

# Access Control Models and Security Labelling

Chuchang Liu

Angela Billard

Maris Ozols

Nikifor Jeremic

Information Networks Division  
Defence Science and Technology Organisation  
PO Box 1500, Edinburgh  
SA 5111, Australia

{*Chuchang.Liu, Angela.Billard, Maris.Ozols, Nikifor.Jeremic*}@dsto.defence.gov.au

## Abstract

Security labels convey information that is utilised to perform access control decisions, specify protective measures, and aid in the determination of additional handling restrictions required by security policies. In discussing security labelling, one of the most important aspects is to investigate access control models and obtain an appropriate technique for specifying the kind of security policies that are required. One problem with previous approaches to the specification of access control policies is that they are based on an idealisation of the real problem and give a first approximation: may or may not a subject access a given object? The binary, logical function is the essential starting point, but is generally insufficient to guide the hard decisions that are required by a variety of applications in the real world. Focusing on the issues regarding security labelling, this paper first proposes a technique for expressing need-to-know policies that are regarded as the basis for security labelling and should be followed in the labelling process. Then, based on the proposed lattice access control model dealing with both security levels and categories of objects, several security labelling principles are given. Finally, we propose a dynamic model for security labelling that not only provides support for dynamic labelling within a system but also a functional base for the design and implementation of a security labelling system.

*Keywords:* open system, security label, access control, security policy, dynamic labelling model, assurance.

## 1 Introduction

Information is widely recognised as a valuable asset, even a commodity, in the current ‘Information Age’. However, the value of information to its possessor is inextricably linked to the ability to protect and share it as the possessor deems appropriate. Information security includes the set of measures taken to protect data from unauthorised or accidental modification, destruction, or disclosure. Information security can be applied to all systems, whether automated, paper-based, human-based, or a combination of these. The systems considered in this paper are those that may be classed as open systems. An open system is viewed as a set of one or more computers, associated with software, peripherals, terminals, human operators, physical processes, information transfer methods, *etc.*, that forms an autonomous whole capable of storing, processing and transferring information.

Security labelling is one of the mechanisms that can be used in the provision of information security. It supports the correct handling of data within and

between systems, according to the sensitivity of the data. Typically, the value and sensitivity of information objects can vary greatly. Some objects contain critical information that requires a very high degree of protection, while others hardly require more than a very modest degree of protection. Furthermore, while objects may have an equal degree of sensitivity, the nature of these sensitivities may differ. The purpose of security labelling is to provide the means for making a concise assertion about an information object’s sensitivities, which may be used with security policy to usefully determine the appropriate protection and handling. Information contained within security labels can be utilised to control access, specify protective measures, and determine additional handling restrictions required by communication security policies (Internet CIPSO Working Group 1993).

The earliest work regarding security labels can be traced back to (Weissman 1969), where the label of an object is used to reflect the classification of the object. Housley (1993) discussed a security labelling framework designed especially for the Internet; and more recently Liu and Orgun propose a checkable model for security labelling (Liu & Orgun 2006), which captures a variety of security requirements. However, the previously proposed model is a static model, and does not address dynamic changes to security labels. Therefore, a motivation of this work is to investigate the issues regarding the influence of dynamic factors, and the intention is to provide a dynamic model for security labelling. The notion of security labelling also appears in security-typed languages, which have been proposed recently to enforce security properties including confidentiality and integrity by type checking (Heintze & Riecke 1998, Myers & Liskov 2000, Zdancewic *et al.* 2001). In security-typed languages, types are extended with security labels to enforce information flow control, but these labels are usually applied only to denote security classes associated with users and the resources that programs access (Zheng & Myers 2004).

Security labels should reflect and satisfy the range of requirements specified in the policy for the system. Security policies are varied, and include policies for data integrity and data confidentiality. Biba (1977) proposed a model that specifies the rule-based controls for writing and deleting that are necessary to preserve data integrity. It also specifies rule-based controls for reading to prevent a high integrity process from relying on data that has less integrity than the process. With regard to confidentiality, Bell and LaPadula (1976) defined a model that specifies the rule-based controls for reading that are necessary to preserve data confidentiality, and it also specifies rule-based controls for writing to ensure that data is not copied to a container where confidentiality can not be

guaranteed. In both the Biba and the Bell-LaPadula models, the security label is a static attribute of the data.

In discussing security labelling, one of the most important aspects is to investigate access control models and obtain an appropriate technique for specifying the kind of security policies that need to be applied. The problem is to find an appropriate model for security labelling. One problem with previous approaches to specifying access control policy is that they are based on an idealisation of the true problem and give a first approximation: may or may not a subject access a given object? The binary, logical function is the essential starting point, but is generally insufficient to guide the hard decisions that are required in implementation. In order to provide evaluative criteria by which adequacy of security architectures may be assessed and compared, new approaches for describing the full access control requirements are demanded.

Although there is a standard for security labelling defined in (FIPS188 1994) specifically for the purposes of information transfers, there is a lack of formal techniques for modelling security labelling. Therefore, one of the motivations of our work is to discuss these issues in general and provide a formal methodology for modelling security labels. We do not intend to describe specific practical labelling systems, but the methods and techniques of modelling security labelling would support the design and implementation of such systems.

One major contribution of this paper is to propose a lattice-based access control model for the support of security labelling. Focusing on the issues regarding security labelling, this paper first proposes a technique to express need-to-know policies that are regarded as a basis for security labelling and should be followed in the labelling process. Then, based on the proposed lattice access control model dealing with both security levels and categories of objects, we propose several security labelling principles. The other major contribution of the paper is that we propose a dynamic model for security labelling. This model not only provides support for dynamic labelling within a system but also provides a functional base for the design and implementation of a security labelling system. The main difference between the dynamic model and the checkable model proposed in (Liu & Orgun 2006) is that the dynamic model is capable of dealing with the dynamics of a system, including changes to a policy, and provides support for relabelling. In contrast, the checkable model is static and has no such ability although it captures a variety of security requirements in the labelling process.

The structure of the paper is as follows. Section 2 discusses access control models, and proposes a formal approach to specifying need-to-know policies. Section 3 presents several security labelling principles based on the lattice access control model. Section 4 discusses the security labelling framework, and proposes a model for security labelling. Section 5 discusses implementation of this model, and Section 6 discusses other issues, including the dynamic aspects of our security labelling model and further considerations in relation to assurance. Section 7 concludes the paper.

## 2 Access Control Models

A security policy is a set of criteria for the provision of security services that defines and constrains the activities of data processing facilities within a system; it states strict requirements for how material is to

be handled during storage, processing and transmission in order to maintain the security conditions that should be satisfied. With regard to access control, a security policy in its simplest form is a subset of a system-level security policy that defines the means for enforcing the access control policy (Ferraiolo, Kuhn & Chandramouli 2003).

Security labels provide information for making access control decisions within a security system, while the access control policy is the foundation on which we base security labelling. In this section, we firstly discuss how to express an access control policy. We then present a formal technique for specifying a need-to-know policy, which combines the security classification and the security categories of objects, and propose a practical access control model based on the previous lattice-representation method proposed by Sandhu (1993).

### 2.1 Subjects and Objects

First let us define a few terms.

*Subjects* are active entities that can perform actions within a system. Typically, we consider subjects to be processes executing on behalf of users. Users may also be considered to be subjects; however, because users may run many processes concurrently, when a user is treated as a subject, it is in association with a single process. For a given system, there is a set of subjects that is expressed as:

$$\mathcal{S} = \{s_i \mid i = 0, 1, \dots, n\}.$$

*Objects* are passive entities that can undergo actions. Typical examples of objects are resources, tools, or mechanisms used to maintain a condition of security in a computerized environment. Similarly, for the given system, we also have a set of objects expressed as:

$$\mathcal{O} = \{o_j \mid j = 0, 1, \dots, m\}.$$

In terms of access control, we say that the entity requesting access to a resource is the subject of the access and the resource that the subject attempts to access is the object of the access.

We now define a relation and an operation over the object set  $\mathcal{O}$ , respectively, as follows:

- If object  $o$  is a part of object  $o^+$  or  $o^+$  is a new object generated by adding something (data information) to object  $o$ , then we say that  $o^+$  contains  $o$  or  $o$  is contained in  $o^+$ , denoted by  $o \sqsubseteq o^+$ . Furthermore, if  $o \neq o^+$ , then  $o \sqsubset o^+$ , and we say  $o$  is properly contained in  $o^+$ .
- Let  $o_1, o_2 \in \mathcal{O}$ , the union of  $o_1$  and  $o_2$  is a new object in  $\mathcal{O}$ , denoted by  $o = o_1 \sqcup o_2$ .

It is easy to see that  $\sqsubseteq$  is a partial ordering relation over the object set. This relation can be extended when we consider a more generic case where  $o^+$  is produced with  $o$  as an input. Similarly, the operation  $\sqcup$  can be also extended by viewing  $o$  as the result produced with the inputs  $o_1$  and  $o_2$ . Considering that this paper focuses on security labels for data transferring, we restrict both  $\sqsubseteq$  and  $\sqcup$  to a simple case where  $o^+$  is regarded as a new object obtained by adding extra information into  $o$ , and  $o_1 \sqcup o_2$  is a new object generated by putting two objects together.

## 2.2 Security Classification and Mandatory Access Control (MAC)

Security classification, or level, is regarded as a hierarchical indicator of the degree of sensitivity to certain threats. For a given system, assume we have a particular set of security levels associated with a partial ordering relation. Formally,

**Definition 1** We call  $(\mathcal{L}, \leq)$  a security classification system, where  $\mathcal{L}$  is a set of security levels, and  $\leq$  is a partial ordering relation defined on  $\mathcal{L}$ . For any  $m, l \in \mathcal{L}$ ,  $m \leq l$  means that  $l$  is a higher security level than  $m$ , and is read as “ $m$  is dominated by  $l$ ”. Furthermore, if  $m \leq l$  but  $m \neq l$ , then we write  $m < l$  and say that  $m$  is strictly dominated by  $l$ .

Note that the infix notation  $m \leq l$  means the same as  $(m, l) \in \leq$ . Therefore,  $(m, l) \notin \leq$  means that  $m$  is not dominated by  $l$ .

Without loss of generality, the security classification system  $(\mathcal{L}, \leq)$  can be considered a (finite) lattice. In fact, every partial ordering set can be embedded in a lattice by including additional security levels if needed (Sandhu 1993). As a lattice,  $(\mathcal{L}, \leq)$  has two operators which are derived from the relation  $\leq$ . Formally, we have the definition as follows:

$$\begin{aligned} l \oplus m &= l.u.b.\{l, m\} \\ l \odot m &= g.l.b.\{l, m\} \end{aligned}$$

where  $l.u.b.\{l, m\}$  is the least upper bound of the set  $\{l, m\}$ , that is, assume  $l.u.b.\{l, m\} = a$ , then  $l \leq a$ ,  $m \leq a$  and, for any  $b$ , if  $l \leq b$  and  $m \leq b$  then  $a \leq b$ . Symmetrically,  $g.l.b.\{l, m\}$  is called the greatest lower bound of the set  $\{l, m\}$ . We will see in our discussion that the operation  $\oplus$  is the more useful of the two.

There also exists the unique element, say  $l_{min}$ , such that for all  $l \in \mathcal{L}$ ,  $l_{min} \leq l$ , and, symmetrically, there exists the unique element, say  $l_{max}$ , such that for all  $l \in \mathcal{L}$ ,  $l \leq l_{max}$ . Thus,  $l_{min}$  and  $l_{max}$  are called the minimum element and the maximum element of this lattice respectively or, accordingly, the lowest security level and the highest security level of the classification system.

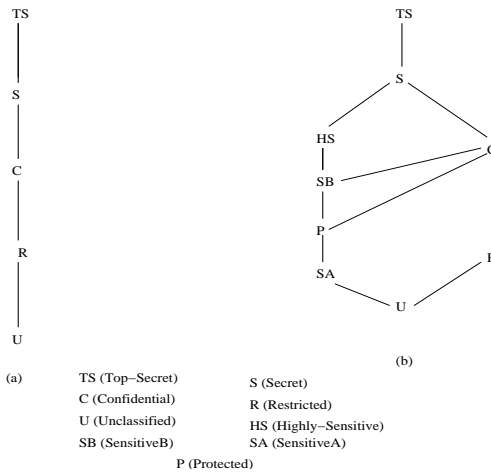


Figure 1: Security Classification Systems

Figure 1 illustrates two classification systems. The classification system in Figure 1(a) is a linear hierarchy, consisting of five security levels  $U$  (*unclassified*),  $R$  (*restricted*),  $C$  (*confidential*),  $S$  (*secret*) and  $TS$  (*top-secret*) with the ordering relation  $U < R < C < S < TS$ . This kind of linear security classification system, typically used in the military and government

sectors, can have any number of security classes. In the linear hierarchy, there are no incomparable security classes, and the definition of  $l \oplus m$  and  $l \odot m$  are then simply the maximum and the minimum, respectively, of  $l$  and  $m$  with respect to the relation  $\leq$ , e.g., we have  $R \oplus S = S$  and  $R \odot S = R$ . In a more generic classification system, such as the one in Figure 1(b), there are some incomparable security classes, e.g.,  $HS$  and  $C$ , but we have  $HS \oplus C = S$ , and  $HS \odot C = SB$ . In Figure 1, both of the systems have  $U$  as their lowest security level and  $TS$  as their highest. We can show that it is possible to implement a single security labelling system which encompasses multiple classification systems based on the lattice classification method.

Security policies may include inconsistencies, ambiguities, gaps, and conflicts with related policies. Usually, the reason for this is the lack of techniques for accurately expressing complex and detailed instructions written in a natural language. Therefore, it is important to provide a formal methodology that can be applied for expressing and reasoning about a variety of security policies.

In the following, we show that the mandatory access control policy can be expressed in terms of security classes attached to subjects and objects. Formally, there is a function that assigns a security level to each object and subject of a system:

$$\mu : \quad \mathcal{S} \cup \mathcal{O} \rightarrow \mathcal{L}$$

Any particular security level associated with an object implies a specific level of protection for the object, according to the enforced security policy, while a security level associated with a subject represents the level of clearance of the subject.

With the function  $\mu$  above, the mandatory access policy, BLP rules (see (Sandhu 1993)), can be expressed as follows:

- *No-read-up rule*: Subject  $s$  can read object  $o$  if and only if  $\mu(o) \leq \mu(s)$ .
- *No-write-down rule*: Subject  $s$  can write object  $o$  only if  $\mu(s) \leq \mu(o)$ .

## 2.3 Categories and Need-to-Know Policy

Categories are a different concept from security levels. For example, categories may correspond to different tasks in a system. Agents with the same security level may not have the same right to access an object with a particular category. According to BLP rules, a subject with a certain security level is usually allowed to read a file (an object) with the same or lower security level. However, in some situations, such as within an implementation of a military project, some individuals may not have a need-to-know for this project, although they may have the same or higher security level than the one assigned to the project files. Therefore, there is a need to study the notation of categories deeply with the purpose of providing an appropriate access control model for such systems. We will show that combining categories together with security levels can rule out access to resources by individuals who do not have a need-to-know.

Given the object set  $\mathcal{O}$ , the subject set  $\mathcal{S}$ , the security classification system  $(\mathcal{L}, \leq)$  and a function  $\mu$ , we define

$$\mathcal{O}_l(\mu) = \{o \in \mathcal{O} \mid \mu(o) = l\}, \text{ where } l \in \mathcal{L}.$$

That is,  $\mathcal{O}_l(\mu)$  is defined as the set consisting of all objects whose security level is  $l$ , for simplicity, we write it as  $\mathcal{O}_l$ . Thus, for any object  $o$ , since it has exactly one security level, there is an unique set, i.e.,  $\mathcal{O}_{\mu(o)}$ , such that  $o \in \mathcal{O}_{\mu(o)}$ . Therefore, for any  $l, m \in \mathcal{L}$ , if  $l \neq m$ , then  $\mathcal{O}_l \cap \mathcal{O}_m = \emptyset$ , the empty set. Furthermore, we have  $\bigcup_{l \in \mathcal{L}} \mathcal{O}_l = \mathcal{O}$ , which can easily be proved, so the sets  $\mathcal{O}_l$  partition  $\mathcal{O}$ .

Introducing (new) categories into a system may have a variety of reasons, such as new projects being established and restricted to be known by only some agents. Assuming there is a set of categories defined for a given system, then by defining an operation/relation on this set, we are able to form a category algebra for this system. Formally, we have:

**Definition 2** A system,  $(\mathcal{C}, |)$ , is called a category algebra, where  $\mathcal{C}$  is the category set and  $|$  is a binary operator defined on  $\mathcal{C}$ , that satisfy the following conditions:

- $\epsilon \in \mathcal{C}$ , where  $\epsilon$  is the null category;
- For any  $c, e \in \mathcal{C}$ ,  $c|e = e|c$ . We may read  $c|e$  as ' $c$  or  $e$ ', which represents a new category produced by combining two categories as one with the operation  $|$ ;
- For any  $c \in \mathcal{C}$ ,  $c|\epsilon = c$ , and  $c|c = c$ .

According to this definition, a partial ordering relation  $\leq$  can be introduced with the following definition:

- For all  $c \in \mathcal{C}$ ,  $\epsilon \leq c$ ; and  $c \leq c|e$  for any  $e$  in  $\mathcal{C}$ .

Furthermore, we say a category is *single* if it is not a combination of two or more actual categories with the operation  $|$ . An actual category means that it is not the null category  $\epsilon$ . In contrast, a combination of two or more actual categories is not a single category.

Thus, the category algebra  $(\mathcal{C}, |)$  is also expressed as the partial ordering set  $(\mathcal{C}, \leq)$ . We can then embed the partial ordering set in a lattice by including additional categories as required. For example,  $\mathcal{C} = \{\epsilon, c, e\}$ , we need to add  $c|e$  to  $\mathcal{C}$ ,  $(\mathcal{C}, \leq)$  then becomes a lattice. Therefore, in general, we can view the category algebra as a lattice. Note that the partial relation in  $(\mathcal{C}, \leq)$  is in fact a different relation from the one in  $(\mathcal{L}, \leq)$ , but without confusion, we denote both by  $\leq$ .

In the following, we discuss how to combine security levels with categories to establish an extended security classification system. Given a security classification system  $(\mathcal{L}, \leq)$ , and a category algebra  $(\mathcal{C}, \leq)$ , we first obtain a subset, say  $\Omega$ , of  $\mathcal{L} \times \mathcal{C}$  by the recursive definition given below.

- For all  $l \in \mathcal{L}$ ,  $(l, \epsilon) \in \Omega$ ,
- If an object with category  $c$  is contained or will be created in  $\mathcal{O}_l$ , we have  $c \in \mathcal{C}$  and then  $(l, c) \in \Omega$ .
- For all  $l, m \in \mathcal{L}$ , if  $l \leq m$  and  $(l, c) \in \Omega$ , then  $(m, c) \in \Omega$ .
- For all  $l \in \mathcal{L}$ , if  $(l, c) \in \Omega$  and  $(l, e) \in \Omega$ , then  $(l, c|e) \in \Omega$ .

Then we define a partial ordering relation,  $\preceq$ , on  $\Omega$  as follows:

- For any  $(l, c) \in \Omega$  and  $(m, e) \in \Omega$ ,  $(l, c) \preceq (m, e)$  if and only if  $l \leq m$  and  $c \leq e$ .

Thus,  $(\Omega, \preceq)$  is an extended security classification system with categories. It is also a lattice classification system. Figure 2 is an example of such a classification system, where the set of security levels is  $\{U, R, C, S, TS\}$  and the set of categories contains only three different elements (i.e. three single categories),  $X, Y$  and  $Z$ , not including  $\epsilon$ .

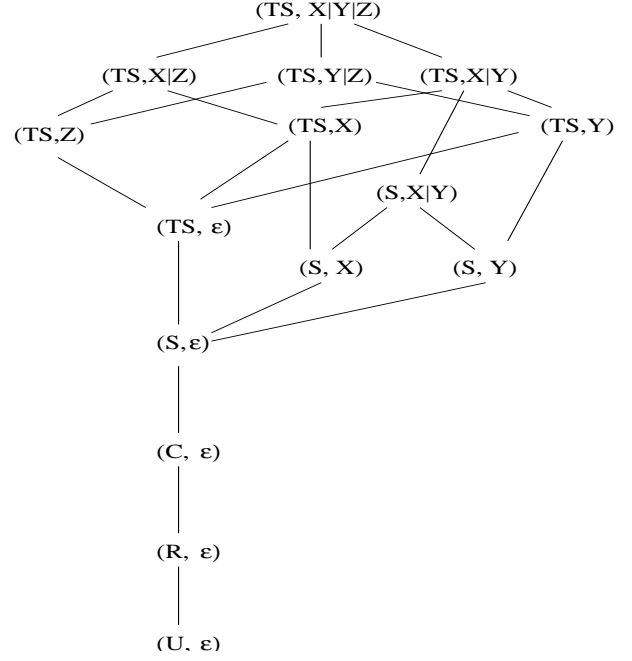


Figure 2: A Classification System with Categories

It is easy to show that, if  $\mathcal{C} = \{\epsilon\}$ , i.e., there is no actual category, then  $(\Omega, \preceq)$  and the classification system  $(\mathcal{L}, \leq)$  are isomorphic.

A need-to-know security policy can be expressed in terms of the security levels and categories attached to subjects and objects. Formally, there is a function that assigns a security level with categories to each object or subject within a system:

$$\lambda : S \cup \mathcal{O} \rightarrow \Omega$$

$\lambda(o) = (l, c)$  implies the category of  $o$  is  $c$  and the specific level of protection for this object is  $l$  according to the security policy being enforced, while  $\lambda(s) = (l, c)$  gives the clearance of the subject  $s$ . Note that, if there is no specific category assigned to an object  $o$  with the security level  $l$ , then  $\lambda(o) = (l, \epsilon)$ . Similarly for subjects.

Thus, with the function, a need-to-know policy can be expressed as:

- *Need-to-know rule*: Object  $o$  can be accessed by subject  $s$  if and only if  $\lambda(o) \preceq \lambda(s)$ .
- *Object-creating rule*: Subject  $s$  can write object  $o$  only if  $\mu(s) = \mu(o)$  and  $\lambda(o) \preceq \lambda(s)$ .

The need-to-know rule is used to guarantee that there is no one who is able to access resources for which he does not have a need-to-know. The object-creating rule is similar to the no-write-down rule, which guarantees that any subject can only write objects with the same security level as the subject possesses.

### 3 Security Labelling Principles

In a system, objects are created and destroyed dynamically. There is no bound on the number of objects that can be created, but the assumption that at all times the set of objects is finite is appropriate, therefore, practically we are able to label all objects. We note that, when an object is created, a security level must be assigned to it at the same time by the applicable security policy. Also, with some applications, objects may need to be assigned an appropriate category as well.

Based on the given mandatory access policy (BLP rules) above, the following principles should be applied in assigning security levels to objects: For all  $o, o_1, o_2 \in \mathcal{O}$ , and any  $s \in \mathcal{S}$ ,

- (1) If object  $o$  is created (owned) by the subject  $s$ , then  $\mu(o) = \mu(s)$ .
- (2)  $\mu(o) \leq \mu(o^+)$ , for any object  $o^+$  in which  $o$  is contained.
- (3)  $\mu(o_1 \sqcup o_2) = \mu(o_1) \oplus \mu(o_2)$ .

These principles give a method (constraint) for assigning a security level to any (new) object. As we will see, the security level of an object must be contained, as a field, in the object's label. Therefore the principle actually provides a technique for generating (part of) the label.

It is easy to show that these principles are consistent with the mandatory access BLP rules. In other words, these principles satisfy the mandatory access policy. In fact, according to (1), the subject is not allowed to create an object whose security level is lower than the subject's security level (no-write-down); and, according to (2) and (3), subjects whose security level is lower than an object's level can never get the information contained in this object (no-read-up).

In some applications, principle 1 may be shifted as follows:

- (1') If object  $o$  is created (owned) by the subject  $s$ , then  $\mu(s) \leq \mu(o)$ .

However, with this principle, there must be some auxiliary rules for determining the exact security level that is assigned to any particular object generated by a subject. Similarly, in principle 2,  $\mu(o^+)$  depends on what is added to  $o$  to obtain  $o^+$ . Therefore, in a practical application, we may also need some auxiliary rules for determining  $\mu(o^+)$ , the security level of  $o^+$ , case by case.

Based on a need-to-know policy, we have the following principles for assigning security levels and categories to objects: For all  $o, o_1, o_2 \in \mathcal{O}$ , and any  $s \in \mathcal{S}$ ,

- (4) If object  $o$  is created (owned) by the subject  $s$ , then  $\mu(o) = \mu(s)$  and  $\lambda(o) \preceq \lambda(s)$ .
- (5)  $\lambda(o) \preceq \lambda(o^+)$ .
- (6) Assume  $\lambda(o_1) = (l_1, c_1)$  and  $\lambda(o_2) = (l_2, c_2)$ , then  $\lambda(o_1 \sqcup o_2) = (l_1 \oplus l_2, c_1|c_2)$ .

It is also easy to show that principles (4) – (6) are consistent with the need-to-know and object-creating rules that are employed to express the need-to-know policy.

### 4 A Dynamic Model for Security Labelling

In this section, we present a model for security labelling systems. Additionally, if a labelling system does not provide sound labels that satisfy the required security properties, then it cannot be regarded as a correct system. In particular, if the security policy within a label is not satisfied, then the label is useless. Therefore, we follow with a discussion on label validation, as well as some issues regarding relabelling within this model.

#### 4.1 Security Tags and Label Format

A security label attached to an object should address the range of security aspects required by the applicable security policy. These include the level of protection to be given to the object, who is authorised to access the object, how it can be transmitted between systems, and so on. Information can be encoded in security labels by the use of security tags or tag sets. A *security tag* is an information unit that contains a representation of certain security-related information (e.g., a restrictive attribute bit map). A named *tag set* is the field containing a *TagSetName*, which may be a numeric identifier, and its associated set of security tags.

Caveats are a concept closely related to security tags. We assume that in a system/organisation there are a number of standard caveats from which the user (label creator) may choose when generating security labels. A caveat has the following format:

$$(TagType, TagName, \_)$$

where “\_” is a field associated with the *TagName* and may be used to contain caveat qualifiers. If no qualifiers are required, then this is deemed by the specification to have the value “NULL”.

There are five basic defined security tag types that are recommended for use in labelling systems (ITU-T Recommendation X.841 2000). They are *Restrictive bit map* (REST), *Enumerated* (ENUM), *Range* (RANG), *Permissive bit map* (PERM), and *Informative* (INFO).

*TagName* gives the name of the tag, such as, for example, *SendTo* and *DeptOnly*. The meaning of any *TagName* should be defined at the design stage of the labelling system development, for instance, *SendTo* stands for “send to” and *DeptOnly* for “department members only”. Two example caveats are given as follows:

$$\begin{aligned} w &= (PERM, SendTo, \{A_1, \dots, A_n\}) \\ u &= (REST, DeptOnly, \_) \end{aligned}$$

A *security label* can consist of one or more security tag sets and, informally, it is a marking that is bound to an object, designating the security attributes of that object. In this view, a *security labelling system* is a system that is used to generate security labels for information objects that need to be labelled.

To simplify our discussion, we assume that each object will be assigned only one label at a time, and every label has only one tag set. In practice, a label may contain information regarding a security label identifier and the length of the label etc. In compliance with X.841 structures (ITU-T Recommendation X.841 2000), security labels are defined as having the following format:

$$[PolicyID, Classification, Category, TagSet]$$

where the field *PolicyID* is the object identifier for the applicable policy, the field *Classification* is the security level of the object to be labelled, the field *Category* is the security category of this object, and the field *TagSet* is a set of caveats.

As an example, given a security policy,  $p$ , and two caveats,  $w$  and  $u$ , as above, then we may have labels  $[p, S, X, w]$ ,  $[p, S, X, u]$ , and  $[p, S, X, \{w, u\}]$ , where we assume  $S$  to be the security level and  $X$  the category shown in Figure 2.

Liu and Örgun (Liu & Örgun 2006) have pointed out that for any label with a single caveat, based on the definition of the caveat together with the security level and the category (as it appears in the label format proposed in this paper), one should be able to identify the subjects to whom an object with this label can be delivered or who are permitted to access this object. For example, an object labelled with  $[p, S, X, w]$  can be delivered to only those agents who are in  $\{A_1, \dots, A_n\}$  and hold a security clearance equal to or higher than  $S$  and have the right to access category  $X$ ; and an object labelled with  $[p, S, X, u]$  can be delivered to only those department members who hold a security clearance equal to or higher than  $S$  and have the right to access category  $X$ . For a label containing more than one caveat, the delivering domain can be obtained by performing an intersection of the delivering domains associated with each individual caveat in the tag set. For example, an object labelled with  $[p, S, X, \{w, u\}]$  can be delivered to only those who are both department members and in the set  $\{A_1, \dots, A_n\}$ , hold a security clearance equal to or higher than  $S$ , and have the right to access category  $X$ .

## 4.2 The Model

In the discussion of security labelling models, we propose a framework for security labelling as follows:

**Definition 3** *Given an open system  $G$ , let  $S$  be the subject set,  $\mathcal{O}$  the object set,  $\mathcal{P}$  the policy set, and  $(\mathcal{L}, \leq)$  the lattice-based security classification system and  $(\mathcal{C}, |)$  the security category algebra. Let  $\Omega \subseteq \mathcal{L} \times \mathcal{C}$  and  $\preceq$  be the partial ordering relation over  $\Omega$ , derived from  $(\mathcal{L}, \leq)$  and  $(\mathcal{C}, |)$  mentioned as above, such that  $(\Omega, \preceq)$  forms a security classification system with categories for  $G$ ; and let  $\lambda$  be a total function defined from  $S \cup \mathcal{O}$  to  $\Omega$ :*

$$\lambda : S \cup \mathcal{O} \rightarrow \Omega,$$

*then we call  $\langle S, \mathcal{O}, \mathcal{P}, (\Omega, \preceq), \lambda \rangle$  the security labelling framework for the system  $G$ .*

The framework changes dynamically. The changes come from two aspects: dynamic changes to the security policies, which may directly lead to changes in  $(\Omega, \preceq)$ , and dynamic changes to subjects and objects (the sets  $S$  and  $\mathcal{O}$ ), from which corresponding changes to the function  $\lambda$  may result. Considering that security policies are relatively fixed for some period of time within a system, at any given time or within a reasonable time-period, the framework is definitely determined.

Based on the security labelling framework, a formal definition for security labelling models is given as follows:

**Definition 4** *Let  $\Theta = \langle S, \mathcal{O}, \mathcal{P}, (\Omega, \preceq), \lambda \rangle$  be the security labelling framework for a given system  $G$ . Then a model for security labelling is a function depending*

*on the framework, denoted  $\nu(\Theta)$ , that assigns a label to each object. That is, for any object  $o \in \mathcal{O}$ , we have the security label  $\nu(\Theta)(o) = [p, l, c, \{w, \dots, u\}]$  bound to the object, where  $p \in \mathcal{P}$ ,  $(l, c) \in \Omega$  and  $\lambda(o) = (l, c)$ , and  $w, \dots, u$  are taken from the set of caveats. We may simply call  $\nu(\Theta)$  a security labelling system for the system  $G$ .*

Intuitively, a security labelling system for a given system is a function based on the security labelling framework, which assigns labels to all objects in this system.

Since the function  $\nu(\Theta)$  depends on the security labelling framework  $\Theta$  and  $\Theta$  can change dynamically, the model is a dynamic model. That is, at any particular time, the label assigned to each object depends on the security labelling framework at that time. This model is distinguished from the mechanically checkable model proposed in (Liu & Örgun 2006) in that the checkable model is, in fact, a static model that does not address the effects of dynamic changes to the framework and therefore does not support relabelling directly; while the model proposed in this paper captures the dynamic changes within a system, and provides support for relabelling when necessary.

## 4.3 Security Domains and Label Validation

As we said before, access control policies are the basis on which security labelling relies. An access control policy is defined not only in terms of its access control rules but also in terms of the access control domain that determines the subjects and objects to which the policy applies (Bidan & Issarny 1998). Informally, a security domain is a collection of subjects, to which a single security policy applies that is administered by a single authority.

Assume that an organisation has a security policy as follows:

- only those who have the key  $k$ , used to open a restricted room, can enter that room.

This policy may formally be expressed by

$$P : \forall x \forall r (\text{restricted}(r) \wedge \text{enter}(x, r) \rightarrow \text{has}(x, k_r)).$$

where  $k_r$  is the key to open the room  $r$ . Obviously, the security domain related to a specific policy is a subset of the subject set in a given system. With the policy  $P$  above, the security domain should consist of all individuals who are capable of walking from one place to another (including, for example, human beings, robots etc.); any other subjects do not belong to this domain. In other words, the security domain is a set of all subjects that are affected by this policy, while other subjects are not involved in this policy.

We now consider some notions related to security domains, which can be used for label validation.

**Definition 5** *Let  $p$  be a security policy, and  $o$  an object. A policy-implementation domain related to  $p$  and  $o$  can be expressed by*

$$PID(p, o) = \{s | s \in SD(p) \ \& \ p^*(s, o)\},$$

*where  $SD(p)$  is the security domain for the policy  $p$  and  $p^*$  is regarded as a property corresponding to the policy  $p$ . That is, the domain  $PID(p, o)$ , as a subset of  $SD(p)$ , consists of all subjects that satisfy the policy  $p$  regarding the access to the object  $o$ .*

With the policy  $P$  above, consider a restricted room  $A$ , denoted  $rA$ , then the property applied to find the policy-implementation domain could be  $p^*: has(s, k_{rA})$ . Thus, we have  $PID(P, rA) = \{s | has(s, k_{rA})\}$ , which consists of all individuals who can enter the room  $A$ .

The policy-implementation domain provides a mechanism (or an algorithm) for guaranteeing that the policy is satisfied. According to the above definition, the policy-implementation domain seems to depend on the security property that corresponds to the policy, therefore, we may have a number of different expressions for the domain, but the policy-implementation domain itself is unique.

With an access control policy, the policy-implementation domain explicitly gives the scope containing all individuals (subjects) who are allowed to access an object under the policy. The security label that is bound to an object provides the actual domain that contains those individuals who may finally be able to touch (or have) the object according to this label. Recall the example above, assume that the label  $[P, R, \epsilon, (REST, DeptOnly, \_)]$  is bound to  $rA$ . In the label,  $P$  is only the object identifier of the policy. We may not know whether other fields in the label are consistent with the policy, as the policy's actual content does not appear in the label, while the label implementation seems to be based only on other fields. Therefore, there is another notion, called a label-implementation domain, that may not be the same as the policy-implementation domain. We have:

**Definition 6** Let  $\alpha$  be the label bound to an object  $o$ . The label-implementation domain of the object  $o$  based on  $\alpha$ , denoted  $LID(\alpha, o)$ , is defined by  $LID(\alpha, o) = \{s | can\_have(s, o) \text{ based on } \alpha\}$ , where  $can\_have(s, o)$  means that the subject  $s$  may access  $o$  according to the label  $\alpha$ .

Continuing the example of the restricted room, let  $\alpha = [P, R, \epsilon, (REST, DeptOnly, \_)]$  be the label bound to the room  $rA$ , then there are two cases:

- case 1:  $LID(\alpha, rA) = PID(P, rA)$ . The case holds if the key that is used to open the room  $rA$  is issued only to those who are in the department and have the security clearance equal to or higher than  $R$ .
- case 2:  $LID(\alpha, rA) \neq PID(P, rA)$ . This case happens when someone has the security clearance equal to or higher than  $R$  and is not in the department, but has the key to open the room  $rA$ , or someone who has no key to open this room but is in the department are allowed to enter the room.

Furthermore, we have:

**Definition 7** Given the model  $\nu(\Theta)$  and an object  $o$ , let  $\nu(\Theta)(o) = \alpha = [p, l, c, \{w, \dots, u\}]$ , then we say that the label  $\alpha$  violates the policy  $p$  if  $LID(\alpha, o) \neq PID(p, o)$ ; and we say  $\alpha$  is consistent with the policy  $p$  or  $\alpha$  is valid if  $LID(\alpha, o) = PID(p, o)$ .

#### 4.4 Relabelling

Given a system  $G$ , assume that initially, say at time 0, the security labelling framework for the system is  $\Theta$  and there is no change to  $\Theta$  until time  $t$ , then at time  $t$  the security labelling framework for  $G$  remains  $\Theta$ . More generally, we have:

**Definition 8** Assume that at time  $t_1$  the security labelling framework for a given system  $G$  is  $\Theta$  and there is no change to  $\Theta$  until time  $t_2$ , then we say the security labelling framework for the system  $G$  is relatively static in the time interval  $[t_1, t_2]$ .

Thus, the following assertion is straightforward.

- Given a system  $G$ , assume that at time  $t_1$  the security labelling framework for the system is  $\Theta$  and it is relatively static in the time interval  $[t_1, t_2]$ , then if a label is valid at  $t_1$ , then it is valid at all times in  $[t_1, t_2]$ .

That is, if the security labelling framework has not changed, then there is no need for relabelling.

Inversely, if at time  $t$ , the security labelling framework for a system is  $\Theta$  but there are some changes to  $\Theta$ , then it is obvious that at the next moment in time,  $t + 1$ ,  $\Theta$  is no longer the security labelling framework for the system.

There are several problems that arise from such dynamic changes to the security labelling framework which are worth investigating. They are:

- For any given object  $o \in \mathcal{O}$ , when does relabelling become necessary? In other words, is the label  $\nu(\Theta)(o)$  no longer valid at time  $t + 1$ ? Or do we have  $\nu(\Theta')(o) = \nu(\Theta)(o)$  where  $\Theta'$  is the security labelling framework at  $t + 1$ ?
- If there is a need for relabelling, how do we deal with it?

Assume  $\Theta = \langle \mathcal{S}, \mathcal{O}, \mathcal{P}, (\Omega, \preceq), \lambda \rangle$  and  $\Theta' = \langle \mathcal{S}', \mathcal{O}', \mathcal{P}', (\Omega', \preceq), \lambda' \rangle$  are the security labelling frameworks for the system at time  $t$  and at  $t + 1$ , respectively.  $\Theta'$  is the framework that results from applying changes to  $\Theta$ . Since changes can be complex, it is not easy to analyse the effects of a variety of changes to the security labelling framework. We do not intend to investigate all such changes here, but consider two simple cases as examples to provide some techniques for dealing with the relabelling issues.

*Case 1:* In this case, the only change is that several new objects are created at  $t$  and added to the object set. Thus, we have  $\mathcal{S}' = \mathcal{S}$ ,  $\mathcal{P}' = \mathcal{P}$ ,  $\Omega' = \Omega$ . The differences between  $\Theta$  and  $\Theta'$  are:  $\mathcal{O} \subset \mathcal{O}'$  and  $\lambda \neq \lambda'$ . However, because the policy set and the classification system are not changed, the function  $\lambda'$  must satisfy that

$$\lambda'(x) = \begin{cases} \lambda(x) & \text{if } x \in \mathcal{S} \cup \mathcal{O} \\ (l, c) & \text{otherwise} \end{cases}$$

Therefore, in this case, for all  $o \in \mathcal{O}'$ , if  $o \in \mathcal{O}$ , then there is no need for relabelling; if  $o \notin \mathcal{O}$ , we assign a label to it.

*Case 2:* In this case, we assume the only change is that the policy  $p$  is replaced by  $p'$ . Then relabelling involves only those objects whose label contains the object identifier of the policy  $p$  in the the field *PolicyID*.

Assume  $\Theta = \langle \mathcal{S}, \mathcal{O}, \mathcal{P}, (\Omega, \preceq), \lambda \rangle$  is currently the security labelling framework for a system, and there is a change  $\sigma$  to  $\Theta$ , then at the next moment in time there must be some objects in  $\mathcal{O}$  that require relabelling. One question is whether we can, based on the change  $\sigma$  to  $\Theta$ , determine the maximum subset

of  $\mathcal{O}$  where no object needs to be relabelled; or what kind of changes would allow one to determine such a maximum subset. The other question is how to define a relabelling function, such as the one suggested by Foley *et al* (Foley et al 1996), cooperatively within our model to satisfy the requirements for relabelling. We leave such problems for future work.

## 5 Implementation – Technical Issues

The implementation of our model involves many technical aspects, including the representation of the label syntax, the specification of security policy, label creation and validation, label binding mechanisms, etc. Implementing a practical security labelling system is not in the scope of this paper. Instead, we consider several technical problems, and identify some essential elements in implementing a security labelling system.

In this section, we first discuss the syntactical representation of labels and the technique for the specification of security policies, then explore two specific aspects – techniques for binding security labels to objects, and the basic components for implementing a security labelling system.

### 5.1 Label Specification

Having given the label format, the specification of labels can be done in many different ways. One way is based on the (traditional) formal specification technique, where the labels are specified with an abstract syntax, such as the specification proposed in (Liu & Orgun 2006). The advantage of this method is that such a specification provides a clear hierarchical structure for the system designer, and it is easily understood; however, the specification may be a little far from the actual programming language adopted in the implementation.

ASN.1 (Dubuisson 2000) also gives formal notations for high level specification of information to be exchanged between interacting applications, operating systems, programming languages, or other specifics associated with these applications. Considering that dynamic changes may often happen within a security labelling framework, the ASN.1 specification technique would be a good candidate for label specification, as recommended in (ITU-T Recommendation X.841 2000). With this notation, the security label is specified as follows:

```

Label ::= SEQUENCE {
    policyID      SecurityPolicyIdentifier
    classification SecurityLevel
    category      SecurityCategory
    tagSet        TagSet }

SecurityPolicyIdentifier ::= ObjectIdentifier
SecurityLevel ::= NaturalNumber
SecurityCategory ::= AlphabetLetter
TagSet ::= SET OF SecurityTag

SecurityTag ::= SEQUENCE {
    tagType      TagType
    tagName      TagName
    associateField AssociateField }

TagType ::= CHOICE of Type
TagName ::= ObjectIdentifier
AssociateField ::= PrivacyMark (Optional)

```

With the type SecurityLevel, there must be a function to map any security level of the set  $\mathcal{L}$  to a natural number representing the security level. As a result, only classifications that are total orders (such as Figure 1(a)) can be supported. Similarly with SecurityCategory, there needs to be a function to map any security category of the set  $\mathcal{C}$  to a letter in a given alphabet representing the category. The specification may also rely on the availability of a registration service to assign *TagSetName* and serve as the repository of the semantics, special handling rules, and other details required for the use of security policy-specific label sets. When security labels are associated with or require processing by a specific application, the TagSet element of the security labels can also be used to carry application specific data.

The ASN.1 format provides a clear structure for specifying security labels that meet the requirements of our proposed security labelling model. In addition, the ASN.1 specification can be easily mapped to actual programming code in the implementation stage.

### 5.2 Security Policy Information File

Security policy is the basis for decisions made by access control mechanisms. Domain-specific security policy is conveyed via the Security Policy Information File (ITU-T Recommendation X.841 2000). The Security Policy Information File (SPIF) contains a sequence of several items, including **securityClassifications**, which maps the classification in the security label to a classification in the clearance attribute, **securityCategories**, which maps the security categories in the security label to the security categories in the clearance attribute, and so on. The SPIF can also be defined with an ASN.1 specification.

Here we would only mention that an interpretation of a SPIF may be ambiguous, since the SPIF is usually obtained based on a security policy written in a natural language. Therefore, in the implementation of our model, it is important to provide a technique for translating the SPIF to a formal representation. Such formal representation of the SPIF not only allows agents to accurately interpret the policy, but also provides support for identifying whether there are conflicts between policies, and for proving the consistency of the label with the policy that is applied.

### 5.3 Binding Techniques for Labels

For binding the security label to an information object, we may or may not choose to use a cryptographic service. As a consequence, we have two methods for binding labels to objects stated as follows:

- *The method without the use of a cryptographic service.* In this method, a copy of the data, say object  $o$ , and a copy of the security label  $\nu(\Theta)(o)$  are stored together, as a single data record, inside the secure boundary of the system. With this binding method, no cryptographic function is needed.
- *The method with the use of a cryptographic service.* In this method, a digital signature algorithm (*SigAlg*) together with the private key ( $K_s$ ) for a public key algorithm is used. We can then generate the digital signature

$$\text{SigAlg}(K_s, H(o), \nu(\Theta)(o)),$$

for the object and its label, where  $H$  is a public function such that  $H(o)$  does not reveal information about  $o$ . In this case, the digital signature  $SigAlg(K_s, H(o), \nu(\Theta)(o))$  is stored together with the object  $o$  and the label  $\nu(\Theta)(o)$  in a data record; the generated digital signature binds  $\nu(\Theta)(o)$  to  $o$ . With this binding method,  $\nu(\Theta)(o)$  and  $SigAlg(K_s, H(o), \nu(\Theta)(o))$  need not be stored inside the secure boundary of the system.

With the first method, the system needs to be capable of protecting the integrity of the security label and the integrity, and possibly also the secrecy, of the data. The second method allows the use of the public key of the public key algorithm as a verification key to verify the signature. If a cryptographic service is invoked with an incorrect value of  $\nu(\Theta)(o)$ ,  $o$  or  $SigAlg(K_s, H(o), \nu(\Theta)(o))$ , then the inconsistency is detected.

#### 5.4 Main Components of a Labelling System

Intuitively, within a security labelling system, the labelling function accepts an object as input, and outputs a security label, which will be bound to this object. In order to implement the system, one needs to build a number of mechanisms to perform a variety of functions. In our model, the security labelling function is performed by three major mechanisms. Most of the mechanisms/components given below are similar to those that have been proposed in (Liu & Orgun 2006), but there is one new component, the RLM (Relabelling Mechanism), which is particularly related to the dynamic model and dealing with changes of environment and relabelling requirements.

- PIM (Policy Identification Mechanism): a mechanism to determine which policy is applied to a specific object;
- SCFM (Security-level & Category Finding Mechanism): a mechanism to find the security level and category for any given object to be labelled;
- CCM (Caveat Choosing Mechanism): a mechanism to determine what caveats may or should be chosen for constructing the label for an object;

All these mechanisms operate based on the security labelling framework, which consists of three modules, SOD (Subjects & Objects Database), PM (Policy Management), and TR (Tags Registration).

Other important components contained in our labelling system are:

- VRE (Validation & Reasoning Engine): a reasoning engine used to check the consistency of security labels produced. It directly connects with the labelling framework as well as the security labelling function.
- LBM (Label Binding Mechanism): a mechanism applied to bind a valid label to an object.
- RLM (Relabelling Mechanism): a mechanism dealing with dynamic changes to the security labelling framework and relabelling those objects that are affected by the changes.

Note that in most systems PIM, SLFM, and CCM require human input and judgement.

## 6 Other Issues

An access control model that supports the dynamic aspects of security labelling is becoming increasingly important in order to accurately represent the use of classifications and categories in the real world. While security classifications tend to remain relatively static, need-to-know categories and the associated policy need to be much more flexible. The access control model we have described addresses a number of issues associated with dynamics, including the dynamic creation and deletion of objects, dynamic creation and modification of policies and the relabelling of existing objects. To support this model, the label format needs to be extensible to allow for the creation of new categories and policies.

The model deals with dynamic changes by viewing the framework over time as a series of distinct static frameworks. This allows for changes without requiring the explicit notion of time. However, there are some limitations to this approach. In particular, a category that changes meaning over time will, in the described model, always require the creation of new categories to represent the different meanings. Additionally, objects associated with the original category may need to be relabelled and the policy changed to support the new framework. In some cases, this may be the most appropriate way of dealing with the issue. However, in other cases, the changes may be subtle or directly related to time, resulting in a proliferation of new categories and relabelling requirements in order to deal specifically with time-related issues. The explicit use of time in the model could alleviate this problem by allowing time-based constraints to be expressed in the policy, rather than in the categories themselves.

We also would mention an important but frequently neglected aspect of access control that relates to assurance requirements. There are several major aspects, but we refer to just two of them here.

Firstly, not all access control need necessarily be carried out to the same assurance level. Within Defence, the access controls relating to mandatory access control, particularly with respect to differences of classification, must be carried out at high levels of assurance. In contrast, many need-to-know restrictions may be implemented using very standard lower-assurance solutions, similar to those acceptable in commercial environments. These differences in assurance requirements are often captured in an ad hoc way. For example, the network encryption policy used to enforce MAC may be very rigorous, while the internal access control policy, used for need-to-know controls, may be less stringent. It would be preferable if these differences could be captured within a single high-level policy; we are investigating one such approach.

The other aspect relates to the assurance techniques involved in carrying out labelling. Currently most solutions aim to provide good trust at the release point, effectively the access control enforcement function. In some solutions, assurance is provided for the integrity of the data used in the access control decision, via the use of binding mechanisms, as discussed above, and in credential certificates for users. But there are other places within a security labelling framework where trust is required, most notably at the actual labelling point itself. Lack of assurance at this point is what results in the lower assurance of DAC over MAC, as discussed by Spalka et al (2000), but is equally problematic for an untrusted labelling system that can be exploited by Trojan processes.

## 7 Conclusion

Need-to-know policies are the basis for security labelling, and should be followed in the labelling process. This paper has proposed a lattice-based access control model for expressing such policies. The lattice-based model is not new, but we provide a formal technique for describing the kind of models that allow one to deal effectively with both security levels and categories within the security labelling process.

In this paper, we have also proposed a dynamic model for security labelling that supports the use of security categories with security classifications, as well as providing support for relabelling and label validation. This model provides a functional base for the design and implementation of a security labelling system.

Future work may include an extended discussion on the dynamic management of security policies and its implementation. The aim would be to discuss the combination of security labels and associated security mechanisms to achieve the required security goals. A further discussion on dynamic aspects and the implementation of assurance requirements associated with our model is needed; and a deep analysis of threats and attacks that directly affect security labels and labelling systems may also need to be considered.

## References

- Bell, D. E. & LaPadula, L. J. (1976), Secure Computer System: Unified exposition and Multics interpretation. MTR-2997, MITRE, Bedford, MA.
- Biba, K. J. (1977), Integrity Consideration for Secure Computer Systems. MTR-3153, The Mitre Corporation.
- Bidan, C. & Issarny, V. (1998), Dealing with multipolicy security in large open distributed systems. In *Proceedings of 5th European Symposium on Research in Computer Security*, pages 51–66.
- Dubuisson, O. (2000), ASN.1 Communication Between Heterogeneous Systems. Translated from French by Philippe Fouquart. Available from <http://www.oss.com/asn1/booksintro.html>.
- Ferraiolo, D. F., Kuhn, D. R. & Chandramouli, R. (2003), Role-Based Access Control. Artech House Inc.
- FIPS PUB 188. (1994), *Standard Security Label for Information Transfer*. Available from [www.itl.nist.gov/fipspubs/fip188.htm](http://www.itl.nist.gov/fipspubs/fip188.htm).
- Foley, S., Li, G. & Qian, X. (1996), A Security Model of Dynamic Labeling Providing a Tiered Approach to Verification. In *the IEEE Symposium on Security and Privacy*, pages 142–154, May 1996.
- Heintze, N. & Riecke, J. G. (1998), The slam calculus: Programming with secrecy and integrity. In *Proceedings of 25th ACM Symposium on Principles of Programming Languages (POPL)*, pages 365–377, San Diego, California.
- Housley, R. (1993), *Security Labeling Framework for the Internet*. Internet RFC 1457, May 1993.
- Internet CIPSO Working Group. (1993), *Common IP Security Option Version 2.3*. Internet Draft.
- ITU-T Recommendation X.841 (2000), *Information technology - Security Techniques - Security information objects for access control*.
- Liu, C. & Orgun, M. A. (2006), Towards security labelling. In V. Estivill-Castro and Gill Dobbie, editors, *Proceedings of the 29th Australasian Computer Science Conference, ACSC2006*, pages 69–76, Hobart, Australia. Australian Computer Society, Inc.
- Myers, A. C. & Liskov, B. (2000), Protecting privacy using the decentralized label model. *ACM Transactions on Software Engineering and Methodology*, 9(4):410–442.
- Sandhu, R. V. Lattice-based access control models. *IEEE Computer*, 26(11):9–19 (1993).
- Spalka, A., Cremers, A. B. & Lehmler, H. (2000), Protecting Confidentiality against Trojan Horse Programs in Discretionary Access System. In *Proc. Australasian Conference on Information Security and Privacy, ACISP 2000*, Brisbane, Australia, LNCS 1841:1-17, Springer.
- Weissman, C. (1969), Security control in the ADEPT-50 time-sharing system. In *AFIPS Conference Proceedings*, Vol. 35, pages 119-133. FJCC, 1969.
- Zdancewic, S. Zheng, L., Nystrom, N. & Myers, A.C. (2001), Untrusted hosts and confidentiality: Secure program partitioning. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 1–14, Banff, Canada.
- Zheng, L. & Myers, A.C. (2004), Dynamic security labels and noninterference. In *Proceedings of the 2nd International Workshop on Formal Aspects in Security and Trust (FAST)*, Toulouse, France.