formal Security Policy Model," SRI Interim Report A003, SRI International (1987).
5. R. J. Feiertag, K. N. Levitt, and L. Robinson, "Proving Multilevel Security of a System Design," *Proc. Sixth ACM Symposium on Operating Systems Principles* pp. 57-65 (1977).
6. J. A. Goguen and J. Meseguer, "Security Policies and Security Models," pp. 11-20 in *Proc. 1982 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press (April, 1982).
7. J. T. Haigh, "A Comparison of Formal Security Models," pp. 88-119 in *Proc. 7th DOD/NBS Computer Security Conference*, IEEE Computer Society Press (Sept. 1984).
8. C. Landwehr, C. Heitmeyer, and J. McLean, "A Security Model for Military Message Systems," *ACM Transactions on Computer Systems* 2(3) pp. 198-222 (August 1984).
9. J. McLean, "A Formal Method for the Abstract Specification of Software," *J. ACM* 31(3) pp. 600-627 (July 1984).
10. J. McLean, "Reasoning about Security Models," pp. 123-131 in *Proc. 1987 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press (April, 1987).
11. J. K. Millen, "A1 Policy Modeling," pp. 137-160 in *Proc. 7th DOD/NBS Computer Security Conference*, IEEE Computer Society Press (Sept. 1984).
12. D. L. Parnas, "The Use of Precise Specifications in the Development of Software," pp. 861-867 in *Proceedings of IFIP 77*, North Holland (1977).