Safety Analysis For The Extended Schematic Protection Model

P.E. Ammann R.S. Sandhu

Center for Secure Information Systems
and
Department of Information and Software Systems Engineering
George Mason University, Fairfax, VA 22030-4444

Abstract

Access control models provide a formalism and framework for specifying control over access to information and other resources in multi-user computer systems. Useful access control models must balance expressive power with the decidability and complexity of safety analysis, *i.e.* the determination of whether or not a given subject can ever acquire access to a given object. The access matrix model of Harrison, Ruzzo, and Ullman (HRU) has very broad expressive power. Unfortunately, HRU also has extremely weak safety properties; safety analysis is undecidable for most policies of practical interest. In this paper we show the remarkable result that an alternate formulation of HRU gives us strong safety properties. This alternate formulation is called the Extended Schematic Protection Model (ESPM). ESPM is derived from the Schematic Protection Model (SPM) by extending the creation operation to allow multiple parents for a child, as opposed to the conventional create operation of SPM which has a single parent for a child. It has earlier been shown that ESPM is equivalent in expressive power to HRU. Here we analyze the safety properties of ESPM. We show that, despite its equivalence to HRU, ESPM retains tractable safety analysis for a large class of protection schemes that are of practical interest.

1. Introduction, Background And Motivation

The need for access controls arises in any computer system that provides for controlled sharing of information and other resources among multiple users. Access control models (or protection models) provide a formalism and framework for specifying, analyzing and implementing security policies in multi-user systems. These models are typically defined in terms of the well-known abstractions of subjects, objects and access rights with which we assume the reader is familiar.

In this section we first give a brief review of access control models emphasizing the fact that the conventional black and white distinction between mandatory and discretionary access controls is inadequate. We then argue that models based on propagation of access rights actually transcend this distinction. This leads us to a discussion of the central topic of this paper, *viz.* the safety problem of determining whether or not a given subject can ever acquire access to a given object. Following this we give an outline of the principal contribution of this paper, which is the formal demonstration that very general expressive power and strong safety properties are simultaneously achieved by the Extended Schematic Protection Model (ESPM). This result is established and elaborated in the main body of the paper.

1.1. Access Control Models

The first access control models to be proposed [12, 17] were completely discretionary in that the creator of an object was vested with absolute freedom regarding who may or may not access the object. The vulnerability of such completely discretionary access controls (DAC) to Trojan Horse attacks is well known [11, for instance]. This vulnerability led to development of the so-called mandatory access controls (MAC) of the Bell and LaPadula model (BLP) [4]. Since then the MAC/DAC distinction has served as a basic principle for computer security. For instance it has been been embodied in the TCSEC [10] (popularly known as the Orange Book).

It recent times it has become increasingly clear that useful access control models must go beyond the traditional MAC/DAC distinction. Indeed opinion on this matter has changed so rapidly that what would have been considered heresy a few years ago is now being accepted without controversy. There are several major lines of argument that have together resulted in bringing about this rapid conversion of opinion. We enumerate these below.

(1) *Arguments based on secrecy policies.* As might be expected these arguments have come from the military sector [13, 24]. In the main they consist of the demonstration that there are document handling policies in the military---such as ORCON (originator control) and NOFORN (no foreign)---which cannot be readily expressed in BLP and are indeed not quite MAC or DAC in the conventional sense. A more abstract description of this need was given earlier by Millen [29].

(2) *Arguments based on integrity policies.* The black and white MAC/DAC distinction of BLP was carried over to integrity by Biba [5]. An early attempt by Lipner [20] to apply Biba's model showed the need for additional controls on program execution. Boebert and Kain [7] pointed out limitations of Biba's MAC and proposed the type enforcement controls of LOCK. Attempts by Lee [19] and Schockley [37] to implement the Clark and Wilson "commercial" integrity policy [8] within the framework of Biba demonstrated that either additional "mandatory" controls must be enforced or the policy must be emasculated by requiring certain aspects of it to be statically specified. Clark and Wilson [9] have described a notion of mandatory controls which is derived from their model. Collectively the papers cited here make a compelling case that the BLP and Biba notions of MAC are simply too limited for integrity policies.

(3) *Arguments based on a more general notion of MAC.* Sandhu [34, 36] has given alternate definitions of "mandatory" which show that the conventional BLP notion of MAC is but one special case of the general notion of access controls based on properties of subjects and objects rather than their identities. In the military non-disclosure context these properties turn out to be best expressed as partially ordered labels. In other contexts these properties are more naturally obtained in other ways. For instance the data type of an object determines what operations can be executed on that object. In [36], Sandhu argues that the real issue we need to focus on is whether or not the properties on which our "mandatory" access controls are based are static or dynamic. He also argues that attempts to define a notion of "mandatory" controls as something which lie between label-based mandatory and discretionary controls should be dropped in favor of definitions such as his which treat label-based mandatory controls as a special case of general "mandatory" controls. Abrams et al [1, 18] have also made similar arguments in their concept of a generalized framework for access control.

(4) *Arguments based on foundational inadequacies of BLP.* The foundational inadequacies of the BLP notion of mandatory controls were demonstrated in a series of papers by McLean [25, 26, 27, 28]. The thrust of McLean's argument is that if labels are considered to be changeable then the state-invariant properties of BLP can be preserved by changing labels which in effect automatically downgrades information on demand. He shows that the Basic Security Theorem of BLP can be proved for such an obviously insecure system. The main lesson from McLean's work for our purpose here is that we must attend to the dynamic aspects of access control.

We now draw attention to a line of research in access control models based on the idea of controlling the propagation of access rights. The basic concept is that the access-matrix is used not only to control access of subjects to objects but also to control the transport of access rights from one subject to another. The seminal work was that of Harrison, Ruzzo and Ullman [14] who described an access control model, commonly called HRU, in which complex policies for transport of access rights could be easily stated. The BLP model is known to be a particularly trivial case of HRU [30] as is the completely discretionary access matrix of [12, 17]. Therefore HRU does indeed transcend the traditional MAC/DAC distinction and could readily handle many of the concerns enumerated above. The problem with HRU however is that it has weak safety properties which we explain next. Before doing so we wish to emphasize and reiterate that our contribution in this paper is the demonstration of a model, *viz.* ESPM, which is formally equivalent in expressive power to HRU and yet has strong safety properties.

1.2. The Safety Problem

As noted above, access control models not only specify the control of access rights to resources but also specify the propagation of such rights. The safety problem for access control models is the determination of whether or not a given subject can ever acquire access to a given resource. Thus protection models must satisfy two conflicting requirements:

(1) The need for expressive power sufficient to conveniently describe security policies of practical interest.

(2) The need for tractable analysis of the safety problem.

1.2.1. Weak Safety Properties Of HRU

The most general access control model, the access matrix model of Harrison, Ruzzo, and Ullman [14] (HRU), has broad expressive power; unfortunately, it also has weak safety properties. Harrison and Ruzzo [15] proved the following demarcation for the decidability of safety analysis in HRU:

(1) Safety is undecidable for *bi-conditional* schemes; i.e., the condition part of every command has at most two terms.

(2) Safety is decidable for *mono-conditional* schemes; i.e., the condition part of every command has at most one term.†

Mono-conditional systems are very restrictive and can accommodate only the simplest security policies. At the same time, models such as take-grant [16], which require bi-conditional commands, do have efficient safety analysis. Thus the demarcation of decidable safety in HRU is too pessimistic.

A restriction on expressive power that can have substantial benefits for safety analysis is that of monotonicity; monotonic models do not allow the deletion of access privileges. It must be noted that a strictly monotonic model is too restrictive to be of much practical use, since the ability to delete access privileges is an important requirement. We are really interested in models which can be reduced to monotonic models for purpose of safety analysis. In particular, we can ignore deletion of an access privilege P whenever the deletion can itself be undone by regranting P. This is by far the most common form of revocation and it is indeed fortunate that monotonic models can accommodate such deletion.

Since monotonic models do not permit the deletion of access privileges, backtracking in analysis can be avoided. However, monotonicity by itself is insufficient for tractable safety analysis. The monotonic version of the access matrix model of Harrison, Ruzzo, and Ullman [14] (HRU) retains broad expressive power; unfortunately, despite its monotonicity, it also retains the weak demarcation of safety cited above. Safety analysis remains undecidable even for monotonic bi-conditional HRU schemes [15].

† This result has only been shown for monotonic schemes and for nonmonotonic schemes with "Delete" but not "Destroy". The decidability for the general nonmonotonic case with "Destroy" remains open [15].

1.2.2. Strong Safety Properties Of SPM

In response to the relatively weak safety properties of HRU, a variety of models‡ with more desirable safety properties have been proposed [6, 16, 21, 22, 23]. However, a substantial gap in expressive power exists between these models and HRU. Sandhu's Schematic Protection Model (SPM) [31, 32] was developed to fill this gap in expressive power while sustaining efficient safety analysis. The various models referenced above are all subsumed by SPM. SPM has remarkably strong safety properties and has been shown to represent a wide variety of cases of practical interest.

Despite SPM's expressive power, attempts to demonstrate the equivalence of SPM to HRU have so far failed. There is now strong reason to believe that SPM is in fact less expressive than HRU [2, 3], although formal demonstration of this fact remains an important open question.

Accordingly, SPM has been extended. The single parent creation operation in SPM has been redefined to allow multiple parents for a child. With the joint creation operation, the new model, called ESPM, has been shown to be precisely equivalent to monotonic HRU [3].

The question naturally arises as how safety analysis is affected by the increase in expressive power provided by the joint creation operation. In this paper, we present a safety analysis algorithm for a restricted subclass of ESPM schemes, namely those schemes in which the create structure is acyclic with the possible exception of attenuating loops (particular kinds of cycles of length one). The algorithm given here is based upon the SPM algorithm presented in [31]. However, the machinery of [31] is insufficient for ESPM schemes. Here we demonstrate that the inclusion of a joint creation operation in the ESPM model still permits tractable safety analysis for many cases of practical interest. We also offer guidelines for how to employ the joint creation operation so as to minimize the computational effort of the safety analysis.

‡ These models are monotonic in the sense describe above. They generally include the kind of revocation in which the revocation itself can be undone. Thus they are monotonic only in the technical sense of being reducible to monotonic operations for purpose of safety analysis.

1.3. Outline of the Paper

The organization of the rest of this paper is as follows. In section 2 we describe ESPM, an extension of SPM that has been shown equivalent in expressive power to monotonic HRU [2, 3]. In section 3, we present the our main result, which is a safety analysis algorithm for ESPM schemes with acyclic attenuating loops. The computational complexity of analyzing such schemes is given, and it is shown that the complexity is feasible for cases of practical interest. In section 4, we compare the safety results for ESPM and HRU. Section 5 concludes the paper.

2. ESPM

In this section we define ESPM (Extended Schematic Protection Model). The description of ESPM here is of necessity terse and formal. Motivation of the various components of the model is given in [3, 31].

ESPM is derived from SPM (Schematic Protection Model) by extending the creation operation to allow multiple parents for a child, as opposed to the conventional create operation of SPM which has a single parent for a child. This is the only difference between SPM and ESPM. For convenience, here we have chosen to define ESPM directly rather than first defining SPM and then defining ESPM as an extension.

ESPM is based on the key principle of protection types, henceforth abbreviated as types. ESPM subjects and objects are strongly typed, *i.e.*, the type of an entity (subject or object) is determined when the entity is created and does not change thereafter. Types are an abstraction of the intuitive notion of properties that are security relevant. An ESPM scheme is to a large extent, but not exclusively, defined in terms of types. The dynamic privileges in ESPM are tickets of the form Y/r where Y identifies some unique entity and r is a right. The notion of type is extended to tickets by defining type(Y/r) to be the ordered pair type$(Y)/r$. That is the type of a ticket is determined by the type of entity it addresses and the right symbol it carries.

ESPM has only two operations for changing the protection state, viz., create and copy.† These operations are authorized by rules which comprise the scheme defined by specifying the following (finite) components.

† We note that the original definition of SPM [31] included a third operation called demand that has since been shown to be redundant [33].

(1) Disjoint sets of subject types TS and object types TO. Let $T = TS \cup TO$.

(2) A set of rights R. The set of ticket types is thereby $T \times R$.

(3) A can-create function:

$$cc: TS \times TS \times \cdots \times TS \rightarrow 2^T$$

(4) Create rules of the form:

$$cr_{p_i}(u_1, u_2, \ldots, u_N, v) = c/R_1^i \cup p_i/R_2^i \text{ for } i = 1..N$$

$$cr_c(u_1, u_2, \ldots, u_N, v) = c/R_3 \cup p_1/R_4^1 \cup p_2/R_4^2 \cup \cdots \cup p_N/R_4^N.$$

where p_i is the *ith* parent and c is the child.

(5) A collection of link predicates $\{link_i\}$

(6) A filter function $f_i : TS \times TS \rightarrow 2^{T \times R}$ for each predicate $link_i$.

An ESPM scheme is itself static and does not change. We now explain how the scheme controls and regulates the propagation and creation of access rights.

The Create Operation

Creation is authorized exclusively by types. Subjects of type $u_1, u_2, \ldots, u_N$ can (jointly) create entities of type v if and only if $v \in cc(u_1, u_2, \ldots, u_N)$. N may take on any positive value, although for any given scheme this value is of course bounded. The case of $N=1$ corresponds to the conventional creation operation in SPM. The case of $N>1$ makes ESPM different from SPM by authorizing multiple parents to jointly and cooperatively create a child subject or object. Note that, if type constraints are met, we allow a subject to participate as more than one parent in a joint create operation.

Tickets introduced as the side effect of creation are specified by create-rules. In the create rules c is the name of the jointly created entity and p_i is the name of the *ith* parent. The sets R_1^i, R_2^i, R_3, and R_4^i, for $i = 1..N$ are subsets of R. When subjects $U_1, U_2, \cdots U_N$ of type $u_1, u_2, \cdots u_N$ create entity V of type v the parent U_i gets the tickets V/R_1^i and U_i/R_2^i as specified by cr_{p_i}. The child V similarly gets the tickets V/R_3 and U_i/R_4^i for each parent U_i as specified by cr_c. As an example, consider the single parent creation case in which *file* $\in$ *cc(user)* authorizes *users* to creates *files*; cr_p*(user, file)* $= c/rw$ and cr_c*(user, file)* $= \emptyset$ gives the creator r and w tickets for the created file. Note that the superscript i is used to specify a (potentially) different set for each of the N parents. Also note that the parents are not allowed to directly exchange tickets with other parents as a result of

creation; the copy operation is required to do this.

The Copy Operation

A copy of a ticket can be transferred from one subject to another leaving the original ticket intact. Permission to copy a ticket Y/r depends in part on possession of the ESPM copy flag, c, for that ticket, denoted Y/rc. Possession of Y/rc implies possession of Y/r but not vice versa. It is possible to copy Y/r only, or to copy Y/rc, in which case the ticket may be further copied. Let $dom(U)$ signify the set of tickets possessed by U. Three independent pieces of authorization are required to copy Y/r from U to V.

(1) $Y/rc \in dom(U)$, *i.e.*, U must possess Y/rc for copying either Y/rc or Y/r.

(2) There is a link from U to V. Links are established by tickets for U and V in the domains of U and V. The predicate $link_i(U, V)$ is defined as a conjunction or disjunction, but not negation, of one or more of the following terms for any $r \in R$: $U/r \in dom(U)$, $U/r \in dom(V)$, $V/r \in dom(U)$, $V/r \in dom(V)$, and **true**. Some examples from the literature are given below [21, 23, 32, respectively]:

$$link_{tg}(U, V) \equiv V/g \in dom(U) \vee U/t \in dom(V)$$
$$link_{t}(U, V) \equiv U/t \in dom(V)$$
$$link_{sr}(U, V) \equiv V/s \in dom(U) \wedge U/r \in dom(V)$$
$$link_{u}(U, V) \equiv \textbf{true}$$

(3) The final condition is defined by the filter functions f_i, one per predicate $link_i$. The value of $f_i(u, v)$ specifies types of tickets that may be copied from subjects of type u to subjects of type v over $link_i$. Also f_i determines whether or not the copied ticket can have the copy flag. Example values are $T \times R$, $TO \times R$, and $\emptyset$ respectively authorizing all tickets, object tickets and no tickets to be copied via a $link_i$.

In short Y/r can be copied from U to V if and only if there exists some $link_i$ such that:

$$Y/rc \in dom(U) \wedge link_i(U, V) \wedge y/r \in f_i(u, v)$$

where the types of U, V and Y are respectively u, v and y. To copy Y/rc from U to V, it must also be the case that $y/rc \in f_i(u, v)$.

3. Safety Analysis Of ESPM

The fact that safety analysis is undecidable for *arbitrary* ESPM schemes is immediate from either of the following two observations:

(1) ESPM is equivalent in expressive power to HRU [2, 3], and safety is undecidable for HRU [14, 15].

(2) ESPM is a generalization of SPM, and safety is known to be undecidable for arbitrary SPM schemes [35].

However, relatively minor restrictions (from a practical point of view) on ESPM schemes permit decidable and indeed tractable safety analysis. As in SPM [31], the tractability of safety analysis for ESPM given here turns out to be determined by the creation operation.

The safety analysis of ESPM is modeled on Sandhu's treatment of SPM in [31]. There are significant differences in the details. In this section we give an outline of how the results of [31] can be modified to apply to ESPM. Section 3.1 reviews the general strategy of unfolding and the canonical and maximal states employed in [31]. Section 3.2 defines the analog of SPM's acyclic attenuating create operations in ESPM. This is a straightforward generalization. Section 3.3 defines the analog of SPM's surrogate function in ESPM which we call the *ID* function. This requires some care because of the multi-parent creation in ESPM. Section 3.4 defines the construction of a canonical state in ESPM by unfolding. Unlike in SPM, we have to be very careful regarding the sequence of creates to ensure that the unfolding process will terminate for acyclic attenuating creates. Section 3.5 sketches a proof of correctness for this construction. Section 3.6 discusses the complexity of safety analysis, including some observations on how the complexity can be kept manageable in practice.

3.1. Safety Analysis from the Maximal State

In his safety analysis for SPM, Sandhu first shows that we can assume without loss of generality that all create operations occur before any copy operations [31, lemma 12]. This property also clearly applies to ESPM, and will in fact be true for many monotonic models.† This fact motivates the following strategy for safety analysis.

(1) Starting with the given initial state, first create as many subjects as are necessary to account for the worst-case behavior of the system with respect to propagation of access rights. Call this the *canonical state* of the system.

† The property that all create operation operations may be considered to occur first holds for monotonic models in which creation is statically authorized. For example, if creation depends upon possession of certain tickets, then the property no longer holds.

(2) Given the canonical state, perform all copy operations until the state does not change any further. Call this the *maximal state* of the system.

(3) A specific safety question such as, "Can subject X obtain right r for object Y?" is then answered by looking at the maximal state and seeing whether or not X actually possesses the right r for Y in this worst-case state.

The second step in this procedure is guaranteed to terminate because the canonical state has a finite number of subjects, objects and rights and therefore the copy operations will eventually be unable to propagate any new privileges. The problem lies in the first step where we need some criteria to determine when all the necessary create operations have occurred. In other words we need to be able to recognize a canonical state. The undecidability result of [35] shows that it is impossible in general to recognize a canonical state. However, there are reasonable restrictions on the can-create function of SPM which make construction of the canonical state trivial [31]. In particular if can-create has no cycles or only has the so-called attenuating cycles of length one the canonical state can be constructed by an operation called *unfolding*. These restrictions on can-create are eminently reasonable as evidenced by the fact that no practical policy to date has required non-attenuating cycles in can-create [31, 32].

3.2. Restrictions on Joint Create Operations

In the safety analysis given for SPM in [31], the create function is restricted to be acyclic except for certain cycles of length one, or loops. In other words, the *cc* function partially orders the types. We define the *create graph* to be the directed graph with types as nodes and edges defined by *cc*. The loops that are allowed are for create operations with *attenuating* create rules. Attenuating create rules specify that for those tickets acquired as a result of a create operation, the tickets acquired by the child are a subset of those acquired by the parent. The idea is that the parent can then simulate any possible action of the child, and thus the creation of the child may be ignored.

For the analysis of the joint create operation in ESPM we similarly restrict the multiple parent create function to be acyclic except for attenuating create loops. Here the create graph is a hypergraph; each directed edge originates at a set of nodes. The restriction that at least one parent in a multiple-parent, loop create operation must be able to simulate the child effectively means that, in general, child tickets may not be distributed to either the child or its parents, and that no parent tickets may be distributed to the child. Thus for creation operations in which the type v of the child is the same as the type u_i of one or more parents, we have the general rule:

$$cr_{p_i}(u_1, u_2, ..., u_N, v) = p_i/R_2^i \text{ for } i = 1..N$$

$$cr_c(u_1, u_2, ..., u_N, v) = \emptyset .$$

Note that a parent can only use an attenuating loop create operation to increase the set of tickets that it holds for itself.

However, the general rule given above is too pessimistic in two important situations. First, for single parent creation, these rules are relaxed, as was done in [31], so that the parent's tickets need only be a superset of the child's tickets. Second, for multi-parent creation, *one* of the parents with type matching the child may be treated in a similar manner; the child may receive tickets for itself and that one parent if that parent receives a superset of these tickets.† If multiple parents in an attenuating loop creation were allowed to receive tickets for the child, no single parent would be able to simulate the actions of the child. For this reason, attenuating create rules do not allow such a situation.

3.3. The ID Function

To realize the strategy of section 3.1 we need to provide a sufficiently rich set of canonical entities and show how to map actual ESPM entities onto canonical entities. In [31], a special function, called the *surrogate* function, is introduced to provide correspondence between canonical SPM entities and entities in arbitrary SPM histories. The surrogate function turns out to be inadequate for ESPM schemes because, except for *individual* entities in the initial state, the surrogate function is based strictly on the notion of type. For joint create, we need to capture the notion of grouping entities together. We therefore define a function, which we call the *ID* function, which assigns a name or identification to every ESPM entity.

The *ID* function is recursively defined below. Consider an entity V of type v. If V is not in the initial state, it is assumed to have parent(s) $U_1 .. U_N$ of types $u_1..u_N$. There are three cases to consider:

(1) If V is in the initial state then $ID(V) = V$. The *ID* of an entity in the initial state is simply the name

† Without loss of generality, we may assume that the parent which is allowed to receive tickets for the child is the first appropriately-typed parent listed in the parameters of the can-create function. The assumption simplifies the *ID* function's definition, given below.

of that entity.

(2) If $v \neq u_i$ for all $i = 1..N$ then $ID(V) = C_v(ID(U_1), \ldots, ID(U_N))$. The *ID* of an entity that is strictly below all of its parents in the create/joint-create graph is simply a grouping of the *ID*'s of the child's parents. The group is tagged with the child's type. Note that since there are a finite number of types, we may effectively regard the C_v as constants.

(3) If $v = u_i$ for some $i = 1..N$ then $ID(V) = ID(U_i)$ for the first i for which $v = u_i$. Since any child produced by an attenuating create rule can be simulated by at least one of the parents, the *ID* of an entity that has been created with an attenuating loop create rule is the *ID* of a parent of the matching type. Since there may be multiple parents with the same type as the child, we simply define the *ID* function to map to the first such parent.

We define one canonical entity for each element in the range of the *ID* function. Thus questions about the correspondence between ESPM entities and canonical entities reduce to questions about the *ID* function. Our first task is to show that there are a finite number of canonical entities for acyclic attenuating schemes, *i.e.* that the range of *ID* is finite:

Lemma 1 Given any acyclic attenuating ESPM scheme, the range of the *ID* function is a finite set.

Proof Consider each of the three cases. Clearly, case 1 presents no difficulty since the case represents the base case of the recursion and there are a finite number of entities in the initial state. In a depth-first evaluation, Case 2 cannot be applied more often than there are types in the ESPM scheme due to the acyclic structure of the create graph. Since the tree structure introduced by case 2 has a bounded number of children at each node and a finite depth, the number of entries in the tree must be finite. Finally, case 3 does not alter the value of the ID function, so it may be invoked an arbitrary number of times without affecting the function's value. Since cases 1 and 2 are the only rules that can be used to generate distinct names, and since each case can only be applied a finite number of times, the range of the *ID* function is finite.

3.4. The Unfolding Algorithm

Our goal is to unfold an initial state and have the result be the complete collection of canonical entities, one entity for each element in the range of the *ID* function. As a first step, we define cc', to be the acyclic part of the function cc. Now, if we simply apply arbitrary rules from cc' first to entities in an initial state and then to entities in each resulting state, it is not at all clear whether the unfolding process ever terminates. However, there is an order for applying the creation rules in cc' such that the unfolding process is guaranteed to terminate. The order depends upon the type of entity produced by a create operation, but not upon the parent type(s).

To develop the correct ordering, we reformulate the can-create function cc' as a relation. For each tuple of types $(u_1, u_2, ..., u_N)$ in the domain of cc', we replace the mapping $cc'(u_1, u_2, ..., u_N) = S$, where $S = \{s_1, s_2, ..., s_M\}$ is a subset of T, with the relation, $\{((u_1, u_2, ..., u_N), s_1), ((u_1, u_2, ..., u_N), s_2), ..., ((u_1, u_2, ..., u_N), s_M)\}$. Note that in each ordered pair in the relation, the abscissa, or first element, is the tuple of parent types and the ordinate, or second element, is a single child type. We refer to an ordered pair in this relation as a create-tuple.

Now we (partially) order the create-tuples based on the ordinate (*ie* the child type). That is, for every pair of types s and t, if s precedes t in the create graph, then every create-tuple with s as an ordinate must precede every create-tuple with t as an ordinate. We may then make the key observation, which is guaranteed by the acyclic structure of cc', that in the ordered list of create-tuples, every create-tuple that employs type t as a parent follows all of the create-tuples that produce type t as a child. The observation allows us to consider a create-tuple once during the unfolding process and be sure that the create-tuple will not be subsequently "re-enabled" with a new set of parents as a result of some later creation operation.

We build the canonical state as follows: First, we put the entities from the initial state into the canonical state. Next, we proceed down the ordered list of create-tuples and apply each create-tuple once to each possible tuple of parent entities in the canonical state. The resulting entity from each application of a create-tuple is placed in the canonical state. This process is illustrated with an example in fig. 1. In fig. 1, entities are represented by subscripted upper case letters; the type of a given entity is the same letter in lower case.

Lemma 2 The unfolding operation terminates.

Proof That the unfolding procedure terminates can be seen by noting that no application of a particular create-tuple can result in either that create-tuple, or any other create-tuple

Types:
$T = \{x, y, z\}$

Create Rules:
$cc(x, y) = \{y, z\}$
$cc(x) = \{x, y\}$

Acyclic Portion Of Create Rules:
$cc'(x, y) = z$
$cc'(x) = y$

Ordered Create Tuples from cc'
$((x), y)$
$((x, y), z)$

Initial State
$\{X_1, X_2, Y_1\}$

Canonical State and *ID* function values:

X_1	$ID(X_1) = X_1$
X_2	$ID(X_2) = X_2$
Y_1	$ID(Y_1) = Y_1$
Y_2	$ID(Y_2) = C_y(ID(X_1)) = C_y(X_1)$
Y_3	$ID(Y_3) = C_y(ID(X_2)) = C_y(X_2)$
Z_1	$ID(Z_1) = C_z(ID(X_1), ID(Y_1)) = C_z(X_1, Y_1)$
Z_2	$ID(Z_2) = C_z(ID(X_1), ID(Y_2)) = C_z(X_1, C_y(X_1))$
Z_3	$ID(Z_3) = C_z(ID(X_1), ID(Y_3)) = C_z(X_1, C_y(X_2))$
Z_4	$ID(Z_4) = C_z(ID(X_2), ID(Y_1)) = C_z(X_2, Y_1)$
Z_5	$ID(Z_5) = C_z(ID(X_2), ID(Y_2)) = C_z(X_2, C_y(X_1))$
Z_6	$ID(Z_6) = C_z(ID(X_2), ID(Y_3)) = C_z(X_2, C_y(X_2))$

Fig. 1. Example of Unfolding An Initial State.

considered before it, being applicable to a previously unconsidered tuple of entities. Thus for each create-tuple, there are a fixed number of applications possible. Since each create-tuple is considered only once, the procedure terminates.

Lemma 3 The *ID* function assumes a unique value for each entity in the unfolded state.

Proof The proof proceeds by induction. Clearly the property holds in the initial state. Assume that the property holds at some arbitrary point in the unfolding and consider the application of the current create-tuple. The *ID* function for the new entity must be acquired by applying case 2 of the definition of the *ID* function. That the resulting *ID* is unique follows from the fact that the *ID*s of all existing canonical entities are unique and the fact that each create-tuple is being applied only once to each unique tuple of parents.

Finally, we consider the attenuating loop portion of *cc*. We allow a single application of each attenuating rule to each possible entity or set of entities in the unfolded state. The resulting entities are *not* placed into the canonical state, since the *ID*s for these entities are already present. The effect is to supply each canonical parent with all tickets that can be acquired as a result of creation.

3.5. Proof of Correctness

To complete our analysis we need to show that

(1) From the safety perspective, an arbitrary sequence of acyclic attenuating ESPM create and copy operations can be subsumed by copy operations only on the canonical state.

(2) Safety questions in the canonical state can be answered in polynomial time.

Both of these question are dealt with for SPM in [31]; the first question is theorem 17, and the second is discussed in the text immediately following corollary 18. We have carefully arranged our analysis here such that if the surrogate function from [31] is replaced with the *ID* function, the analysis for the two questions listed above can be taken directly from [31]. A complete formal proof of these results is much too long for the scope of this paper and may be found in [2].

3.6. Complexity of Safety Analysis

We now present the safety algorithm's cost in computational complexity and offer guidelines on how to minimize costs in practical applications. As is the case for SPM, the time required to construct the canonical state for ESPM is exponential in the number of types if the *cc* relation is dense. This may not be a serious obstacle if the scheme employs relatively few types.

On the other hand, the required time is also multiply exponential in the number of parents of the joint creation operator. The reason is that each application of an N-parent create operation that produces a specific child type, v, results in on the order of I^N new entities being added to the canonical state, where I is the number of entities in the canonical state before any entities of type v are produced. For each case in which the child of a joint create can participate as a parent in another joint create (excluding attenuating loops) this expansion is repeated.

Let x be the number of create-tuples derived from cc', N_i, $i = 1..x$ be the number of parents in the creation operation corresponding to the *ith* create-tuple, and I be the size of the initial state. Note that N_i may well equal 1, which corresponds to using the SPM create operation. The application of the first create-tuple results in a canonical state whose size, I_1, is on the order of $I_1 = O(I + I^{N_1})$. The application of the second create-tuple results in $I_2 = O(I_1 + I_1^{N_2})$. This process continues up to the complete canonical state I_x, $I_x = O(I_{x-1} + I_{x-1}^{N_x})$.

Clearly, the joint create operation needs to be used with great care to keep the analysis feasible. Some simple rules are:

(1) Use single parent creation where possible.

(2) Do not use a value for N that is any larger than necessary, *ie* keep the number of parents in joint creates as small as possible.

(3) Minimize the opportunities for descendants of a joint creation operation to participate in further creation operations.

(4) Avoid dense *cc* functions.

These appear to be reasonable guidelines which can be easily achieved in practice.

4. Expressive Power Of ESPM

In this section we discuss results regarding the expressive power of ESPM from both the formal and pragmatic aspects.

4.1. Formal Equivalence Of ESPM And HRU

With respect to the formal expressive power of ESPM, it has been shown that ESPM is precisely equivalent in expressive power to the monotonic case of HRU [2, 3]. Since monotonic HRU is the most general monotonic protection model to date, the theoretical expressive power of ESPM is thereby the most general known thus far. Equivalence of ESPM and HRU is established by simulating monotonic HRU in ESPM and vice versa. The simulation of HRU in ESPM is by far the more difficult part of this proof. Since the complete details are lengthy [2], we briefly sketch below just the key role played by the joint creation operation.

An ESPM simulation of monotonic HRU is required to account for all of the various operations involved in executing an arbitrary HRU command. It turns out that there is no particular difficulty in simulating most of these operations, even though the details of the simulations are quite intricate. Simulating the evaluation of an HRU conditional, the creation of a new HRU entity, and the entering of a right into a cell in the access matrix all turn out to be fairly straight forward, *provided* that one has access to exactly those HRU entities that are participating in the particular invocation of the HRU command.

It is the grouping of HRU entities into the parameter lists for HRU commands that appears to lie beyond the expressive power of SPM. On a particular HRU command invocation, it is necessary to consider a specific subset of HRU entities, and to exclude from consideration all other HRU entities. On a subsequent HRU command invocation, it is necessary to consider a different subset of HRU entities, and so on. The single parent creation operation of SPM is not suited this task, nor, apparently, is any other SPM feature. However, the joint creation operation of ESPM is ideal. In the simulation in [2], the joint creation operation is used with each (existing) entity in the HRU parameter list participating as a parent. The child entity of this creation is given a ticket to identify each parent and its position in the parameter list. These tickets enable the child to oversee the simulation of the various parts of an HRU command.

Several points warrant notice. First, something like a joint creation capability seems to be necessary to achieve the expressive power of monotonic HRU. SPM by itself appears to be less expressive than HRU. Second, since the create structure used in the construction is cyclic (i.e., entities of type p can indirectly create other type p entities), the safety of the scheme is outside the cases known to be decidable [31, 35]; this characteristic is consistent with the weak safety properties of HRU. Finally, the construction is not the most natural way to implement policies. Due to the general nature of the construction, even simple policies, such as Take/Grant, are transformed into lengthy schemes. For instance, a straightforward implementation of Take/Grant in SPM requires only two link predicates [31]. However, defining Take/Grant in HRU, and then applying the construction outlined in [2] results in over twenty link predicates.

4.2. Pragmatic Expressive Power Of ESPM

Not only does ESPM have the theoretical expressive power discussed above but it also has a natural expression of many practical policies which have been published in the literature. A great deal of evidence for this assertion comes from the known expressive power of SPM in this regard. It has been shown [32] that SPM subsumes several well-known protection models, including BLP [4] and take/grant [16, 21] as special cases. Moreover SPM subsumes these models within its

decidable cases for safety analysis. Therefore SPM subsumes these models not only in terms of its expressive power but also in terms of safety analysis. This is in sharp contrast to HRU, which does subsume these models but outside its known decidable classes for safety. The joint creation capability of ESPM in addition provides significant pragmatic benefit in solving a variety of well-known security problems such as mutual suspicion, protected subsystems, confinement and separation of duties [3]. These problems have very natural and intuitive solutions based on joint creation.

5. Conclusion

In this paper, we have enumerated a compelling list of arguments as to why the MAC/DAC framework for defining security policies is inadequate. The arguments are based on the requirements for secrecy and integrity policies, the lack of generality of the MAC framework, and the foundational weaknesses of the Bell LaPadula model. More general access control models can address these arguments and include MAC and DAC as special cases.

We have argued that models based on propagation of access rights already transcend the MAC/DAC distinction. The challenge with access control models is to provide adequate expressive power without sacrificing safety analysis. To date, models with broad expressive power, *eg* HRU, exhibit weak safety properties, and models with desirable safety properties exhibit less expressive power than HRU. The most expressive model to date with strong safety properties is Sandhu's Schematic Protection Model (SPM). However, its relative power with respect to monotonic HRU remains an open question. We now conjecture that SPM is actually less expressive than HRU.

We have demonstrated that by extending SPM with a joint creation operation, the resulting model, called ESPM, simultaneously enjoys the expressive power of monotonic HRU and yet retains efficient safety analysis for protection schemes of practical interest. We have presented a safety analysis algorithm for ESPM schemes with acyclic attenuating loops in the create structure, and we have given the computational complexity of the safety algorithm. ESPM is therefore in effect an alternate formulation of HRU with strong safety properties.

References

[1] Abrams, M., Eggers, K., LaPadula, L., Olson I., "A Generalized Framework for Access Control: An Informal Description", *Proceedings Thirteenth National Computer Security Conference*, Washington, DC, October, 1990.

[2] Ammann, P.E. and Sandhu, R.S., "The Extended Schematic Protection Model", Technical Report, George Mason University, 1990.

[3] Ammann, P.E. and Sandhu, R.S., "Extending the Creation Operation in the Schematic Protection Model", *Proceedings Sixth Annual Computer Security Application Conference*, Tucson, AZ (1990).

[4] Bell, D.E. and LaPadula, L.J., "Secure Computer Systems: Unified Exposition and Multics Interpretation", *Mitre Technical Report MTR-2997*, Bedford, MA (1975).

[5] Biba, K.J., "Integrity Considerations for Secure Computer Systems", *Mitre Technical Report MTR-3153*, Bedford, MA (1977).

[6] Bishop, M. and Snyder, L., "The Transfer of Information and Authority in a Protection System", *7th ACM Symposium on Operating Systems Principles*, 45-54 (1979).

[7] Boebert, W.E. and Kain, R.Y., "A Practical Alternative to Hierarchical Integrity Policies", *Proceedings Eighth National Computer Security Conference*, 18-27 (1985).

[8] Clark, D.D. and Wilson, D.R., "A Comparison of Commercial and Military Computer Security Policies", *Proceedings 1987 IEEE Symposium on Security and Privacy*, Oakland, CA, May, 1987.

[9] Clark, D.D. and Wilson, D.R., "Evolution of a Model for Computer Integrity", *Report of the Invitational Workshop on Data Integrity*, NIST Special Publication 500-168, (1989).

[10] Department of Defense National Computer Security Center, *Department of Defense Trusted Computer Systems Evaluation Criteria*, DoD 5200.28-STD, (1985).

[11] Downs, D.D., Rub, J.R., Kung, K.C. and Jordan, C.S., "Issues in Discretionary Access Control", *Proceedings 1985 IEEE Symposium on Security and Privacy*, Oakland, CA, May, 1985.

[12] Graham, G.S. and Denning, P.J., "Protection - Principles and Practice", *AFIPS Spring Joint Computer Conference*, 40:417-429 (1972).

[13] Graubart, R.D., "On the Need for a Third Form of Access Control", *Proceedings Twelfth National Computer Security Conference*", Baltimore, MD, October, 1989.

[14] Harrison, M.H., Ruzzo, W.L. and Ullman, J.D., "Protection in Operating Systems", *CACM*, 19(8):461-471 (1976).

[15] Harrison, M.H. and Ruzzo, W.L., "Monotonic Protection Systems", In *Foundations of Secure Computations*, DeMillo, R.A., Dobkin, D.P., Jones, A.K. and Lipton, R.J. (Editors), Academic Press (1978).

[16] Jones, A.K., Lipton, R.J. and Snyder, L., "A Linear Time Algorithm for Deciding Security", *17th IEEE Symposium on the Foundations of Computer Science*, 337-366 (1976).

[17] Lampson, B.W., "Protection", *5th Princeton Symposium on Information Science and Systems*, 437-443 (1971). Reprinted in *ACM Operating Systems Review*, 8(1):18-24 (1974).

[18] LaPadula, L.J., "Formal Modeling in a Generalized Framework for Access Control", *Computer Security Foundations Workshop*, 100-109 (1990).

[19] Lee, T.M.P., "Using Mandatory Integrity to Enforce "Commercial" Security", *Proceedings 1988 IEEE Symposium on Security and Privacy*, Oakland, CA, May, 1988.

[20] Lipner, S.B., "Non-discretionary Controls for Commercial Applications", *Proceedings 1982 IEEE Symposium on Security and Privacy*, Oakland, CA, April, 1982.

[21] Lipton, R.J. and Snyder, L., "A Linear Time Algorithm for Deciding Subject Security", *JACM*, 24(3):455-464 (1977).

[22] Lipton, R.J. and Budd, T.A., "On Classes of Protection Systems", In *Foundations of Secure Computations*, DeMillo, R.A., Dobkin, D.P., Jones, A.K. and Lipton, R.J. (Editors), Academic Press (1978).

[23] Lockman, A. and Minsky, N., "Unidirectional Transport of Rights and Take-Grant Control", *IEEE Transactions on Software Engineering*, SE-8(6):597-604 (1982).

[24] McCollum, C.J., Messing, J.R. and Notargiacomo, L., "Beyond the Pale of MAC and DAC - Defining New Forms of Access Control", *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May, 1990.

[25] McLean, J., "A Comment on the 'Basic Security Theorem' of Bell and LaPadula", *Information Processing Letters*, 20(2):67-70 (1985).

[26] McLean, J., "Reasoning About Security Models", *Proceedings 1987 IEEE Symposium on Security and Privacy*, Oakland, CA, May, 1987.

[27] McLean, J., "The Algebra of Security", *Proceedings 1988 IEEE Symposium on Security and Privacy*, Oakland, CA, May, 1988.

[28] McLean, J., "Specifying and Modeling Computer Security", *IEEE Computer*, 23(1):9-16 (1990).

[29] Millen, J.K., "A1 Policy Modeling", *Proceedings Seventh National Computer Security Conference*, 137-145 (1984).

[30] Pittelli, P., "The Bell-LaPadula Computer Security Model Represented as a Special Case of the Harrison-Ruzzo-Ullman Model", *Proceedings Tenth National Computer Security Conference*, Baltimore, MD, October, 1987.

[31] Sandhu, R.S., "The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes", *JACM*, 35(2):404-432 (1988).

[32] Sandhu, R.S., "Expressive Power of the Schematic Protection Model", *Computer Security Foundations Workshop*, 188-193 (1988).

[33] Sandhu, R.S., "The Demand Operation in the Schematic Protection Model", *Information Processing Letters*, 32(4):213-219 (1989).

[34] Sandhu, R.S., "Terminology, Criteria and System Architectures for Data Integrity", *Report of the Invitational Workshop on Data Integrity*, NIST Special Publication 500-168, (1989).

[35] Sandhu, R.S., "Undecidability of the Safety Problem for the Schematic Protection Model with Cyclic Creates", *JCSS*, to appear.

[36] Sandhu, R.S., "Mandatory Controls for Database Integrity", in *Database Security, III: Status and Prospects*, Spooner, D.L. and Landwehr, C., editors, Elsevier Science Publishers B.V. (North Holland), (1990).

[37] Shockley, W.R., "Implementing the Clark/Wilson Integrity Policy Using Current Technology", *Proceedings Eleventh National Computer Security Conference*, Baltimore, MD, October, 1988.