A Role Description Framework and its Applications to Role-Based Access Control*†

Mats Gustafsson and Nahid Shahmehri

Laboratory for Intelligent Information Systems
Department of Computer and Information Science
Linkping University, S-581 83 Linkping, Sweden
Telephone: (+46) 132 826 08, 132 820 66, Fax: (+46) 132 826 66
Email: {matgu,nahsh}@ida.liu.se

May 30, 1996

Abstract. This paper presents a novel framework for role descriptions for increased information security. The generality of the framework makes it suitable for modeling roles in a wide range of applications such as Role-Based Access Control and Workflow Management. The framework can also be used as a common modeling platform in information systems.

We make a clear distinction between the definition of a role and the uses it is put to. Apart from an application data block that can be used to store access control information, a role description also includes the relations a role has to other roles.

Applied to Role-Based Access Control (RBAC), the framework can be used to model most aspects important in this area, for example constraints and role hierarchies. Our framework can complement and extend previously proposed frameworks by placing roles into a wider organizational context.

Keywords: Organizational Models, Role Descriptions, Role Based Access Control, Information Security

1 Introduction

Role-Based Access Control (RBAC) is an area that has received increasing attention in recent years. By separating the assignment of rights and permissions to roles from the assignment of users to roles, the security structure can be kept essentially static even though users may change over time.

The role concept, however, is wider than what is reflected in RBAC. In this paper we want to show how roles can be placed in a broader organizational perspective, and how roles viewed from that perspective relate to RBAC.

We present a basic role description framework that has several important merits. Our framework provides a general way of modeling organizations by using roles and inter-role relations. In this paper we show how the framework can be used to describe RBAC. However,

*This work is supported by the Swedish National Board for Industrial and Technical Development (NUTEK), project P5953-1.

†Presented at the IEEE WET ICE '96 International Workshop on Enterprise Security at Stanford University, Palo Alto, June 19–21 1996.

the application of the framework is not limited to this area. In future work, for instance, we intend to use the framework to integrate roles into other fields such as Workflow Management and Business Process Modeling.

Apart from being suitable for building models of organizations for use in information systems, we believe that the framework is also a useful tool for analyzing organizations themselves.

The rest of this paper starts out with a review of and discussion on the role concept in section 2. In section 3 we identify the desirable properties of a role description framework, before we go on to propose a framework in section 4. Section 5 exemplifies how the framework can be used to model important aspects of RBAC. We compare our work to that of others in section 6 and we make some concluding remarks in section 7.

2 Different views on the concept of role

The concept of role has long been used in the behavioral sciences and in enterprise modeling [Bid79]. Considering this background, Sloman defines a *manager role* as a set of authorization and obligation *policies* for which a particular *manager position* is the subject [Slo94]. A manager position is defined as a particular position within an organization[1]. Further, Sloman mentions four important facets associated with a position within an organization, namely *authority*, *responsibility*, *functions*, and *interactions*. Focusing on organizational policies, Sloman sees roles mainly as a tool for disseminating policies within an organization. This is also the approach taken by Marriott and by Lupu, Marriott, Sloman, and Yialelis [Mar93, LMSY95].

Sandhu, Coyne, Feinstein, and Youman adopt a similar view when they stress the importance of distinguishing roles as policy from the mechanisms used to implement them in an access-control system [SCFY94]. In the article the authors derive a taxonomy of security models into which roles, and RBAC, are categorized as *computer policy models.* In [SCFY96] the same authors propose a family of reference models for RBAC. The base model is composed of *users*, *roles*, *permissions* and *sessions*, to which the *permission to role* and *user to role* assignment relations are added. In an extended model the authors also include *role hierarchies* and *constraints.*

A common operational approach, exemplified by Nyanchama and Osborn, is to define roles simply as named collections of privileges, whose specification may be subjected to any type of security policies [NO93a, NO93b]. Another approach is given by Demurjian, Hu, Ting, and Kleinman who propose a three level, user-role definition hierarchy consisting of *user classes*, *user types*, and finally *user roles* [DHTK93]. A modified version of this model has been adopted by Mohammed and Dilts [MD94].

In the literature there is a broad consensus about the administrative merits of RBAC. To us, one of the important benefits of RBAC is that it is possible to define a *static* security structure that is independent of individual users. By isolating the invariant parts, administration is reduced to a minimum. Many authors also find that RBAC has advantages compared to traditional security approaches like mandatory and discretionary access control (*MAC* and *DAC*) [CS92, DHTK93, SCFY94, SF94, MD94]. Sandhu and colleagues suggest that RBAC in a sense is *policy neutral*, as most other approaches, such as MAC and DAC, can be realized using RBAC [SCFY96].

2.1 Defining the role concept

As we have seen, there are many approaches to define the notion of role. Not many authors place roles into a broader context or reflect aspects beyond access control in their role

[1]The word *manager* is here given a broad meaning denoting "someone who manages something".

definitions. We believe it is important to view RBAC, or any security approach, not only as an administrative tool or a purely technical issue of implementation, but also as one of several components in an overall effort towards good computer security. Even though technically advanced, any security scheme will lose its effectiveness if there are not well defined *security goals*, and *security policies* derived from these goals, within the organization. It is also necessary that these policies are understood and endorsed by all those concerned. One goal for this paper is to expand the role concept and place it in a wider perspective. In doing so we also want to make clearer the distinction between the definition of roles and their uses. In the following we adopt the following, very basic, definition

> **(Role).** *A role is a position or job function within an organizational structure.*

From this definition it can be seen that, in order to completely describe a role, it is necessary to describe both the role *function* as well as its place in an overall structure.

3 Requirements for a role representation

To be able to put roles into a broader context, the representation of a role within an information system must allow for more than just privileges and access rights. What then, are the requirements for a such a representation? Apart from general requirements, such as *extensibility* and *modifiability*, commonly found in Software Engineering, we have identified the following general criteria:

Formal foundation. In order to be able to make any predictions at all about whether a given role framework meets specified security goals and policies or not, it must be possible to reason about roles. This requires a formal foundation.

Completeness. The model should be able to capture a wide range of facets of the role concept. We feel that roles are a powerful tool for describing organizations that have several uses besides that of access control. E.g. Lupu and colleagues have made a strong coupling between organizational policies and roles [LMSY95]. We believe there are other interesting connections to be made, for example in the area of Workflow Management.

Scalability. Any general framework runs the risk of becoming very complex in order to achieve completeness. To prevent role descriptions from becoming unwieldy, it must be possible to omit from an instantiation those parts of the framework that are not applicable to that particular case.

Applicability. In many cases RBAC can be used to describe existing security policies. Role-based security schemes can also often be implemented, or emulated, using existing tools and mechanisms. It should be possible to apply a role description framework also to these cases. By allowing for legacy systems in the framework in this way, comparisons between present and proposed systems become possible.

4 A role description framework

As mentioned when defining the notion of role in section 2.1, there are two main components in any role description, the *role function* and the *relations* the role has to other roles. For roles to be meaningful, it must also be possible to assign users to roles. In addition to these components, it should be possible to associate *application data* with the role description. In the case of RBAC this may be privileges and permissions given to the role.

4.1 The Role Descriptor Object

The basic entity in our role description framework is the *Role Descriptor Object* (RDO). The RDO is a representation of the external role concept used within the information system and contains the information associated with a role. We define the RDO to be a 4-tuple $<F, R, A, U>$ where

- F is a *function description* for the role,
- R is an entity describing the *relations* the role has to other roles,
- A is a block of *application data* associated with the role,
- U is a list of *users* that can assume the role.

An *organizational model* $\mathcal{M}$ is a set of RDOs.

4.2 Function description

The size of the final organizational model depends greatly upon the granularity with which role function descriptions are made.

Taken to one extreme, it would be possible to allot a role to each *task* to be performed within an organization, resulting in a multitude of roles of type *performer-of-task-x*. This poorly fits our given definition of a role, and it would also be very cumbersome to administrate.

On the other extreme we could try to make our role descriptions as general as possible in order to reduce the number of roles, and maybe thereby also the administrative effort. Again, the resulting roles may fail to reflect the true organizational structure and the fact that many persons, although having only one *job*, serve their organization in several different capacities.

Obviously, the exact granularity and extent of a role function description is highly dependent on the demands of the organization of which it is a part. We believe that this judgment in practice will come naturally, provided the underlying organizational structure is well understood.

As it is hard to make any statements regarding the extent of the function description it is also hard to enforce any specific structure to the description. At this stage we will not go further than to specify the function description attached to the RDO to be a textual one, intended for human readers.

4.3 Relations between roles

As the name suggests, positions within an organization are defined in terms of how they relate to other positions. A common relation of this kind is the *supervisor-subordinate* relation.

There may also be other relations between the roles in an organization. By using other relations it is possible to get other views of the organization. Lupu and associates mention several classes of such relations; *client-server*, *contractual*, *collaborative*, and *peer-to-peer* [LMSY95].

The designated *type* of any relation imposes constraints on how objects can be related. E. g. ordering relations are anti-symmetric and generate a graph that is free from cycles. Symmetric relations, on the other hand, require that $(a, b) \in R$ **iff** $(b, a) \in R$.

There are many ways to structure an organization and also many ways to model these structures. It would be presumptuous to think that all conceivable structures and models can be expressed with a fixed set of relations. For this reason it should be possible to define relations on a case-by-case basis.

Formally, relations are often seen as a set of tuples $<o_1, o_2, \ldots, o_n>$ designating related objects (c.f. relational databases). In our framework we will adopt another way to represent relations. Instead of a set of tuples, we maintain for each object a list R of related objects or object tuples[2]. Together with a *Relation Specification*, $\boldsymbol{r}$, we form a *Relation Descriptor*, $\mathbf{RD}{=}{<}r, R{>}$. The relation specification describes properties of the relation such as name and type.

4.4 Application data

In many applications the application data is probably the largest. The application data entity, **AD**, of an RDO should be seen as a container for application information that may originate from many sources. Any application may specify an arbitrary object that can be attached to the AD of a role. In this way flexibility is preserved while the basic role framework is kept small. This also emphasizes the distinction between roles and the uses they are put to.

4.5 Assignment of users to roles

Regardless of how roles are otherwise defined or used, we must always be able to assign *users* to roles. Outside the information system this amounts to determining which *persons* should be able to assume a certain role. In the information system a *user* can be a person, but also a program (agent).

To represent a user we use a persistent *User Descripton Object* (UDO) within the system.

In an RDO, the users that are capable of assuming the role are listed in a *Role User List* (RUL). For flexibility we also allow the RUL to contain *user groups.* A user group can be seen as a set of UDOs. If we let $\mathcal{U}$ designate the set of all UDOs in the system, and $\mathcal{G}$ the set of all user groups, then a UDO $\mathbf{u}$ is a *role user* **iff** $(\mathbf{u} \in \mathcal{U} \wedge \mathbf{u} \in \mathrm{RUL}) \vee (\exists g \in \mathcal{G}: g \in \mathrm{RUL} \wedge \mathbf{u} \in g))$.

5 Applying the framework to RBAC

In the preceding we have presented a general, yet concise, framework for modeling roles within an organization. We believe that the framework can be used to describe various applications of the role concept.

Today, the most prominent use of roles in information systems is that of Role-Based Access Control (RBAC). When designing a system using RBAC, a number of important issues must be considered [SCFY94, SCFY96]. These issues include *Role definition and administration*, *Assignment of users to roles*, *Assignment of priviliges*, *Role hierarchies*, *Constraints*, and *Role use.* In the following we shall examine how our framework can be applied to handle these issues.

5.1 Role definition and administration

Our framework implicitly defines at least three necessary logical information repositories. The *Role repository* services other parts of the system with information about roles, their users and other information present in the RDO. The *Organizational relations database* contains information about the relations used to structure roles. In addition, there must also exist a *User and group database* that maintains unique UDOs for each user in the system, as well as information about user groups. Which roles and relations that are defined depends on the organizational structure and policy.

We believe it to be a complex task to analyze an organization in order to find the roles and relations that accurately model it. Our framework offers a way to structure this work.

[2]This approach is most suited for binary relations, but n-ary relations can also be represented in this way by maintaining a list of (n-1)-ary tuples instead of a simple list of objects.

In the case of RBAC, the explicit separation of the role definition from role use makes it natural to let organizational structure govern the grouping of access privileges.

5.2 Assignment of users and priviliges

An important motivation for RBAC is that the static security structure provided by roles simplifies the administration when users join, move within, or leave an organization. However, in large organizations the sheer number of roles and users may make centralized administration cumbersome. To alleviate this situation, a mechanism can be introduced to delegate, in a controlled manner, the authority to add or remove users to/from roles.

The task of assigning system object access privileges to roles is analogous to that of assigning users to roles in that it often can be delegated. One should however keep in mind that these two tasks are separate and probably should be performed by different roles.

Assignment of users to roles is expressed directly in our framework while the assignment of privileges to roles must be realized through the application data entity in the role descriptor object. Abstractly, the privilege information is viewed as being stored *within* the RDO. In practice, privileges might of course be kept in a separate database.

5.3 Constraints

Many security policies cannot be directly expressed in terms of access privileges to system resources. Instead they take the form of constraints. One such constraint is a policy of *separation of duty* that mandates that two roles must not have any common role users. An often cited example is that of a payment that must be authorized by two different roles, e.g. `clerk` and `supervisor`, to minimize the risk of fraud. Obviously, the purpose of this policy is nullified if a user can take on both roles.

There are also several other types of constraints, as mentioned by Sandhu and colleagues [SCFY96]. These include *cardinality constraints* that can be used to limit the number of users that can be assigned to a role and *session constraints* that restrict the ways in which a user can activate roles.

In our framework, *inter-role relations* can be used to express constraints. Mutually exclusive roles can be expressed as related through a symmetrical, anti-transitive, and anti-reflexive relation. This relation can be expressed as a list of pairs[3], $\{(r_1, r_2), (r_1, r_3), (r_2, r_4), \ldots\}$. The relation is anti-transitive, thus a user may take on both roles r_1 and r_4. At the same time, a user of r_2 must neither be a user of r_1 nor r_4. In some instances it may be required to generalize this binary relation to encompass three or more roles.

Simple cardinality constraints may be expressed using unary relations by defining one relation for each cardinality. This, however, is an awkward approach. There are also other constraints, such as *time limitations*, that are difficult to express as general relations. For constraints that do not involve other roles it therefore seems more appropriate to place such constraint information in the application data block.

5.3.1 Enforcing constraints

Apart from a way to describe constraints, there must also be means to enforce them. How this is achieved in detail is of course implementation specific. Still, a few general observations can be made.

Given an organizational model, constraints can be seen as *assertions* about the model that must hold at all times. A model in which all constraints are satisfied is called *consistent*. A way to preserve consistency is to check that changes to the model, such as the assignment of a user to a role, do not cause any constraint to be violated. What is more difficult to

[3]The symmetry requirement implies that also the pairs $\{(r_2, r_1), (r_3, r_1), (r_4, r_2), \ldots\}$ are in the relation.

handle is the addition of new constraints. Imposing a new constraint may affect many roles in the model and have unforeseen ramifications. Because of this, there must also be a way to check the consistency of an entire model instance.

Not all constraints concern properties of the organizational model itself. For instance there may be constraints that regulate the time of day a user may assume a role, or that impose restrictions on which roles a user may have active simultaneously. We here need to introduce a concept of *user sessions*, managed by a trusted *session manager*. When a user asks the session manager for access to a role the session manager is responsible for checking pertinent constraints before granting the access. One constraint is always checked; the user wanting access has to be listed as a role user.

5.3.2 Categorizations

As we have seen, when looking at how constraints can be enforced, it is natural to make a distinction between constraints on the static properties of the organizational model and constraints on the actual use of roles. This categorization is only one of many possible. Figure 1 schematically illustrates some possible categorizations. One such categorization is that of *inter-role constraints* versus *single role constraints*, e.g. mutually exclusive roles compared to cardinality constraints.

It is also possible to classify along the axis of *what is constrained* by the constraint. Mutually exclusive roles constrain the *assignment of users* to roles, whereas time limitations constrain the *use of a role*. Yet another category would be constraints on *privilege assignment* to roles.

Constrained entity	Number of roles invovled		
	Two or more	Single role	
User assignment	X	X	Constraints on static structure
Privilege assignment	X	X	
User cardinality		X	
Time restrictions on role use		X	Constraints on system dynamics
Concurrency restrictions on role use	X		

"X" = Conceivable constraint

Figure 1: Different categorizations of constraints

5.4 Role hierarchies

When defining roles, it often is desirable to create a role that is a specialization of another role. One would, for example, in most cases expect the role of `store-manager` to include the role of `store-salesperson`. This naturally leads to the idea of role hierarchies. In hierarchies there can also be *pseudo-roles*. Pseudo-roles have been suggested by Parker and Sundt based on practical experiences [PS95]. If used with moderation, pseudo-roles can be used to stretch the role concept a little, making it possible to define general roles such as `department-employee` or more specific ones that only serve to provide grouped access to a set of resources.

In our framework it is possible to express role hierarchies by introducing a subsumption relation, $\sqsupseteq$. In the example above, the role of `store-salesperson` would be subsumed by the role `store-manager`.

Subsumption means to properly include one role within another, implying that all attributes in a role r_1 also are present in r_2 if $r_1 \sqsubseteq r_2$[4]. When r_2 subsumes r_1 in this way one usually says that r_2 *inherits* the attributes from r_1.

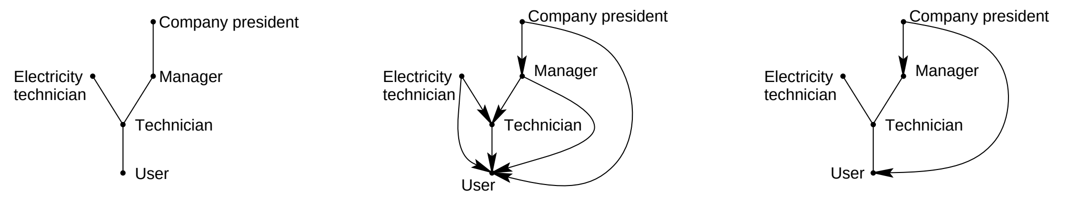
Consider as an example the following roles: `user`, `technician`, `electricity-technician`, `manager` and `company-president`. Figure 2(a) shows how these roles are related using subsumption:

`user` $\sqsubseteq$ `technician` $\sqsubseteq$ `electricity-technician`

and

`technician` $\sqsubseteq$ `department-manager` $\sqsubseteq$ `company-president`.

However, it may not be appropriate for the `company-president` role to inherit the privileges of the `technician` role. To address this situation a relation *inherits-from* can be defined, this relation is illustrated in figure 2(b). The relation is not transitive, thus the `company-president` role will only inherit from the `manager` role those attributes that are specified specifically for that role. Privileges available to the `manager` role by inheritance are not available to the `company-president` role. However, exclusive use of the *inherits-from* relation is cumbersome, in practice the combination shown in figure 2(c) is more likely.



(a) Subsumption relation (b) Anti-transitive "inherits-from" relation (c) A combination of (a) and (b)

Figure 2: Inheritance relations

We believe that role hierarchies primarily should be seen as administrative tools that are not visible to the user. By defining role hierarchies it is possible to avoid having to duplicate the assignment of attributes to similar roles. As we have seen it can sometimes be advantageous to define "pseudo-roles" that only are used for inheritance. Pseudo-roles can be assigned a cardinality constraint of "0" to prevent them from being assigned directly to users.

5.5 Role use

In section 5.3.1 we introduced the concepts of *user session* and *session manager*. The session manager handles the dynamic side of RBAC that arises when users take on and relinquish roles. We have not yet included these entities into our framework in a formal way.

The primary task for the session manager is to grant access to roles and to convey to the user the privileges associated with these roles. As we have mentioned, granting role access involves checking a number of constraints, the basic one being that the user should be listed in the list of allowed users for the role.

[4] The subsumption relation written as $x \sqsubseteq y$ should be read as *x is subsumed by y*, which is equivalent to *y subsumes x*, written $y \sqsupseteq x$.

The nature of the privileges granted to the user are very implementation specific. There will probably be a large difference between a centralized system and a distributed one. A centralized system can implement an access server that mediates all access to protected system objects, whereas in a distributed system privilege information has to be conveyed to remote applications.

6 Relations to other work

The relation and interaction components in the RDOs of our framework have been inspired by the ideas set forth by Lupu, Marriott, Sloman and Yialelis in [LMSY95]. An RDO in our framework corresponds to a *Manager Position Domain* in the terminology of Lupu and colleagues. What these authors term *User Representation Domain* roughly corresponds to our UDOs. Relating to earlier work, the authors commit to use a policy description language for expressing privileges and constraints. We have not adopted this approach as this would limit the ability of our framework to model a wide range of systems. It would, however, be possible to associate policy descriptions with RDOs using the application data block.

Other authors who have presented a coherent framework for describing RBAC are Sandhu, Coyne, Feinstein and Youman [SCFY96]. In this work, four models for role-based access control are described. $RBAC_0$, the basic model, describes *users*, *roles*, *permissions*, and *sessions*. The model defines two relations relating permissions and users to roles. $RBAC_0$ also introduce the concept of *session*. $RBAC_1$ extends $RBAC_0$ with a role dominance relation, $\geq$, to introduces the concept of *role hierarchies*. $RBAC_2$ extends $RBAC_0$ to include *constraints*. Finally, $RBAC_3$ combines $RBAC_1$ and $RBAC_2$ into a consolidated model. Our framework can be adopted to include as a subset the components of the models proposed here, with the exception of the described mappings from *sessions* to *users* and *roles*. Sandhu and associates have also proposed a *management model* for RBAC. This is an aspect that we have not yet studied in the context of our framework. We do, however, believe it is possible to adopt the management model in our framework.

7 Concluding remarks

In this paper we have presented a novel role description framework. The purpose of the framework is to provide a way to describe organizational roles in a general way that is also suitable for use in information systems. We have intentionally made a distinction between role descriptions in our framework and the potential uses roles can be put to. Our framework does not replace other proposed frameworks for RBAC. Our work instead contributes by serving as complement to existing frameworks by putting the roles of RBAC into a wider context and by allowing role definitions to be used also in other parts of the information system.

Before defining the framework we have identified a number of requirements. Of these, we have demonstrated applicability by showing how our framework can be used to model some of the more important aspects of RBAC. As we have not specified any component of the framework as mandatory it is possible to chose which parts should be included and in that way scale the complexity of the resulting model. In a simple implementation it may, for instance, be satisfactory to represent in the information system only the application data pertaining to each role.

As we this far only have defined a number of data structures, it is not possible to make any statements about the formal properties of the models created using our framework. We intend the definitions we have made to serve as a basis for future extensions of the framework. An important refinement would be if constraints could be expressed as general formulae of predicate logic, defined over the components in the framework. This would make it possible to take advantage of work on constraint handling in AI.

The goal for our ongoing research is to create a security management model that is based on roles but that transcends traditional RBAC by integrating aspects from Workflow Management. To provide a workbench for this work we intend to implement a prototype system based on our framework and SESAME [PP95]. SESAME is an implementation of a distributed access control that incorporates a role mechanism, and is based on a set of ECMA[5] standards [ECM].

During this work, the role and function of the session manager needs to be further defined. We will also define an application programming interface for manipulating the organizational model while maintaining consistency under imposed constraints. Before making an implementation it is also necessary to study the nature of the system objects that are to be accessed through RBAC. Areas of special interest are differentiated protection of parts in composite documents and protection differentiated according to the evolutionary state of a document. In connection with our implementation we also need to study further is the issue of *management* raised in [SCFY96].

In [PS95] Parker and Sundt suggest *affiliation* as a third dimension, apart from access identity and role, to be used in access control decisions. We believe that this might be a useful idea, but more work is necessary in order to incorporate it into our framework.

References

[Bid79] Bruce J. Biddle. *Role Theory – Expectations, Identities and Behaviours.* Academic Press, 1979. ISBN 0-12-095950-X.

[CS92] Silvana Castano and Pierangela Samarati. An Object-Oriented Security Model for Office Environments. In Larry D. Sanson, editor, *International Carnahan Conference on Security Technology 26*, pages 146–152. IEEE, 1992.

[DHTK93] S. A. Demurijian, M.-Y. Hu, T. C. Ting, and D. Kleinman. Towards an Authorization Mechanism for User-Role Based Security in an Object-Oriented Design Model. In *Twelfth Annual International Phoenix Conference on Computers And Communications*, pages 195–202, New York, NY, USA, 1993. IEEE. ISBN 0-7803-0922-7.

[ECM] ECMA – Standardizing Information and Communication Systems. URL. http://www.ecma.ch/.

[LMSY95] Emil C. Lupu, Damian A. Marriott, Morris S. Sloman, and Nicholas Yialelis. A Policy Based Role Framework for Access Control. In *First ACM Workshop on Role-Based Access Control*, 1995. URL ftp://dse.doc.ic.ac.uk/dse-papers/management/rbac95.ps.Z.

[Mar93] Damian A. Marriott. Management Policy Specification. Research Report DoC 94/1, Imperial College, Department of Computing, 180 Queens's Gate, London, SW7 2BZ, Uk, Nov 1993. URL ftp://dse.doc.ic.ac.uk/dse-papers/management/policy_spec.ps.gz.

[MD94] Imtiaz Mohammed and David M. Dilts. Design for Dynamic User-Role-Based Security. *Computers & Security*, 13(8):661–71, 1994.

[NO93a] Matunda Nyanchama and Sylvia Osborn. Role-Based Security, Object Oriented Databases & Separation of Duty. *SIGMOD RECORD*, 22(4):45–51, Dec 1993.

[5] ECMA=European Computer Manufacturers Association

[NO93b] Matunda Nyanchama and Sylvia Osborn. Role-Based Security: Pros, Cons & Some Research Directions. *ACM SIGSAC Review*, 2(2):11–17, June 1993. URL http://www.csd.uwo.ca/tech-reports/442/roles.ps.Z.

[PP95] Tom Parker and Denis Pinkas. SESAME V4 – OVERVIEW. URL http://www.esat.kuleuven.ac.be/cosic/sesame/doc-ps.html, Dec 1995.

[PS95] Tom Parker and Chris Sundt. Role Based Access Control in Real Systems. In *First ACM Workshop on Role-Based Access Control*, Dec 1995.

[SCFY94] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control: A Multi-Dimensional View. In *Proceedings of the 10th Annual Computer Security Applications Conference*, Orlando, Florida, Dec 5–9 1994. URL http://www.list.gmu.edu/~sandhu /papers/confrnc/acsac/ps_ver/a94rabca.ps.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, Feb 1996. URL http://www.list.gmu.edu/progress_papers.html.

[SF94] Ravi S. Sandhu and Hal Feinstein. A Three Tier Architecure for Role-Based Access Control. In *Proceedings of the 17th NIST-NCSC National Computer Security Conference*, pages 138–149, Baltimore, MD, October 11-14 1994. URL http://www.list.gmu.edu/~sandhu/papers/confrnc/ncsc/ps_ver/b94rbac.ps.

[Slo94] Morris Sloman. Policy driven management for distributed systems. *Journal of Network and Systmes Management*, 2(4), 1994. URL ftp://dse.doc.ic.ac.uk/dse-papers/management/pdman.ps.Z.