An Attribute-Based Access Matrix Model

Xinwen Zhang
Lab for Information Security Technology
George Mason University
xzhang6@gmu.edu

Yingjiu Li
School of Information Systems
Singapore Management University
yjli@smu.edu.sg

Divya Nalla
School of Information Systems
Singapore Management University
divyanalla@smu.edu.sg

ABSTRACT

In traditional access control models like MAC, DAC, and RBAC, authorization decisions are determined according to identities of subjects and objects, which are authenticated by a system completely. Modern access control practices, such as DRM, trust management, and usage control, require flexible authorization policies. In such systems, a subject may be only partially authenticated according to one or more attributes. In this paper we propose an attribute-based access matrix model, named ABAM, which extends the access matrix model. We show that ABAM enhances the expressive power of the access matrix model by supporting attribute-based authorizations. Specifically, ABAM is comprehensive enough to encompass traditional access control models as well as some usage control concepts and specifications. On the other side, expressive power and safety are two fundamental but conflictive objectives in an access control model. We study the safety property of ABAM and conclude that the safety problem is decidable for a restricted case where attribute relationships allow no cycles. The restricted case is shown to be reasonable enough to model practical systems.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access*

General Terms

Security

Keywords

access control, access matrix model, safety analysis, decidability

1. INTRODUCTION

Protection systems aim at protecting various resources from damage or unauthorized access, and allowing multiple users to share the same resources. A model should be defined in such a way that it is capable of expressing practical systems, most of which are dynamic. In a dynamic system, the state is changed to a new state by some external actions, or operations from subjects inside the system (including the system administrator). For an access control system, a state is specified by a set of subjects, objects, and access control configurations which determine the authorization relation for subjects on objects. Traditionally, an state change can be happened by an administrative operation. Modern access control systems are more dynamic that an access operation of a subject can change the system state.

Access control matrix is a simple but useful model first formalized by Harrison, Russo, and Ullman (HRU model) [2]. In HRU, an object is represented by a column and a subject is represented by a row and a column. A right in a cell specifies that the subject of the row has the right on the object of the column. An access control matrix is specified by a set of subjects, a set of objects, and a fixed set of commands. A command takes a set of object parameters as input and updates the matrix by adding or removing right(s) in a cell, or creating or destroying an object based on checking presence of rights in a number of cells. A system state changes through execution of commands. The strong feature of the HRU model is its expressive power. It could model most of the protection systems in use at that time when it was proposed and many years thereafter. However, the safety of the HRU model is undecidable in general. The safety problem is the one to determine whether or not a given subject can eventually obtain an access privilege to a given object; if there is an algorithm that is able to correctly decide this, then the safety problem is said to be decidable.

In this paper we propose ABAM, an attribute-based access matrix model, by combining the simple access matrix model, as well as attributes-based authorization which is motivated by recently presented access control systems such as DRM and trust management. In such systems, an authorization decision is determined by predicates over subject or object attributes. For example, a web user (subject) can download a music file (object) only when his/her credit (attribute) is more than a constant value. We show that ABAM has more expressive power than HRU, and its safety is decidable with some reasonable restrictions.

The rest of the paper is organized as follows. Section 2 presents some background and related work. Section 3 gives a formalized specification of ABAM model. In Section 4, we study the safety problem of ABAM model. Section 5 summarizes this paper.

2. BACKGROUND AND RELATED WORK

2.1 Expressive Power and Safety Analysis

Since the proposal of HRU, a number of protection models were developed to provide stronger safety. Lipton and Snyder [4] intro-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA
Copyright 2005 ACM 1-58113-964-0/05/0003 ...$5.00.

duced the take-grant model with linear time algorithms for safety. It was deliberately designed to be of limited expressive power, so that it would not exhibit the undecidable safety of HRU. Sandhu's schematic protection model (SPM) [6] was developed to fill the gap in expressive power between take-grant and HRU, while sustaining the efficient safety analysis. Sandhu also proposed a typed access matrix (TAM) model [7] which has stronger safety properties and broad expressive power. The concept of strong typing is introduced in HRU to achieve stronger security. To study the safety properties of TAM, the monotonic case of TAM was considered. Monotonic TAM (MTAM) consists of only the monotonic primitive operations (the remove and destroy operations are not present). MTAM is an acyclic MTAM when it is assumed that an object cannot create another object of the same type, and that there are only a finite number of generations (since there are finite number of types in the model). The safety of MTAM is undecidable in general, but is decidable (NP-hard) in the acyclic case. Specifically, a simplified version of acyclic MTAM (acyclic case of ternary MTAM) is defined, for which, the safety has polynomial complexity. The TAM has a variety of decidable safety cases, but most of which are limited to monotonic systems.

Safety problem has been proven to be decidable mostly for strongly constrained systems (like monotonic systems). Recently, the dynamic typed access matrix model (DTAM) was proposed [8] which allows the type of an object in TAM to change dynamically. It has been proved that the safety problem is decidable for non-monotonic protection systems in DTAM. To show this, a type-relationship (TR) graph is introduced; the safety problem for non-monotonic systems becomes decidable if the TR-graphs of the systems have no cycle with respect to parent-child relationship between objects. We follow the same approach to study the safety problem in ABAM.

2.2 Attribute-based Authorization

Many attribute-based authorization models have been proposed recently. Al-Kahtani and Sandhu [1] proposed an attribute-based user-role assignment model for RBAC. Traditionally a user is manually assigned to appropriate roles by a security administrator. RB-RBAC (Rule-Based RBAC) provides the mechanism to dynamically assign users to roles based on a finite set of authorization rules which are defined on user attributes. Li et al. [3] presented a role-based trust management (RT) as attribute-based authorization framework. The authorization decisions are based on chains of digitally signed attribute credentials through which credential issuers assert their judgments about the attributes of entities, such as users and organizations.

In our work, ABAM applies the simple structure of HRU model, and extends the primitive operations and commands for attribute update. HRU has been studied well in the literature and widely used in existing systems. We show that ABAM can express traditional access control models, as well as many modern applications.

2.3 Usage Control

The specification of the usage control model ($UCON_{ABC}$) proposed by Park and Sandhu [5] consists of a family of models built around the decision factors authorizations (A), obligations (B), and conditions (C). The decision factors are based on the access rights, and the attributes of the subjects and objects involved. Authorizations evaluate subject attributes, object attributes, and requested rights together with a set of authorization rules for usage decision. Obligations are functional predicates that verify mandatory requirements a subject has to perform before or during a usage exercise. Conditions are environmental or system oriented decision factors. Evaluation of conditions cannot update any subject or object attributes. The conditions are system dependent values which need to be updated by the system. The ABAM model proposed in this paper captures the features of attribute-based authorization and mutability specified in UCON. The difference between ABAM and UCON is that in ABAM, multiple subjects and objects along with their attributes are considered in the decision process, while for UCON, a policy refers to a specific pair of (subject, object). More specifically, in the $UCON_{ABC}$ model, the usage decision (to allow subjects to use certain objects) is made based on an individual pair of subject and object; whereas in our model, multiple conditions are checked involving multiple subjects and objects to perform certain usage decisions.

3. A FORMAL MODEL OF ABAM

ABAM is defined in terms of access control matrix and commands following the traditions of defining HRU, TAM and other access control models. Subjects and attributes, objects and attributes, access rights, access matrix, primitive operations and commands are the basic components constituting the new model.

3.1 Subjects, Objects, and Access Rights

DEFINITION 1. *All the entities to be protected in the system (passive entities) are called objects. The set of all active entities (for example processes and users) that can invoke some access requests or execute some permissions on an object are called subjects.*

A subject can be accessed by another subject. For example, a process can be initiated or killed by another process. Hence subjects can also be considered as objects. Following the general concepts in traditional access control models, we consider the set of subjects in ABAM to be a subset of the set of objects. The objects that are not subjects are pure objects. Sometimes we refer to a subject or an object as an entity when the distinction is immaterial.

We define an access matrix (defined later) with columns representing the subjects and objects in the system, and rows representing the set of all subjects. Each cell in the access matrix represents the rights of the corresponding subject over the corresponding object.

Each object in ABAM is specified by a unique identity recognized by the system. Note that the identity here is not the "identity" in traditional access control models which is totally authenticated by the system, but a simple name or ID assigned by the system when an object is created, and cannot be changed after the creation.

DEFINITION 2. *Access rights are the kind of access that the subjects can execute on objects (e.g., read, write, execute). Let R represent a finite set of access rights.*

3.2 Attributes and Attribute Tuples

DEFINITION 3. *Each entity has a set of attributes. Each attribute is a variable of a specific data type, which is dependent on what the attribute is, and determines a domain from which a a value can be assigned to the attribute.*

DEFINITION 4. *For entity o, $ATT(o)$ is the set of attribute values which is represented by a tuple of values of each of the entity's attributes, called the* attribute value tuple *or simply the* attribute tuple.

For example, an administrator in an organization has an administrative role, while a normal user does not have this attribute. Without loss of generality, we assume that each entity has the same set

of attributes, which is the maximum set in the system. For an attribute that an entity does not have, the value is $null$ in the attribute tuple.

An attribute of an entity is denoted as $ent.att$ where ent is the entity name (i.e., the entity's identity) and att is the attribute name. Hereafter, we assume that an entity name without any attribute specified denotes its identity.

The attribute value tuple of an entity specifies its state in the system. Changing the attribute value(s) of an entity changes its attribute value tuple. For example, consider an object o with attributes $(a_1, a_2, \ldots, a_n)$, where $a_1, a_2, \ldots, a_n$ takes values from domains $V_1, V_2, \ldots, V_n$ respectively. In a particular state of the system, the object has the attribute value tuple $ATT(o) = (a_1 = v_1, a_2 = v_2, \ldots, a_n = v_n)$, where each of $v_i \in V_i$ for $1 \leq i \leq n$. Updating an attribute of a_i from value v_i to v_i' means that the attribute tuple changes from $ATT(o)$ to $ATT(o)' = (a_1 = v_1, a_2 = v_2, \ldots, a_i = v_i', \ldots, a_n = v_n)$, where $v_i' \in V_i$, and $v_i' \neq v_i$.

3.3 Attribute Predicates

A predicate is a boolean expression built from attributes and constants with appropriate operation and relation symbols. We can distinguish different types of predicates in ABAM, such as unary predicates and binary predicates. A unary predicate is built from one attribute variable and constants, e.g., $Alice.credit \geq \$100.00$, $file1.classification = `supersecure'$. A binary predicate is built from two different attribute variables, e.g., $dominate(Alice.clearance, file1.classification)$, $Alice.credit \geq ebook.value$, $(Alice, r) \in file1.acl$, where $file1.acl$ is object $file1$'s access control list. Note that the two attributes in a binary predicate can be from a single entity, or two different entities. In real systems and applications, we can define more general predicates built on any number of subject attributes and/or object attributes. This significantly improves the flexibility and expressive power of ABAM.

3.4 Access Matrix and Protection State

DEFINITION 5. *An* access matrix *is a matrix with a row for each subject with its attribute tuple, and a column for each object with its attribute tuple.*

The set of all subjects along with their attributes are used to represent the rows, and the set of all subjects and objects with their attributes represent the columns. An element in the cell $[s, o]$, representing the row s and column o, stores the set of all access rights that s can exercise over o.

Figure 1 shows an access matrix for two subjects $S = s_1, s_2$ and two objects $O = o_1, o_2$. Each row is represented by a subject and its attributes, and each column by an object with its attributes.

	s_1: ATT(s_1)	s_2: ATT(s_2)	o_1: ATT(o_1)	o_2: ATT(o_2)
s_1: ATT(s_1)		{parent}	{read, write}	
s_2: ATT(s_2)			{read}	

Figure 1: An access matrix

The access matrix in HRU is a special case of the access matrix defined here, where there is no attribute tuple associated with a row or column.

DEFINITION 6. *A* protection state, *or simply* state *of an ABAM system is specified by a set of subjects S, a set of object O, a set of subject attribute tuples $ATT = (ATT(s_1), ATT(s_2), \ldots)$, where each $ATT(s_i)$ is the attribute tuple of subject $s_i \in S$, a set of object attribute tuples $ATT = (ATT(o_1), ATT(o_2), \ldots)$, where each $ATT(o_i)$ is the attribute tuple of object $o_i \in O$, and an access matrix M with a row for every subject with its attribute tuple, and a column for every object with its attribute tuple.*

3.5 Primitive Operations

A primitive operation is an operation that can change the state of a system. The basic primitive operations can be defined as follows.

1. Enter r into $[s, o]$: Enters generic right r into cell$[s, o]$ in the access matrix.
2. Delete r from $[s, o]$: Deletes generic right r from cell $[s, o]$ in the access matrix.
3. Create subject s with attribute tuple $ATT(s)$: Creates a new subject s with attribute tuple. An identity attribute with a unique value is created at the same time and assigned to the subject by the system.
4. Destroy subject s: Removes subject s as well as its attribute tuple from the system.
5. Create object o with attribute tuple $ATT(o)$: Creates a new object o with attribute tuple $ATT(o)$. An identity attribute with a unique value is created at the same time and assigned to the object by the system.
6. Destroy object o: Removes object o as well as its attribute tuple from the system.
7. Update attribute $o.a_i$: $ATT(o) \rightarrow ATT(o)'$: Updates the attribute tuple $ATT(o) = (v_1, v_2, \ldots, v_i, \ldots, v_n)$ to $ATT(o)' = (v_1, v_2, \ldots, v_i', \ldots, v_n)$ for an object o, where $v_i, v_i' \in V_{a_i}$, and $v_i' \neq v_i$, V_{a_i} is the value domain of $o.a_i$.

From above definition, we can see that ABAM has a new primitive operation (the update attribute operation) to change a system state. This enhances the expressive power of ABAM beyond HRU model as shown later.

3.6 Commands

A command in ABAM consists of condition and a sequence of primitive operations. The condition in a command of HRU model searches for existing rights in the access matrix, whereas in ABAM, along with the right existing, some predicates on attributes of the subjects and objects are also evaluated.

A command in ABAM is defined as follows.

Command $\alpha(X_1 : ATT(X_1), X_2 : ATT(X_2), \ldots, X_k : ATT(X_k))$
if $r_1 \in [X_{s1}, X_{o1}] \wedge r_2 \in [X_{s2}, X_{o2}] \wedge \ldots r_m \in [X_{sm}, X_{om}] \wedge$
$p_1 \wedge p_2 \wedge \ldots p_n$
then
$op_1; op_2; \ldots; op_n$
end

Here, α is the name of the command, $X_1, X_2, \ldots, X_k$ are subject or object parameters; r_1, r_2, …, r_m are generic rights; $s1$, $s2$, …, sm and $o1$, $o2$, …, om are integers between 1 and k; p_1, p_2, …, p_n are predicates built over attribute tuple $ATT(X_1)$, $ATT(X_2), \ldots, ATT(X_k)$. $op_1, op_2, \ldots, op_n$ are primitive operations; The if part of the command is called the condition of α.

If a command has no condition, it is an unconditional command, otherwise, it is a conditional command. A command is invoked by substituting actual subjects and objects. The operations are executed sequentially if the condition is true.

It should be noted that a disjunctive form of conditions can also be easily modeled by having one command for each component condition. Also, negated predicates are not required explicitly, since we always can define a normal predicate for a negated one.

3.7 ABAM Model

After introducing the basic components, we complete this section with the definition of ABAM authorization scheme and system.

DEFINITION 7. *An ABAM* authorization scheme, *or simply* scheme, *consists of a finite set of generic rights R, a finite set of attribute predicates P, and a finite set of commands C. An ABAM* system *is specified by an ABAM authorization scheme and an initial state (an initial set of subjects and attribute tuples, objects and attribute tuples, and an initial access matrix).*

In an ABAM system, commands can only be executed serially. And each execution is atomic, that is, either the condition of a command is satisfied and all primitive operations are performed successfully, or no change happens on the current state.

3.8 Expressive Power of ABAM

With the extension beyond HRU model, ABAM has the expressive power not only for traditional models, but also for modern access control systems. One can easily use the new model to express the HRU model (without considering the attributes) and TAM model (allowing no update on the attributes where the attributes can be considered as types). One can also use ABAM to simulate basic usage control models which allow the update of attributes before or after the access. Due to space limit, we will not elaborate on this.

4. SAFETY ANALYSIS OF ABAM

We describe a restricted case of ABAM model and study its safety properties in this section. It is shown that the safety of this restricted case is decidable. The expressiveness of this case is also discussed. For safety analysis we assume that (i)every entity in the system has a finite number of attributes, and (ii)each attribute takes values from a finite value domain.

From the above assumptions it can be said that the set of all possible attribute tuples, which all of the subjects and objects in the system can assume, is finite. Let A denote this set of all attribute tuples. In a particular state, each entity in the system assumes its attribute tuple from A. Any change of a single attribute tuple (i.e., replacing a tuple with another tuple in A) moves system states from one to another. It is reasonable enough to model many practical systems with these assumptions. The expressiveness of the restricted model is explained later in this section.

4.1 Acyclic ABAM Scheme

We first define an attribute-relation graph, based on which we define acyclic ABAM scheme.

DEFINITION 8. *For a command α $(X_1 : ATT(X_1), X_2 : ATT(X_2), \ldots, X_k : ATT(X_k))$, if its body has an operation "create subject/object x_i with attribute tuple $ATT(x_i)$", where $1 \leq i \leq k$, then*

- *$ATT(x_i)$ is a* creating-child attribute tuple *in α;*
- *If $ATT(x_j)$ ($1 \leq j \leq k$) is not a creating-child attribute tuple in α, then $ATT(x_j)$ is a* creating-parent attribute tuple *in α;*
- *if every $ATT(x_j)$ ($1 \leq j \leq k$) is a creating-child attribute tuple in α, then all $ATT(x_j)$ ($1 \leq i \leq k$) are said to be* creating-orphan attribute tuples *in α.*

DEFINITION 9. *For a command $\alpha(X_1, X_2, \ldots, X_k)$, if the body of α has an operation "update attribute of $x_i.a_j$: $ATT(x_i) \rightarrow ATT(x_i)'$" ($1 \leq i \leq k, 1 \leq j \leq n$, n is the number of attributes for an entity), then,*

- *$ATT(x_i)'$ is said to be an* updating-child attribute tuple *in α, and $ATT(x_i)$ is said to be an* updating-parent attribute tuple *in α;*
- *if the body of α has no update attributes operation with respect to x_i, and $ATT(x_i)$ is a creating-parent attribute tuple in α, then $ATT(x_i)$ is both* updating-parent attribute tuple *and* updating-child attribute tuple *in α.*

DEFINITION 10. *A* create command *is a command with a create subject/object operation. Otherwise it is called a* non-creating command.

DEFINITION 11. *If the execution of a command α $(X_1 : ATT(X_1), X_2 : ATT(X_2), \ldots, X_k : ATT(X_k))$ creates new subjects/objects, then x_i ($1 \leq i \leq k$) is said to be a* parent *if $ATT(x_i)$ is a creating-parent attribute tuple in α; otherwise, a* child.

DEFINITION 12. *A descendant of an entity x is recursively defined as itself or a child of a descendant of x.*

From the command structure in ABAM, it can be seen that a command allows conditional creation of an entity. That is, a subject or object is created only when certain conditions are satisfied. Thus, child of an entity is created conditionally, and hence, an entity has conditional descendants.

DEFINITION 13. Attribute-relation (AR) graph (V, E) *of an ABAM system is a directed graph with the set of vertices V to be the set of all attribute tuples (the set A defined early in this section), and the set of edges $E \subseteq V \times V$. A pair $(v_1, v_2) \in E$ iff*

- *$\exists \alpha \in C$, v_1 is a creating-parent attribute tuple in α, and v_2 is a creating-child attribute tuple in α; or*
- *$\exists \alpha \in C$, v_1 is an updating-parent attribute tuple in α, and v_2 is an updating-child attribute tuple in α.*

From an AR graph, it can be seen that, if there is a cycle, an entity in a state can create an infinite number of subjects or objects. Also, subjects or objects with orphan attribute tuples can be created infinitely. The existence of cycles and orphan attribute tuples in the AR graph is critical to decide whether the number of entities in a protection system are finite or not. Note that a self loop is regarded as a special case of a cycle with length one.

LEMMA 1. *Consider an AR graph that has no cycles containing creating-parent attribute tuples. Given a creating command α, if $ATT(x)$ is a creating-parent attribute tuple, then α must update attributes of x to $ATT(x)'$ such that $ATT(x)' \neq ATT(x)$.*

Proof Sketch: The lemma is proved by contradiction. Assume that $ATT(x)$ is a creating-parent attribute tuple in α, and the body of α has no update attributes operations on $ATT(x)$. Then, $ATT(x)$ is both a updating-parent attribute tuple and a updating-child attribute tuple with respect to update attributes in (from Definition 9). This means that there is a cycle in the AR graph with a creating-parent attribute tuple. This is a contradiction to our assumption. □

From the above lemma it can be concluded that creating commands make irreversible changes on attributes of parent entities.

DEFINITION 14. *An ABAM scheme is said to be* acyclic *if (i)the system has no orphan attribute tuples, and (ii)the AR graph of the system has no cycle that contains creating-parent attribute tuples in creating commands. An ABAM system is* acyclic *if its scheme is acyclic.*

The definition of an acyclic protection system means that the set of all entities derived from an initial state have distinct attribute tuples. No two entities derived from the initial state have the same attribute tuples in any state of the system.

4.2 Safety Analysis of Acyclic ABAM Systems

This section gives a formal proof of the decidability of the safety problem for a restricted case of ABAM model.

THEOREM 1. *The safety problem for an ABAM system is decidable if (i)the authorization scheme of the system has no creating-orphan attribute tuples, and (ii)the AR graph of the system has no cycles that contain creating-parent attribute tuples in creating commands.*

Proof Sketch: To prove this, we first show that the number of entities in an arbitrary protection state of an acyclic protection system has an upper bound. Since there are no orphan attribute tuples, every object is a descendant of an entity in the initial state. Now, if we can prove that the number of descendants of an arbitrary entity x in the system is finite, then the total number of entities in an arbitrary protection state will be finite.

Let N_{max} be the maximum number of create operations in a command in the authorization scheme. This number is finite since the number of primitive operations in the body of any command is finite. Consider an existing entity x. If a creating command α can be executed with x as a parameter, the attribute tuple of x is always a creating-parent attribute tuple in α. From Lemma 1, α must update at least one attribute of x by replacing $ATT(x)$ with $ATT(x)'$ from set A.

Thus, if multiple create commands are executed with x as a parameter, the attribute tuple of x is changed for each of the create commands. The bound on the number of execution of these create commands depends on the number of times the attribute tuple of x can be changed. Since the total number of attribute tuples is a finite number $|A|$, the maximum number of create commands that can be executed with x as a parameter is $|A| - 1$. Then the maximum number of direct children of during the lifetime of the system is $N_{max} \times (|A| - 1)$. Also, the maximum number of generations of descendants of x is $|A|$, since if it is greater than $|A|$, there must exist two entities with the same attribute tuple, which contradicts the acyclic scheme property assumed. Thus, the total number of descendants of an arbitrary entities in the system is finite, that is, the number of subjects and objects in an arbitrary protection state of the system has an upper bound. This implies that the total number of distinct protection states of the system is also finite. Hence we can check whether or not a particular subject has a certain right over a particular object in every reachable state from the initial state. Thus, the safety problem is decidable. □

THEOREM 2. *Safety problem for acyclic ABAM systems with finite domain of each attribute is NP-hard.*

Proof Sketch: Our decidable ABAM model can simulate a decidable DTAM model introduced in [8]. Specifically, suppose in ABAM, there is only one attribute $type$ for each object, and its domain is a fixed set of type names. Then the attribute relation graph is the type graph in DTAM. For DTAM systems without creating commands and the type graph is acyclic, the safety problem is NP-hard. Therefore we conclude that the safety problem in ABAM with acyclic scheme and finite domains of attributes is NP-hard since it subsumes this DTAM model. □

This implies that our restricted model has at least the expressive power of acyclic DTAM. In addition, the restricted model inherits the mutability feature (i.e., attribute update) from usage control framework. Thus, one can say that the restricted model is more expressive than the existing model.

5. CONCLUSIONS

The contribution of this paper is two-fold. First, we proposed a new model of attribute-based access control with expressive power greater than HRU. Second, we analyzed the safety properties of this in a restricted case. The main difference between this model and previous models is that attributes are introduced into access control matrix, commands, and predicates. Traditional access control models such as HRU, TAM and DTAM are shown to be special cases of our model. Apart from modelling HRU, TAM, and DTAM, the new model has some features of the usage control model that is recently proposed; thus, it is generic for modern access control applications such as DRM and trust management. The safety problem of this new model is proven to be decidable for a restricted case where the set of attribute values is finite, and the attribute relationships allow no cycles. It has also been shown that the restricted case has broader expressive power than DTAM.

6. REFERENCES

[1] M. A. Al-Kahtani and R. Sandhu. A model for attribute-based user-role assignment. In *Annual Computer Security Applications Conference*, 2002.

[2] M. H. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976.

[3] N. Li, W. H. Winsborough, and J. C. Mitchell. Design of a role-based trust management framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[4] R. J. Lipton and L. Snyder. A linear time algorithm for deciding subject security. *Journal of ACM*, 24(3):455–464, 1977.

[5] J. Park and R. Sandhu. The uconabc usage control model. *ACM Transactions on Information and Systems Security*, 2004.

[6] R. S. Sandhu. The schematic protection model: Its definition and analysis for acyclic attenuating schemes. *Journal of ACM*, 35(2):404–432, 1988.

[7] R. S. Sandhu. The typed access matrix model. In *IEEE Symposium on Security and Privacy*, pages 122–136, 1992.

[8] M. Soshi. Safety analysis of the dynamic-typed access matrix model. In *Proc. 6th European Symposium on Research in Computer Security*, 2000.