Access Rights Administration in Role-Based Security Systems [1]

Matunda Nyanchama & Sylvia Osborn
The Department of Computer Science
The University of Western Ontario
London Ontario N6A 5B7 Canada
FAX: (519) 661-3515
email:{matunda,sylvia}@csd.uwo.ca

Abstract

This paper examines the concept of role-based protection and, in particular, role organization. From basic role relationships, a model for role organization is developed. The role graph model, its operator semantics based on graph theory and algorithms for role administration are proposed. The role graph model, in our view, presents a very generalized form of role organization for access rights administration. It is shown how the model simulates other organizational structures such as hierarchies [TDH92] and privilege graphs [Bal90].

Keywords: Roles, role-based protection, access control, privilege graph, least privilege, Role Graph.

1 Introduction

Role-based protection is a flexible means of administering large numbers of system privileges especially for large databases. A privilege is a *unit* of access to system information. A role is a named collection of such privileges [Bal90, KM92, NO93b]. User authorization to a role grants the user access to the privileges defined in the role.

The advantage of role-based protection is that it eases the administration of privileges because of the flexibility with which roles may be configured and reconfigured [TDH92, NO93b]. System security is served further when the role configuration process is based on the principle of *least privilege* in which a role is equipped only with sufficient privileges to facilitate the intended duty requirements [Tho91].

In an organization with a large number of diverse duty requirements, the number of roles can proliferate as new roles are defined to meet specific duty requirements. Some roles can have overlapping functions (hence overlapping privileges) while others need not overlap. The need to have some formal manner of tracking the distribution and administration of privileges is important to ensure proper exercise of both responsibility and system security. It is important to have a means of formally expressing role relationships – one which reflects the manner of distribution of privileges in a system.

This paper examines what we consider *basic relationships* that can exist among roles in an organization and their application in modeling role organization. Using these basic relationships as the foundation, a model for role organization is proposed. It is possible for the privilege sets of two roles to completely overlap (one is a subset of the other), partially overlap (have a common subset) or have a common superset. These relationships, along with the concepts of *maximum* and *minimum* privilege sets form the basis of the role graph model. To demonstrate the expressive power of this model, we illustrate how it simulates organizational structures such as hierarchies [TDH92] and privilege graphs [Bal90].

[1] To Appear in Database Security VIII: Status & Prospects, August, 1994.

In the next section we discuss the concepts of privileges, roles and the advantages of role-based protection. We formally define the term *role,* as used in this paper, and motivate the need for formal role organization. In section 3 we discuss the basic relationships that can exist among roles and introduce operators to model these relationships. We regard these relationships as forming the basis of role organization modeling. Section 4 formally presents the role graph model, and gives algorithms for role administration. Section 5 discusses model simulation of other role organizational structures. Section 6 contains the summary and conclusions.

2 Introduction to Roles

2.1 Basic Definitions

The idea of a role arises out of the need to provide duty functionality which is then authorized as a single unit. A role can be seen as a job, office, set of actions of a role-holder, a collection of responsibilities and functions or a collection of privileges pertaining to some duty requirements [DM89, Bal90]. A role exists as an entity separate from the role holder or role administrator. It should be equipped with sufficient functionality to enable an authorized user to achieve the duty requirements associated with the role. Hence a clerical role will be given sufficient access rights to enable an authorized user, or user group, to perform clerical duties. Baldwin [Bal90] terms these **Named Protection Domains** (NPDs). Such a role specification captures the responsibilities, rights and obligations associated with what Dobson and McDermid [DM89] term a *functional role.*

The other important component of role definition is its *structural* [DM89] aspect which captures a role's relationship with other roles. For purposes of this paper, we shall use the term role to refer to the functional aspect while the structural aspect of role relationships will be captured by the structure defining their relationships–in our case a *role graph* model.

A role is defined in terms of privileges. A privilege, on the other hand, is defined in terms of access modes and can be viewed as a unit of access rights administration.

Definition 1 *Privilege: A privilege is a pair (x, m) where x refers to an object and m is a non-empty set of access modes for x.* □

The object referred to by x can be a protected data item, an object-oriented (O-O) class definition or extent, a complex object, a resource (e.g. printer), etc. x can be any name or identifier which uniquely specifies the associated object. m, the set of access modes, is composed of valid modes of access to x. Its specification and administration can be subjected to a range of security policies. In systems with simple access modes such as reads, writes, executes, etc. m, is a subset of these access modes. In complex systems, these access modes can be composed of a series of or nested applications of reads, writes and executes. Where x is an object in an O-O environment, m would be the execute mode of one or more methods. In transactional systems, m would be a list of transactions that facilitate access to x. The exact nature of x and m is a matter of the application environment and the associated security policy [NO93a]. Since privileges are intended for security administration, the security policy must specify how they are administered. In our case, the initialization and modification of a privilege must be authorized.

Definition 2 *Role: A role is a named set of privileges. It is a pair (rname, rpset) where rname is the role name and rpset is the privilege set.* □

A role's name **rname** uniquely identifies a role in a system. We use dot notation to refer to a role's name and privilege set. Thus for a given role r, r.rname and r.rpset refer

to the name of the role and its privilege set, respectively. Let $\mathcal{PV}$ denote the universal set of privileges in a given system, and $\mathcal{R}$ the universal set of roles. We also define a function $\Psi : \mathcal{R} \mapsto \mathcal{PV}$, which enumerates the privileges of a given role, so that for every $r \in \mathcal{R}, \Psi(r) = \{pv_1, \cdots, pv_n\} = r.rpset$.

2.2 Strengths of Role-Based Protection

Role-based protection offers *flexibility* in system privilege administration [TDH92, NO93b]. User access rights can be varied either by *explicit* authorization (or revocation of authorization) of a user to a role or by indirectly varying the role privilege set. Further advantage is gained if users are organized into groups such that authorizations are given to groups, as opposed to individuals.

Given that system privileges can be very fine-grained, roles offer a means of managing them incrementally. Considering the manner in which privileges can be assigned/revoked to/from a given role, this method approaches a continuum in system privilege administration [NO93b]. A related advantage is that role-based protection can be used to enforce the principle of *least privilege* where a role is defined to have only the necessary functionality required for the associated duties [Tho91].

This approach offers a simplification of the complexity of system privilege management. With a suitable organizational framework capturing role relationships, it is possible to analyze the implications of given authorizations. Moreover, such a formal framework lends itself to the development of analytical tools. It is also possible that management tools for access rights administration can be used in role management.

Given that role-based protection is designed with a given application in mind, this method provides a chance for incorporation of application level security constraints and semantics [Tho91]. An associated advantage is that roles allow for *multidirectional* information flow policies [Tho91] unlike such models as Denning's lattice model [Den76] and Bell and La-Padula's [BL75] multilevel model. As well, unlike these traditional models which specify what information flows *should not* take place, role-based protection affirms which information flows *can* take place [GMP92].

2.3 Roles & Access Rights Administration

Roles act as *gateways* to system information. The privilege set of a given role determines what information is available via the role. One advantage of role-based protection mentioned in the previous section is that access to system information is accomplished at two levels: via explicit authorization to a role or via inclusion of some privilege in a role. We term the former *user-role* authorization while the latter is termed *role-privilege* authorization (see figure 1). A third form of authorization is *role-role* authorization [Bal90] in which one role is authorized another's privileges. We address each of these in turn.

In **user-role** authorization, a user/group is authorized access to system privileges available via the role. Such authorization must be specified in a role's access control list. For each role, such an access control list contains the user identifier for each user authorized to the role.

Let $\mathcal{UID}$ be the set of all user identifiers, and $\mathcal{GID}$ the set of all group identifiers; $\mathcal{ID} = \mathcal{UID} \cup \mathcal{GID}$.

Definition 3 *Access Control List: A role access control list (racl) is of the form:* $[id_1, \cdots, id_n]$, *where* $id_i \in ID$. □

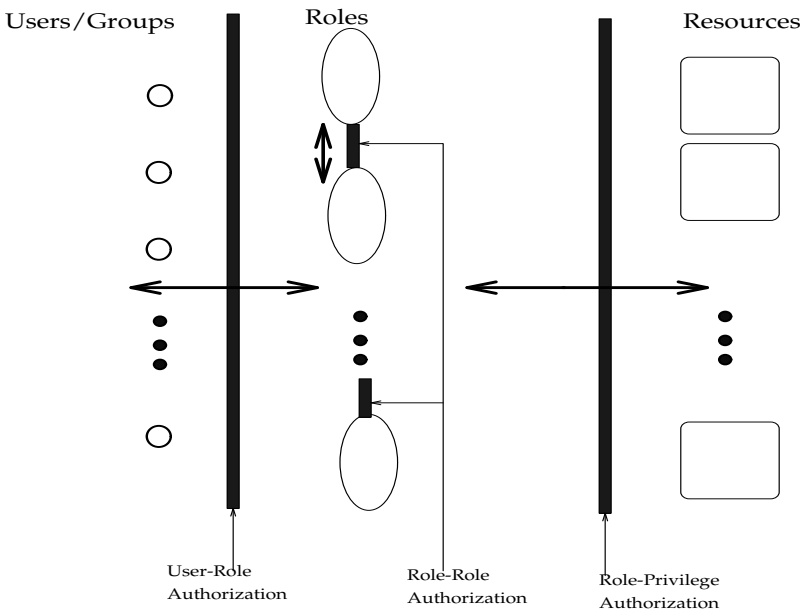

Figure 1: Three Kinds of Authorizations

In a secure system all roles must have access control lists, i.e. $\forall r \in \mathcal{R}, \exists$ r.racl $= [\cdots, id_i, \cdots]$. A role with an associated access control list is called a *secure role.*

Definition 4 *Secure Role: A secure role is a named collection of privileges along with its access control list. It is a triple (rname,rpset,racl), where rname is the role name, rpset is its privilege set and racl is its access control list.* □

Role-privilege authorization involves role configuration in which a privilege is added to the role's privilege set. **Role-role** authorization [Bal90] forms the third kind of authorization. If a role A is authorized to access a role B, it means that all of B's access rights are available via role A. In other words, B's privileges are a proper subset of the privileges of A. Role-role authorization is an aspect of role structure.

Example 1 Suppose we have two roles: **clerk** and **supervisor** in which the **supervisor** role has a role authorization to the **clerk** role. This means that the clerk's access rights are available to the supervisor. A user authorized to the **supervisor** role can perform whatever a user authorized to the clerk role can do.[2] We can view the privilege relationships between the two roles as $\Psi(clerk) \subseteq \Psi(supervisor)$. □

This paper examines role-role authorizations which define role relationships. These have implications on role organization and access rights administration. Role-role authorizations can be complex. To capture the role-relationships completely and be able to carry out an analysis of the implications of privilege assignment and distribution in a system can be very complex without some formal organizational structure. Complexity of analysis of system privilege distribution is one short-coming of role-based protection [TDH92, NO93b].

Baldwin's approach to access rights administration uses privilege graphs (PG) which capture functionality, structure and authorizations. A PG (figure 2) is an acyclic graph with three types of nodes: functionality, role and user/group. A path from a given user node to a functionality node means that the user is authorized to execute the functionality. The

[2] Separation of duty [CW87], on the other hand, ensures that the supervisor does not perform both roles.

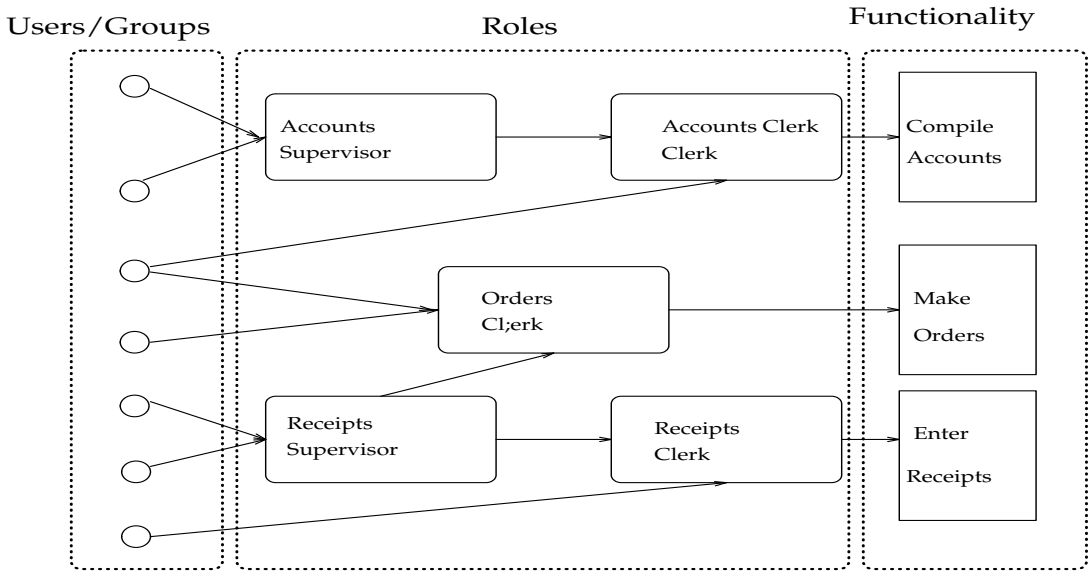

Figure 2: Baldwin's Privilege Graph

access rights available to such a user are all the privileges specified in roles on any such path. Ting et.. al.'s [TDH92] approach utilizes hierarchical ordering of roles in which for any given roles in a path, those lower in the hierarchy have lower functionality than those high in the hierarchy. In general, the path captures a subsetting relationship between the roles such that for a given directed edge $\langle v_i, v_j \rangle$, $\Psi(v_j) \subseteq \Psi(v_i)$. Both of these structures have what we term the *acyclicity property*.

Definition 5 *Acyclicity Property: A role organization structure is said to have the acyclicity property if in a graph of the role relationships, with the roles as nodes, we have a directed edge* $\langle r_i, r_j \rangle$ *whenever* $\Psi(r_i) \subseteq \Psi(r_j)$ *and the graph is acyclic.* □

Property 1 *Role Organization Structure Acyclicity: A role organization must preserve the acyclicity property in order to offer differentiated access to system information via role-based protection techniques.* □

3 Modeling Role Organization

A role is a collection of privileges which facilitates the execution of some *functionality* for an authorized user. Roles in a system can have different kinds of relationships among them based on their associated functionalities and organizational constraints. Thus it is important to develop some formal organizational framework which expresses desirable properties for an enterprise whose security is being enforced and, in the process, captures the relationships among roles. Such a framework will facilitate the analysis of privilege distribution and sharing.

In this section, we discuss and model basic role relationships which form the basis of a role organization framework. We start with relationships between two roles and introduce the concepts of the minimum and maximum privilege sets in a role-based system and their relationship with other roles. Finally, we combine these concepts to yield a framework for role organization.

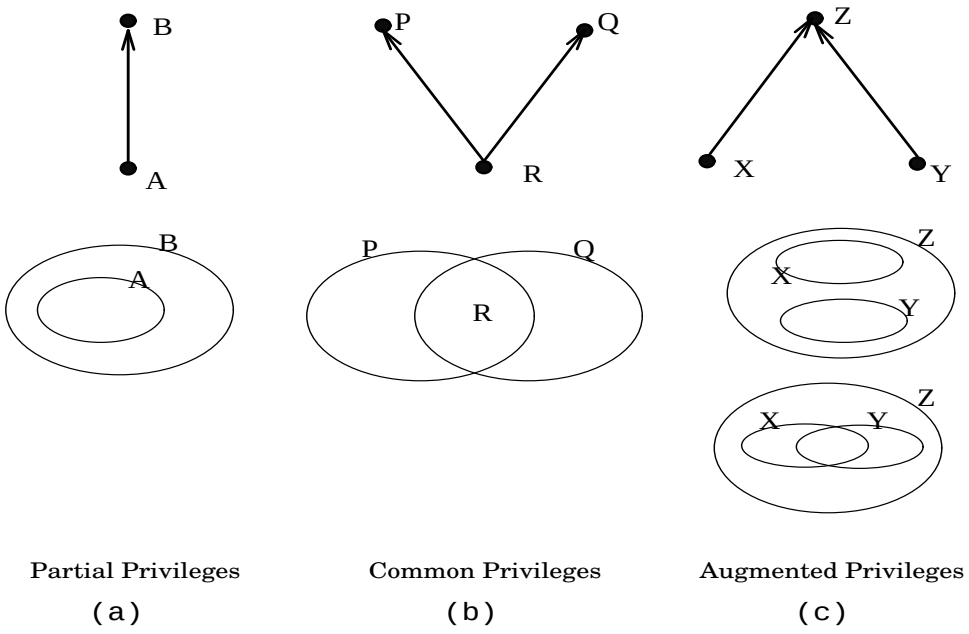

Figure 3: Three Kinds of Basic Role Relationships

3.1 Basic Role Relationships

We identify three kinds of basic relationships: *junior-senior, common "junior"* and *common "senior"*. The *junior-senior* relationship, expressed as **junior→senior**, captures the fact that the senior role's privileges include those of the junior one. A role is a *common junior* of two other roles if it shares some privileges with both of these senior roles. A role which encompasses all of the privileges of two junior roles is called a *common senior* to these roles. Figure 3 shows these three possibilities with Venn diagrams over the associated privileges. In all cases, there is privilege and functionality sharing between two roles.

1. **Partial Privileges**

   With partial privilege sharing, privileges defined in one role are a complete subset of privileges in another role. This implies shared functionality via the shared privileges. For instance, the **clerk** and **supervisor** roles in example 1 share the functionality associated with the **clerk** role, i.e. a user authorized to the **supervisor** role can execute the functionalities associated with both roles (figure 3a).

   We model such *direct* functionality and privilege sharing using the **is-junior** relationship denoted by "→". In our example, **clerk→supervisor**. In general, given two roles $r_i, r_j \in \mathcal{R}$ with $r_i \rightarrow r_j$, we have the following interpretation:

   > *r_i and r_j are "junior" (subservient) and "senior" (superior) roles, respectively. Moreover, r_i's privileges and functionality are available to r_j. Hence $\Psi(r_i) \subseteq \Psi(r_j)$. We say r_i's privileges are* **indirectly** *available to r_j.*

   Definition 6 *is-junior relationship ($\rightarrow$): An* is-junior *relationship exists between two roles r_i and r_j, denoted $r_i \rightarrow r_j$, if and only if $\Psi(r_i) \subseteq \Psi(r_j)$.* □

   The *is-junior* relationship can be seen as a role-role authorization in which the superior role is authorized to the privileges of the junior role.

   If we consider relative authority as a measure of the privileges associated with a role, then the *is-junior* relationship can be seen as specifying which of the two roles has a

higher authority than the other. In our case the junior role exercises less authority than the superior one. Moreover, the *is-junior* relationship can be seen as specifying the flow of authority in which the senior role exercises more authority than the junior one. Further, for this authority to be meaningful, this relationship must be *acyclic*; it must preserve property 1.

2. **Common Privileges**

   Another form of relationship between two roles is where there is privilege sharing in which roles have a non-empty intersection of their privilege sets but with neither of the sets being a subset nor a superset of the other. Such a relationship can be used to express an overlap of responsibility (figure 3b).

   If there exists a role defined whose privilege set is some or all of this intersection, then we say such a role is a **common-junior** of the other two roles. We denote the *common-junior* relationship by "$\odot$". In general, $r_i \odot r_j$ is not unique. Suppose we have roles A, B and C related as $C \in A \odot B$. Suppose the privilege sets associated with A and B are $\Psi(A)$={1,2,3,4} and $\Psi(B)$ ={3,4,5,6,7}, respectively. $\Psi(C)$ must be a common subset of both $\Psi(A)$ and $\Psi(B)$, i.e. $\Psi(C) \subseteq (\Psi(A) \cap \Psi(B)) = \{3,4\}$.

   In general, given three roles $r_i, r_j, r_k \in \mathcal{R}$ and $r_k \in r_i \odot r_j$, we have the following interpretation:

   > *both r_i and r_j are senior (superior) roles to r_k. Moreover, r_k's privileges and functionality are indirectly available to both r_i and r_j. Hence $\Psi(r_k) \subseteq \Psi(r_i)$ and $\Psi(r_k) \subseteq \Psi(r_j)$.*

   Definition 7 *common-junior relationship ($\odot$): Given roles r_i and r_j, $r_i \odot r_j$ is all r_k such that $\Psi(r_k) \subseteq (\Psi(r_i) \cap \Psi(r_j))$.* □

3. **Privilege Augmentation**

   Another important consideration is privilege augmentation. In analyzing privilege distribution it may be necessary to find a role that embodies the functionality and privileges of two given roles. Such a role's privileges will be a superset of both given roles (figure 3c).

   The relationship in such a case is termed **common-senior** and is denoted by "$\oplus$". In general, $r_i \oplus r_j$ is not unique. Suppose we have roles X, Y and Z related as $Z \in X \oplus Y$. Let $\Psi(X)$={1,2,3,4} and $\Psi(Y)$={6,7,8,9}. For Z's privileges to be a common superset of those of X and Y, we must have $(\Psi(X) \cup \Psi(Y)) \subseteq \Psi(Z)$, i.e. {1,2,3,4,6,7,8,9} $\subseteq \Psi(Z)$.

   Given three roles $r_i, r_j, r_k \in \mathcal{R}$ and $r_k \in r_i \oplus r_j$, we have the following interpretation:

   > *both r_i and r_j are junior (subservient) roles to r_k. Moreover, both r_i's and r_j's privileges and functionalities are indirectly available to r_k. Hence $\Psi(r_i) \subseteq \Psi(r_k)$ and $\Psi(r_j) \subseteq \Psi(r_k)$.*

   Definition 8 *common-senior relationship ($\oplus$): Given roles r_i and r_j, $r_i \oplus r_j$ is all r_k such that $(\Psi(r_i) \cup \Psi(r_j)) \subseteq \Psi(r_k)$.* □

The foregoing relationships can be extended to cater for more than two roles.

1. **Partial Privilege Sharing**

From the definition of the *is-junior* relationship, if $(r_i \rightarrow r_j)$ and $(r_j \rightarrow r_k)$ then it must also be true that $r_i \rightarrow r_k$ since $(\Psi(r_i) \subseteq \Psi(r_j)) \wedge (\Psi(r_j) \subseteq \Psi(r_k)) \Rightarrow (\Psi(r_i) \subseteq \Psi(r_k))$. This then captures the *transitive* property of the *is-junior* relationship. In general, if we have a role relationship of the form: $r_i \rightarrow r_{i+1} \rightarrow \cdots \rightarrow r_{i+n}, n \geq 0$, it follows that $\Psi(r_i) \subseteq \Psi(r_{i+1}) \subseteq \cdots \subseteq \Psi(r_{i+n})$. This captures the monotonic increasing property of the privilege function for roles related via the *is-junior* relationship.

Property 2 *The privilege function Ψ increases monotonically with respect to the* is-junior *$(\rightarrow)$ relationship.* □

We denote $r_i \rightarrow r_{i+1} \rightarrow \cdots \rightarrow r_{i+n} \rightarrow r_j$ by $r_i \rightarrow^* r_j$ for $n \geq 0$ and $r_i \rightarrow^+ r_j$ for $n > 0$. This leads to the concept of a path:

Definition 9 *Role Path: A role path, p, between two roles r_i and r_j is of the form $r_i \rightarrow^* r_j$. A trivial path exists between a role and itself.* □

Other properties of the *is-junior* relationship include *reflexivity* and *antisymmetry*. Given roles r_i and r_j, we have $r_i \rightarrow r_i$ (reflexivity) since $\Psi(r_i) \subseteq \Psi(r_i)$. As well, $((r_i \rightarrow r_j) \wedge (r_j \rightarrow r_i)) \Rightarrow r_i = r_j$. This follows from the observation that $(r_i \rightarrow r_j) \Rightarrow \Psi(r_i) \subseteq \Psi(r_j)$ and $(r_j \rightarrow r_i) \Rightarrow \Psi(r_j) \subseteq \Psi(r_i)$. With $\Psi(r_i) \subseteq \Psi(r_j)$ and $\Psi(r_j) \subseteq \Psi(r_i)$ and by the acyclicity property, it follows that $\Psi(r_i) = \Psi(r_j)$, which implies $r_i = r_j$. This is the basis of the following property:

Property 3 *Role Privilege Set Uniqueness: A Role's privilege set must be unique.* □

2. **Common Privileges**

From the *common-junior* $(\odot)$ relationship above, observe that the common subset of two roles need not be an immediate junior role of both roles in question. The following lemma expresses the relationship between the *is-junior* and the *common-junior* operators, $\rightarrow$ and $\odot$, respectively:

Lemma 1 *If $r_k \in r_i \odot r_j$, then $r_k \rightarrow^+ r_i$ and $r_k \rightarrow^+ r_j$.* □

The *common-junior* operator $(\odot)$ is commutative, associative and reflexive; i.e. $r_i \odot r_j = r_j \odot r_i$, $r_i \odot (r_j \odot r_k) = (r_i \odot r_j) \odot r_k$ and $r_i \odot r_i$ is defined and includes r_i.

3. **Privilege Augmentation**

As with the *common-junior* relationship, the *common-senior* relationship need not involve immediate superiors of the role under consideration. The following lemma captures the relationship between the two operators $\rightarrow$ and $\oplus$:

Lemma 2 *If $r_k \in r_i \oplus r_j$, then $r_i \rightarrow^+ r_k$ and $r_j \rightarrow^+ r_k$.* □

The operation $\oplus$ is commutative, associative and reflexive i.e. $r_i \oplus r_j = r_j \oplus r_i$, $r_i \oplus (r_j \oplus r_k) = (r_i \oplus r_j) \oplus r_k$ and $r_i \oplus r_i$ is defined and includes r_i.

3.2 The Concepts of Minimum and Maximum Privilege Sets

It is possible that an organization provides a minimum set of privileges available to every user. Such a basic privilege set, for instance, can be things like the ability/permission to log onto a computer system, the privilege to get into certain areas of an organization's premises, etc. In general, this minimum privilege set represents the very minimum that any valid user can be authorized to.

Since users are authorized to specific roles, it is possible to organize such a basic set of privileges into a role such that they are available via explicit authorization or via role relationships with other roles. We denote the role with the basic privilege set **MinRole**. In general, depending on a particular organization, **MinRole**'s privilege set can be empty.

$$\Psi(\text{MinRole}) = \begin{cases} \text{Minimum mandatory privilege set} & \text{if defined} \\ \emptyset & \text{otherwise} \end{cases}$$

For all $r \in \mathcal{R}, \text{MinRole} \rightarrow^{+} r$ holds.

Property 4 *Minimum Privilege Property:* **MinRole** *is always defined.* □

With the introduction of **MinRole**, there is always at least one *common-junior* for all roles, namely **MinRole**.

As with **MinRole**, we envisage **MaxRole**, some system "chief executive" role, which embodies the collection of all privileges in a given system. Theoretically, a user authorized to **MaxRole** can execute any functionality using the associated privileges in whatever role they are specified. Unlike Ψ(MinRole) which can be empty, Ψ(MaxRole) can never be empty if the system is intended to accomplish anything at all.

$$\Psi(\text{MaxRole}) = \bigcup_{r \in \mathcal{R}} \Psi(r)$$

For all $r \in \mathcal{R}, r \rightarrow^{+} \text{MaxRole}$ holds.

Property 5 *Maximum Privilege Property:* **MaxRole** *is always defined.* □

With the introduction of **MaxRole**, there is always at least one *common-senior* for two roles, namely **MaxRole**.

The *is-junior, common-junior* and *common-senior* relationships introduced in the previous section capture all manner of relationships that can be used to associate two or more roles when there is need for analysis of their interaction. **MinRole** and **MaxRole** express the concepts of minimum mandatory and maximum privilege sets, respectively, in a system. Combining these yields representations such as those in figure 4.

For the purposes of security and the need for dispersion of powers, MaxRole may not be authorized to any one individual in an organization. In an ideal situation, MaxRole conceptually corresponds with the role of a Chief Executive in an organization. It is unlikely that an administrative or a security policy would advocate such singular exercise of powers. Moreover, there is a very realistic risk that allowing exercise of privileges of MaxRole can compromise the system. However, such problems need not arise if we make the exception that no single user can exercise the privileges of MaxRole. This will make MaxRole a non-executable role. Other policies may choose a collective execution of the role, e.g. by a number of votes of authorized users. Whatever the case, authorization to MaxRole with be a matter of a specific security policy. MaxRole, in our modeling, is useful for purposes of completeness. It ensures that every two roles in the system have a common-senior just as MinRole ensures that every two roles have a common-junior.

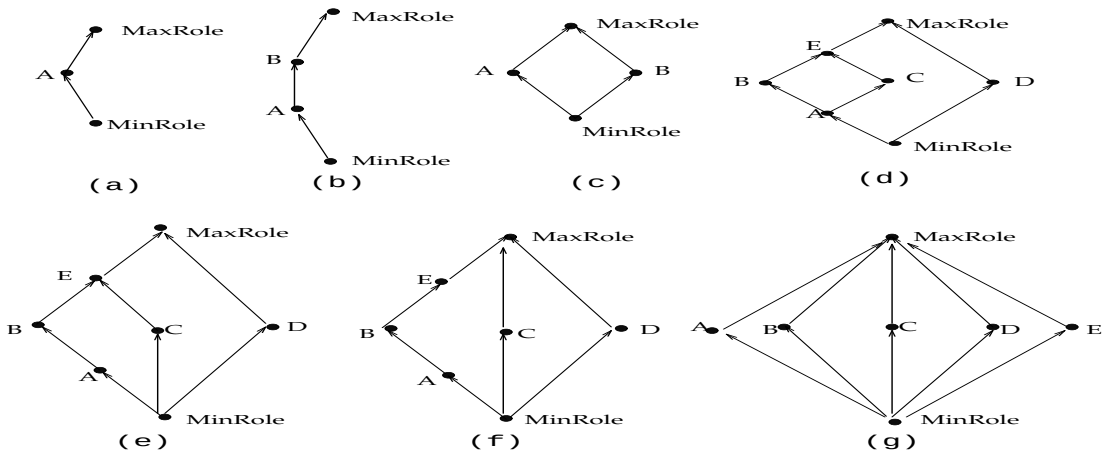

Figure 4: Different Forms of Role Organization

4 A Role Graph Model for Role Organization

The basic role relationships discussed in section 3 point to an acyclic role graph organization for roles. In this section we develop the modeling further using graph theory. We present a role graph model for role organization and develop algorithms for the management of roles and their relationships.

4.1 The Model: Informally

To minimize the task of enumerating the privileges of each role, we organize them using the concepts introduced in section 3 which incorporate acyclicity of the role graph structure and the monotonicity of role privileges for any path. Such a structure, along with rules for role ordering and determining the privileges associated with a role, facilitate a simple, yet elegant, organization of roles to reflect the *authority*[3] attached to each role. Role ordering and role inter-relationships, in turn, offer a means of distributing privileges among the roles. The idea is that we explicitly assign a privilege at the lowest point in the role graph where it is desirable. Since our formulation specifies that high order roles can execute the privileges of the lower order ones with a connecting path, we can make the least number of explicit privilege assignments that would facilitate the desired distribution.

From the ordering, we define *authority paths* that are linear (total) orders of roles according to increasing authority, connected by the *is-junior* ($\rightarrow$) relationship which can be seen to be specifying the flow of authority. In essence, the ordering asserts the fact that higher authority roles have access to more privileges than lower ordered ones in any given path. The effective privileges associated with a role result from those privileges *directly* associated with the role and those *indirectly* associated with it. The former are those privileges *explicitly* specified in the role while the latter are those privileges specified in lower order roles connected by a path to the role.

4.2 The Role Graph Model: Formally

This section presents the formal organization of roles into a role graph $RG = (\mathcal{R}, \rightarrow)$, as shown in figure 5. The nodes of the graph correspond to the roles given, and include **MaxRole** and

[3] Our use of this term will become clear as we advance.

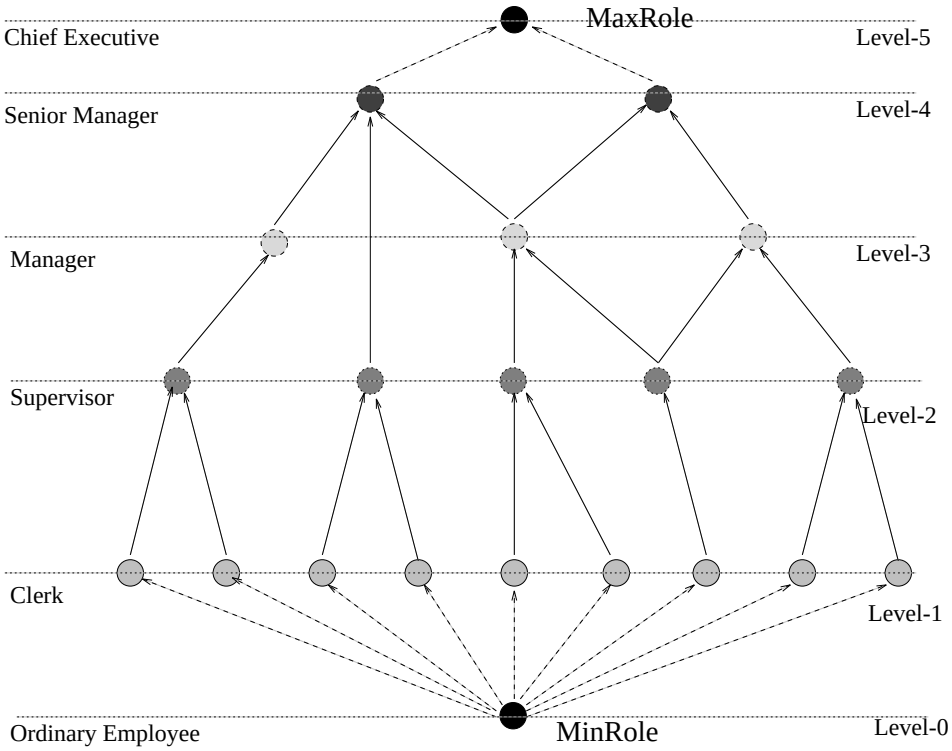

Figure 5: Example of Role Graph

MinRole. $\mathcal{R} = \{r_1, r_2, \cdots, r_n, \text{MaxRole}, \text{MinRole}\}$. The edges are defined by the *is-junior* relationship. Note that by the definition of privileges for **MaxRole** and **MinRole** and the definition of *is-junior*, there is an edge from **MinRole** to every r_i, and an edge from every r_i to **MaxRole**. The common-junior and common-senior relationships, ($\odot$ and $\oplus$) still have the same meaning as previously.

Note that if a system administrator is specifying roles, it is possible that the privileges are specified in a highly redundant fashion. In other words, rather than specifying the minimum set of direct privileges for a role, some indirect privileges might be given as being direct privileges. The function $\Psi(r)$ returns the set of all direct and indirect privileges of a role, which we also call the *effective* privileges. The version of the graph which we will present to the role administrator should *neither* have redundant privilege specifications *nor* redundant *is-junior* relationships (i.e. redundant graph edges), in order to highlight the true nature of the role relationships. We will further explain this reduced form of the graph shortly.

Paths in the role graph not involving **MaxRole** and **MinRole** are of more interest to us. Consequently, we shall use the following role graph path definition in the subsequent sections.

Definition 10 *Role Graph Path: A role graph path, p, is of the form* $r_i \rightarrow r_{i+1} \rightarrow \cdots \rightarrow r_{i+n} \rightarrow r_j, n \geq 0$ *such that* $r_i \neq MinRole \wedge r_j \neq MaxRole$. □

The quadruple $(\mathcal{R}, \rightarrow, \oplus, \odot)$ which includes **MaxRole** and **MinRole**, specifies an *authority structure* for roles. For any role graph path of the form $r_i \rightarrow \cdots \rightarrow r_n, n \geq 1$ we have an authority relation of the form $r_1 < \cdots < r_n$, with the authority embodied in the roles on a path totally ordered. In general, given any two roles $r_1, r_2 \in \mathcal{R}$, $r_1 < r_2$, $r_2 < r_1$ or they are incomparable. Where there is a path (call it an *authority flow path*), the roles in the path form a total order.

Definition 11 *Path Role Set: The role set of a given path, denoted by* $\Gamma(p)$*, is the set of all roles that compose the path. We say that a given role participates in a path if it belongs to the path's role set.* □

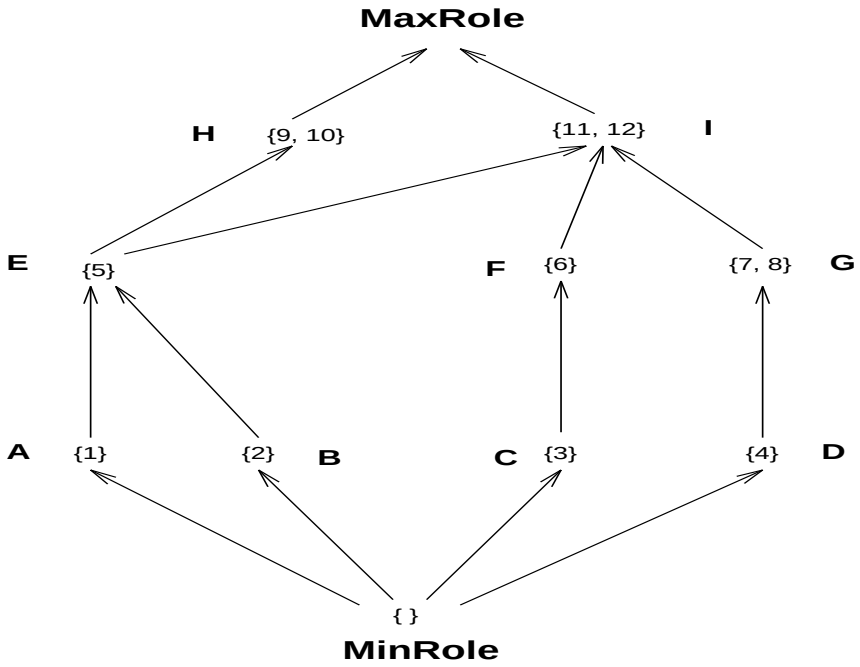

Figure 6: Role Graph with Privileges

Privilege Distribution Table
For Figure 6

Role Name	Direct (D)	Indirect (I)	Effective $(D \cup I)$ Ψ
A	{1}	{ }	{1}
B	{2}	{ }	{2}
C	{3}	{ }	{3}
D	{4}	{ }	{1}
E	{5}	{1,2}	{1,2,5}
F	{6}	{3}	{3,6}
G	{7,8}	{4}	{4,7,8}
H	{9,10}	{1,2,5}	{1,2,5,9,10}
I	{11,12}	{1,2,3,4,5,6,7,8}	{1,2,3,4,5,6,7,8,11,12}

Table 1: Table of Privileges

We extend the function Ψ to paths as follows: for a path p, $\Psi(\Gamma(p)) = \bigcup_{r_i \in p} \Psi(r_i)$.

Definition 12 *Path Independence: Let p_i and p_j be two paths in a role graph. We say p_i is independent of p_j if $\Psi(\Gamma(p_i)) \cap \Psi(\Gamma(p_j)) = \Psi(MinRole)$.* □

In other words, the two role sets are related only via **MaxRole** and **MinRole**. Such independence can be exploited to prohibit privilege sharing by ensuring that the privilege sets of two independent paths are disjoint.

Example 2 Consider figure 6 where we have distinct privileges numbered $1, \cdots, 12$ with privileges $1, \cdots, 6$ directly assigned to roles $A, \cdots, F$ and $\{7,8\}, \{9,10\}, \{11,12\}$ assigned roles G, H, I respectively. We have a role graph specification as follows: $A \rightarrow E, B \rightarrow E, C \rightarrow F, D \rightarrow G, E \rightarrow \{H, I\}, \{F, G\} \rightarrow I$, MaxRole $= H \oplus I$, and MinRole $= A \odot B \odot C \odot D$ with $\Psi(\text{MinRole}) = \emptyset$.

From this we can compute the privileges of various roles and obtain the privileges distribution as in table 1. Moreover, we have the following relationships relating to the $\odot, \oplus, \rightarrow$ operators:

1. The *common-junior* operator, $\odot$, defines a common subset of privileges for any two roles. Consider $E \in H \odot I$ and note that $\Psi(H \odot I) = \Psi(E) = \{1,2,5\} = \Psi(H) \cap \Psi(I)$.

2. The *common-senior* operator, $\oplus$, defines the union of privileges of two roles and as such is a common superset for any two roles. Consider $I \in F \oplus G$ and note that $\Psi(F \oplus G) = \Psi(F) \cup \Psi(G) = \{3,4,6,7,8\} \subseteq \Psi(I) = \{3,4,6,7,8,11,12\}$.

3. The *is-junior* operator, $\rightarrow$, defines a proper subset relationship between two roles, e.g. $E \rightarrow H$. Note that $\Psi(E) = \{1,2,5\} \subset \{1,2,5,9,10\}$. This is true for all roles related via the *is-junior* relationship.

4. Paths $A \rightarrow E \rightarrow H$ and $C \rightarrow F$ are independent paths since their roles sets {A,E,H} and {C,F} are mutually exclusive and the two paths are related via only via **MaxRole** and **MinRole**.

□

The role graph in figure 6 shows only *direct* (non-redundant) privileges for each node, and has no redundant edges. Specifying a role's direct privileges and its *is-junior* relationships with other roles completely specify its effective privileges.

Definition 13 *Direct Privileges: Let $Direct(r)$ denote the direct privileges of a role; i.e. $Direct(r) \subseteq \Psi(r)$ such that for all $r_i \rightarrow r$, $\Psi(r_i) \cap Direct(r) = \emptyset$.* □

For the purpose of the algorithms below, assume that for each role in a role graph, we keep $Direct(r)$ and *is-junior* relationships. By the definition of *is-junior*, the edge set in the role graph will in fact be highly redundant. What we want to present to the role administrator, and maintain, is the transitive reduction of the graph [AGU72]. The transitive reduction of an acyclic graph is a graph in which there are no edges $r_i \rightarrow r_j$ whenever there is a path $r_i \rightarrow^+ r_j$ in the graph. Inputs to and outputs of the algorithms assume *well-formed graphs.*

Definition 14 *Role Graph Well-Formedness: A role graph is well-formed if it is a transitive reduction and if the direct privilege set associated with each role r conforms to the definition of $Direct(r)$.* □

By the original definition of the edge set (based in turn on the *is-junior* relationship which depends on the effective privilege sets of nodes), a path $r_i \rightarrow^+ r_j$ exists in the well-formed role graph whenever $\Psi(r_i) \subseteq \Psi(r_j)$. The following terms will be useful in the algorithms to be presented below:

Definition 15 *Juniors(r) The set of Junior roles for a given role r is all r_i such that $r_i \rightarrow^+ r$.* □

Definition 16 *Seniors(r)The set of Senior roles for a given role r is all r_j such that $r \rightarrow^+ r_j$.* □

Constraint 1 *Role Graph Privilege Set Invariant Constraint: The effective privilege set of every role in a role graph remains invariant unless altered by the system security officer, SSO.* □

The SSO exercises privileges like any other system user by executing in an authorized security administration role. This can be seen as the security information administration role. However, care must be taken to ensure there is no conflict of interest. Hence no one user, whether SSO or not, should be able to administer security information pertaining to one's access rights.

4.3 Role Graph Maintenance Algorithms

We are now ready to introduce some algorithms to assist a role administrator in specifying and modifying a collection of roles. These will ultimately be incorporated in a role maintenance tool.

Our goal is to have all the operations map a well-formed role graph to another well-formed role graph. We assume that the administrator begins with a graph containing only **MaxRole** and **MinRole**. Any direct privileges defined for **MinRole** can be specified at this time.

The role graph can be expanded at any point by adding new roles as need may arise while retaining the role graph structure. This strategy offers a flexible manner of introducing new privileges into the role graph. Such privileges can be incorporated into an existing role graph by introduction of new roles or by increasing the privileges of existing ones. New roles can be introduced by the addition of completely new roles, or by partitioning existing roles either horizontally or vertically. We also consider role deletion. In all these cases, we can have an increment or decrement in the overall privileges associated with paths in which the affected role participates. Such privileges can remain invariant, be reduced or be increased depending on the operation. Given the space constraints here, we address the cases where (1) path privileges are introduced with the addition of a new role, (2) path privileges may or may not remain invariant with the deletion of a role, (3) path privileges are partitioned with the horizontal partition of a role and (4) privileges remain invariant with the vertical partition of a role.

Consequently, after carrying out the operations on the graph, our procedures will confine themselves with the immediate neighbourhood of the target role. In other words we look for redundant arcs generated due to the operation in question. This involves the immediate senior and immediate junior role sets of the roles affected by the operations.

4.3.1 Role Addition & Deletion

By role addition we mean the creation and incorporation of a totally new role into the role graph. Such a role is defined (name and privilege set) before being integrated into the role graph. While the integration process must preserve the role definition, it is important to ensure that if there are privileges defined in the new role that exist in junior roles in the target paths, they must be removed to take away the redundancy. To introduce such a role requires the specification of the target paths and the position in the paths. This involves the specification of the target superior and junior role(s) for the role to be added (see figure 7a).

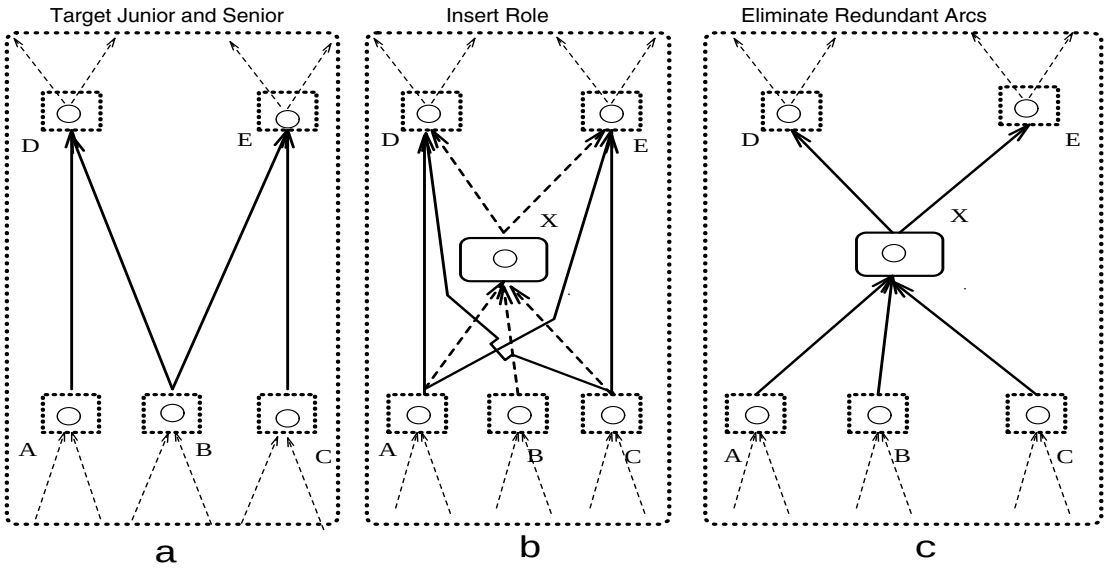

Figure 7: Role Addition

The role to be inserted is added to the node set of the graph, and the appropriate edges are created to indicate the immediate junior and superior roles. It is possible that a node already exists with the same effective privilege set. Once this possibility has been eliminated, redundant paths are removed from the resulting structure. Finally, privilege resolution is done to remove redundant privileges from *Direct* of the new node, and privileges in nodes in Seniors of the the new node made redundant by this insertion. Note that for a node r, the set Seniors(r) can be enumerated by a depth-first search in the role graph starting at node r [CLR90, Man89]. Similarly, the set Juniors(r) can be computed by a depth-first search of the graph formed by reversing the edges in the role graph, again starting the search at node r. The details of these operations will not be given here.

Algorithm 1 in figure 8 and also figure 7 illustrate the role addition process.

The flip side of *role addition* is *role deletion* which involves the elimination of a role from the role graph. This process requires specifying the target role and short-circuiting it by making the target's immediate subservient role(s) the immediate subservient role(s) of the target's immediate superior(s). In doing so, the privileges associated with the deleted role can either be eliminated or distributed. Privilege elimination involves overall privilege reduction of the path associated with the role so deleted.

Retaining the privileges of the deleted role, on the other hand, requires a specification of how these privileges will be distributed among the existing roles. It is reasonable to assume that such role deletion would not affect the effective privilege sets of any superior roles of the deleted role. Hence such privileges must be transferred to the immediate superiors. This would ensure path privilege invariance. This case is illustrated pictorially in figure 9. See the associated algorithm 2 of figure 10.

Example 3 Suppose our target role for deletion is role D in figure 9a with the constraint that all existing paths must keep their privilege sets invariant. For this purpose we choose to shift the privilege set of the target role to its superiors.

To achieve this, first transfer the privileges from role D to both F and G which are both superior to D. This results in roles roles FX and GX which we make immediate superiors of both A and B. The previous edges incident to role D, i.e. $A \rightarrow D \rightarrow F, A \rightarrow D \rightarrow G, B \rightarrow$

Algorithm 1 Role_Addition(rg, target, s.target.set,j.target.set)

```
/* For the addition of a given role into a role graph */
Input: rg = ⟨R, →⟩ (the role graph), target role to be added (role name along with its proposed direct privilege set),
    s.target.set (immediate superior set for the target), j.target.set (immediate junior set for the target),
Output: The role graph with target added and overall privileges of other roles left intact.
    Var r, r_j, r_s: roles;
    Begin
        If ∃(r_s →+ r_j) for any r_s ∈ s.target.set, r_j ∈ j.target.set
            Then abort                                          /* Must not violate acyclicity */
        Else Begin
            1.  Ψ(target):= (∪_{r∈j.target.set} Ψ(r)) ∪ Direct(target);
                    /* Compute the effective privileges of target role */
            2.  If Ψ(r) = Ψ(target) for any r ∈ R
                    Then target := r;                           /* Role privilege sets must be unique */
            3.  If ∃(target →+ r_j) for any r_j ∈ j.target.set
                    Then abort                                  /* Must not violate acyclicity */
                Else Begin
                    a.  R := R ∪ target;                        /* Add target to system roles */
                    b.  For all r_s ∈ s.target.set do add the edge target → r_s;
                    c.  For all r_j ∈ j.target.set do add the edge r_j → target;
                    d.  If for any r ∈ R, Ψ(r) ⊂ Ψ(target) and NOT(r →+ target)
                            Then add the edge r → target;       /* Add this inferred edge */
                    e.  If for any r ∈ R, Ψ(target) ⊂ Ψ(r) and NOT(target →+ r)
                            Then add the edge target → r;       /* Add this inferred edge */
                    f.  Rem_Red_Arcs(rg, j.target.set, s.target.set, target);
                    g.  Red_Priv_Res(rg, j.target.set, target);
                end;
            4.  For all r, r_i, r_j ∈ R if Ψ(r_i) = Ψ(r_j) then  /* Remove any duplicate roles */
                Begin for all r_i → r do add the edge r_j → r
                    for all r → r_i do add the edge r → r_j
                    Delete all edges r_i → r and r → r_i;
                    Remove r_i; end;
        end;
    end.                                                        /* Role_Addition */

    Procedure Rem_Red_Arcs(var rg: role graph; j.target.set, s.target.set: role_set; target: role );
    /* Removes redundant arcs in the immediate neighbourhood of target role */
    Var r_k, r_j, r_s: roles;
    Begin  1.  For all r_j ∈ j.target.set do                    /* Remove direct paths where there is another path */
                   if ∃(r_j → r_k → ··· → target) then
                       Delete the edge r_j → target             /* delete the direct edge*/
           2.  For all r_s ∈ s.target.set do
                   if ∃(target → r_k → ··· → r_s) then
                       Delete the edge target → r_s             /* delete the direct edge*/
    end;                                                        /* Red_Red_Arcs */

    Procedure Red_Priv_Res(var rg: role graph; j.target.set: role_set; target: role);
    Var pv : privilege; r : role;
    Begin  1. For all r in Seniors(r) do                        /* remove redundant privileges from senior roles. */
                   For all pv ∈ Direct(target) do
                       if pv ∈ Direct(r) then
                           Direct(r) := Direct(r) − pv;
           2. For all r in j.target.set do                      /* remove redundant privileges from Direct(target).
                   For all pv ∈ Ψ(r) do
                       if pv ∈ Direct(target) then
                           Direct(target) := Direct(target) − pv;
    end;                                                        /* Red_Priv_Res */
```

Figure 8: Algorithm for Role Addition

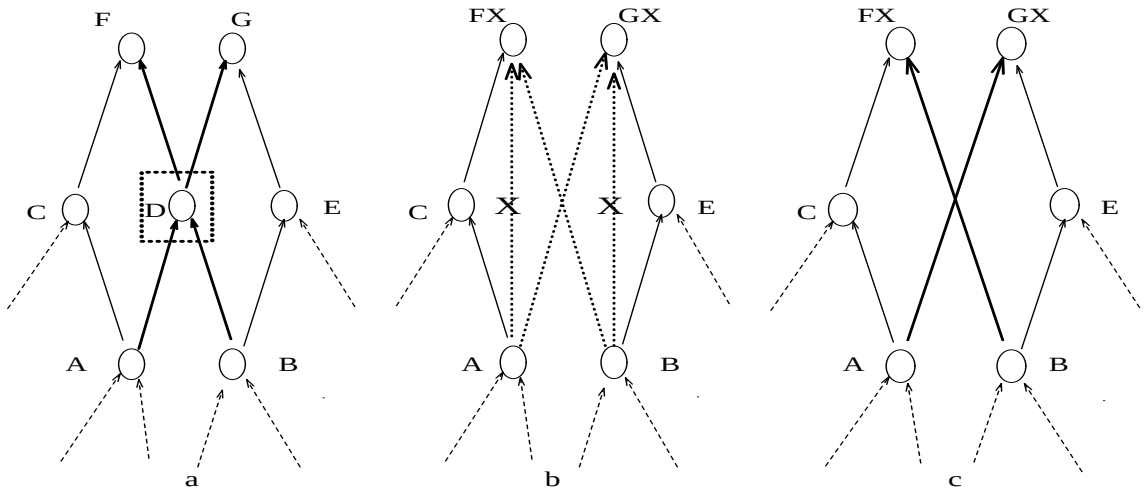

Figure 9: Role Deletion

$D \rightarrow F$, and $B \rightarrow D \rightarrow G$ are replaced by $A \rightarrow FX, A \rightarrow GX, B \rightarrow FX$, and $B \rightarrow GX$, respectively.

The next move is to do away with redundant alternative paths (marked X in the figure 9b) and remove them. We notice that paths $A \rightarrow C \rightarrow FX$ and $B \rightarrow E \rightarrow GX$ contain the set of privileges of paths $A \rightarrow FX$ and $B \rightarrow GX$, respectively. This results in a new role graph structure as shown in figure 9c. □

Both role addition and deletion correspond to real life situations where in creating a new portfolio, a new role is added while in eliminating some "office", a role will be deleted. Role deletion without privilege reduction entails elimination of some "office" in an organization while retaining the total functionality. Privileges of the deleted role would be distributed to other roles.

4.3.2 Role Partition

A role can be partitioned into two or more roles in our role graph. Essentially, the basic partition operations are either vertical or horizontal, and can of course be combined. In both cases it must be specified what the new roles and their corresponding privileges are. Where the order of "seniority" is required, as in the case of vertical partition, it must be specified as well.

In **vertical role partition**, a role is split into two or more roles and an ordering is imposed on them with the *is-junior* relationship. In doing vertical partition, we must specify the target role, the new roles to be created, their direct privileges and their ordering (according to partial privilege criterion). For instance, a role X is not only partitioned into roles $X_1, \cdots, X_n$ but also, these roles must be ordered, e.g. $X_1 \rightarrow \cdots \rightarrow X_n$ (see figure 11b and algorithm 3 of figure 12). Privilege distribution among the new roles is constrained by the privileges associated with the role being partitioned; there *must not* be an increment or decrement of privileges, i.e.

$$Direct(X) = \bigcup_{i=1,\cdots,n} Direct(X_i)$$

Consequently, the privileges associated with the paths in which the role appears neither

Algorithm 2 Role_Deletion(rg, target, inv)

```
/* Deletes a specified role retaining or discarding its privileges depending on inv */
Input: rg = ⟨R, →⟩ (the role graph structure), target (the target role to be deleted),
       inv Boolean indicating whether or not to retain the role's privileges
Output: The role graph structure with target deleted

Var s.set, j.set: role set; r, r_j, r_s: role;
      Begin 1. s.set := Superior_Set(target);                    /* Get the senior set */

            2. j.set := Junior_Set(target);                      /* Get the junior set */
            3. For all r_s ∈ s.set do                            /* Connect Junior and Senior Roles */
                  For all r_j ∈ j.set do add r_j → r_s;
            4. If inv then do
                  For all r_s ∈ s.set do                         /* Transfer Privileges to superiors */
                     Direct(r_s) := Direct(r_s) ∪ Direct(target);
            5. For all r_s ∈ s.set do                            /* Remove all redundant arcs */
                  For all r_j ∈ j.set do
                     If ∃(r_j → r_k ··· → r_s) then delete r_j → r_s;
            6. R := R − target;                                  /* Take out target from system roles
      end.                                                       /* Role_Deletion */

      Function Superior_Set(var rg: role graph; target: role): role_set;
      Var Tempset: role_set; r: role;
      Begin 1. Tempset := ∅;
            2. For all r with target → r do
                  Tempset := Tempset ∪ r;
            3. Superior_Set := Tempset
      end;                                                       /* Superior_Set */

      Function Junior_Set(var rg: role graph; target: role): role_set;
      Var Tempset: role_set; r: role;
      Begin 1. Tempset := ∅;
            2. For all r with r → target do
                  Tempset := Tempset ∪ r;
            3. Junior_Set := Tempset
      end;                                                       /* Junior_Set */
```

Figure 10: Algorithm for Role Deletion

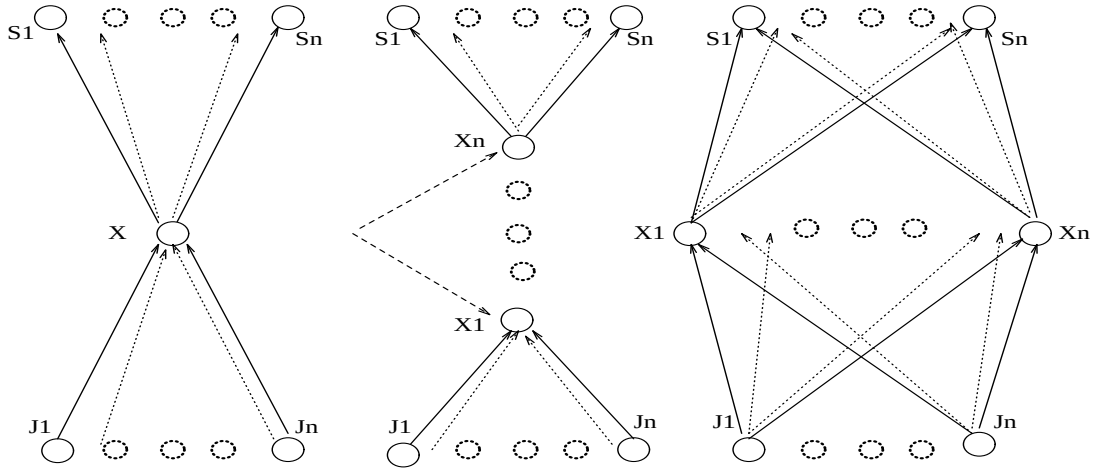

Figure 11: Vertical & Horizontal Role Partition

decrease nor increase. In general, vertical partition leaves the privilege set associated all paths unaffected; only the path length increases.

Further constraints include the requirement for distinct direct privilege sets for the newly created roles, i.e. for any

$$X_i, X_j \in \{X_1, \cdots, X_n\}, Direct(X_i) \bigcap Direct(X_j) = \emptyset$$

Suppose we have a target role for partition (call it X) with a relationship $\{J_1, \cdots, J_n\} \rightarrow X \rightarrow \{S_1, \cdots, S_n\}$ which is partitioned vertically into roles $\{X_1, \cdots, X_n\}$ such that $\{J_1, \cdots, J_n\} \rightarrow \{X_1 \rightarrow \cdots \rightarrow X_n\} \rightarrow \{S_1, \cdots, S_n\}$. It follows that $(X_n \subseteq (S_1 \odot S_2 \odot \cdots \odot S_n)) \wedge (X_1 \subseteq (J_1 \oplus J_2 \oplus \cdots \oplus J_n))$.

Horizontal role partition, on the other hand, involves partitioning a role into two or more roles with none of them being subservient (superior) to another (see figure 11c and algorithm 4 of figure 13). Partition, as used here, merely distributes the direct privileges of the target role among newly created roles that replace it. In partitioning a role, there should be no effective increment or decrement of privileges. In other words, as with vertical partitioning, if role X is partitioned into roles $X_1, \cdots, X_n$, we require that

$$Direct(X) = \bigcup_{i=1,\cdots,n} Direct(X_i)$$

The direct privilege sets of these newly created roles can have empty or non-empty intersections. However, none of them should have identical privilege sets. Note that, unlike vertical partition, horizontal partition can cause a variation of privileges associated with a path when the target role is the senior-most role in the path.

Suppose we have a target role for partition (call it X) with a relationship $\{J_1, \cdots, J_n\} \rightarrow X \rightarrow \{S_1, \cdots, S_n\}$ which is partitioned horizontally into roles $\{X_1, \cdots, X_n\}$ such that $\{J_1, \cdots, J_n\} \rightarrow \{X_1, \cdots, X_n\} \rightarrow \{S_1, \cdots, S_n\}$. It follows that $(\{J_1, \cdots, J_n\} \subseteq (X_1 \odot X_2 \odot \cdots \odot X_n)) \wedge (\{S_1, \cdots, S_n\} \subseteq (X_1 \oplus X_2 \oplus \cdots \oplus X_n))$

Updates to the role graph include the reduction and addition of role privileges which require the specification of the target role and privileges to be removed/added, but do not alter the basic structure and relationships in the role graph structure. These may be addressed within the context of role-privilege authorization.

Algorithm 3 Vertical_Partition(rg, target, $\{((x_i, \rightarrow), x_i.rpset_i)\}$)

```
/* Partitions a given role vertically */
Input: rg = ⟨R, →⟩ (the role organization structure), target (the target role to be partitioned),
       {((x_i, →), rpset_i)} (the new role-direct privilege set pairs and their ordering).
Output: The role graph with target vertically partitioned into {((x_i, →), rpset_i)} and integrated
       into the role graph structure.
Uses Superior_Set and Junior_Set of algorithm 2 in figure 10.

Var s.set, j.set: role set; r_j, r_s: roles;

      Begin
            If Direct(target) ≠ ⋃ (Direct(x_i))
                  Then abort /* Must keep privilege set invariant */
            Else Begin
                  1. R := R ∪ {x_i};                                  /* Add new roles to system */
                  2. s.set := Superior_Set(target);                   /* Generate superior set */
                  3. j.set := Junior_Set(target);                     /* Generate Junior set */
                  4. Add edges x_1 → x_2, x_2 → x_3, ···, x_{n-1} → x_n; /* Create a Path as specified */
                  5. For all x_i do Direct(x_i) := rpset_i;           /* Assign the appropriate privileges */
                  6. For all r_s ∈ s.set do add x_n → r_s;            /* Join the Senior end */
                  7. For all r_j ∈ j.set do add r_j → x_1;            /* Join the Junior end */
                  8. R := R − target;                                 /* Delete target from system */
            end;
      end.                                                            /* Vertical_Partition */
```

Figure 12: Algorithm for Vertical Partition

Algorithm 4 Horizontal_Partition(rg, target, $\{(x_i, x_i.rpset_i)\}$)

```
/* Partitions a given role horizontally */
Input: rg = ⟨R, →⟩ (the role organization structure), target (the target role to be partitioned),{(x_i, rpset_i)}
       (the new role-direct privilege pairs to replace target).
Output: The role structure with target horizontally partitioned into {(x_i)} and integrated into rg.
Uses Superior_Set and Junior_Set of algorithm 2 in figure 10.

Var s.set, j.set: role set; r, r_j, r_s: roles;
      Begin If Direct(target) ≠ ⋃ (Direct(x_i))
                  Then abort                                          /* Must keep privilege set invariant */
            Else begin
                  1. R := R ∪ {x_i};                                  /* Add new roles to system Roles */
                  2. s.set := Superior_Set(target);                   /* Generate the superior set */
                  3. j.set := Junior_Set(target);                     /* Generate the junior set */
                  4. For all x_i ∈ {x_1, ···, x_n} do                 /* assign the privilege set to the new role */
                        Direct(x_i) := rpset_i;                       /* Assign respective privilege sets */
                  5. R := R − target;                                 /* Delete target */
                  6. For all x_i ∈ {x_1, ···, x_n} do
                     begin For all r_s ∈ s.set do add x_i → r_s;      /* Link New Roles to seniors */
                           For all r_j ∈ s.set do add r_j → x_i;      /* Link New Roles to juniors */
                     end;
                  7. For all r, r_i, r_j ∈ R if Ψ(r_i) = Ψ(r_j) then  /* Remove any duplicate roles */
                     Begin
                           for all r_i → r do add the edge r_j → r
                           for all r → r_i do add the edge r → r_j
                           Delete all edges r_i → r and r → r_i;
                           Remove r_i;
                     end;
            end;
      end.                                                            /* Horizontal_Partition */
```

Figure 13: Algorithm for Horizontal Partition

4.4 The Role Graph & Role Coupling

Considering our role graph model proposed in section 4.2, we term the extent of *linkage* between roles a *coupling* which is related to the extent to which privileges are shared among roles. We can have a variety of cases, e.g. where each role is independent of all others or where some roles are *coupled* and hence dependent on each other.

Definition 17 *Coupling: Coupling exists between two roles r_i and r_j if $\exists r_k$ such that $r_k \in r_i \odot r_j$ and $r_k \neq MinRole$. We call r_k a coupling role between r_i and r_j.* □

Definition 18 *Role Independence: Two roles r_i and r_j are independent if and only if $r_i \odot r_j = \{MinRole\}$, i.e. their only coupling is the role common to all roles in the role graph. In other words their only greatest lower bound is MinRole.* □

Independent roles have no coupling between them.

5 Comparison with Hierarchies, Privilege Graphs & Others

The role graph model presented here can simulate a hierarchical organization. We can convert a role graph into a tree (hierarchy) and vice versa. To obtain a tree from a given role graph, we designate **MaxRole** as the root of the hierarchy and do a recursive *bread-first* or *depth-first* traversal for every node with a relationship with **MaxRole**. A given path terminates when **MinRole** is encountered which forms the leaves of all paths in the resulting tree (hierarchy). This tree contains *all* paths present in the associated role graph. In going from a tree to a role graph, we designate the root of the tree to be **MaxRole**, do a depth-first traversal of the tree and equating nodes whenever equal privileges are encountered. The resulting role graph can then be augmented with **MinRole** if necessary. The advantage with the role graph is its *compactness*, i.e. shared nodes lower in the hierarchy, need not be duplicated. This is a major advantage in that it reduces the extent to which shared privileges are scattered among roles which makes the task of tracking their use easier.

To simulate privilege graphs [Bal90], attach to every role an associated functionality that specifies the associated duty requirements/title/etc. With the role's access control list (racl) acting as the user/group node (figure 2), it is possible to determine the authorized users for any role. An authorized user's access rights are determined by the effective privilege set Ψ(r) of the associated role r to which the user is authorized. Further, remove **MaxRole** and assign its explicit privileges to roles with direct partial privilege relationship with it. As well, remove **MinRole** and assign its privileges to those roles with a direct partial privilege relationship with it. The result is a privilege graph.

Finally, although this model is based on subsets with an acyclic graph, it is different from the Bell and LaPadula Model (BLPM). Moreover, although both are meant for security application, they have different approaches to realizing protection. The BLPM relies on subsets, acyclicity and is static. However, it is based on the classification of information as opposed to the execution of operations as is the case in our model. The BLPM specifies two simple operations of either read or write access depending on object classification and subject clearance. This approach realizes multilevel security. In our model, privileges represent pre-defined executions designed in a manner intended to realize certain desired functionality in a system. These operations are designed from considerations of desired system functionality. Once defined, the operations are distributed among roles in the system in the manner that suits organizational requirements. The executions can be simple reads and writes. They can

be a combination of simple reads and writes. But they can also be complex executions such methods in object-oriented programming. These operations need not merely alter or return the information relating to a given object but can also create other objects and invoke other operations.

In the BLPM, once classification has been done, access to information is governed by the simple security property and the *-property. Its specification is static. In our model, execution of privileges can cause the assignment or revocation of privileges pertaining to some role. In that respect, our model is dynamic.

6 Summary & Conclusions

It is important to have a means for role organization that reduces the complexity of privilege management in a role-based security system. This paper has presented a model for role organization derived from three basic role relationships, viz: partial, shared and augmented privileges. These lead to a role graph formulation and use of role graph theory. The model allows for the assignment of privileges in a particular role and through role relationships, we determine the extent of privilege sharing. Given the *acyclicity property*, the role graph model facilitates role partial ordering and privilege subsetting among roles. With an appropriate assignment of privileges to roles and specification of role relationships, the role graph can ease the task of access rights rights administration in a system. Our model has the expressive power of both hierarchies [TDH92] and privilege graphs [Bal90].

The issue of role administration was addressed and algorithms for role management presented. These include algorithms for role addition, deletion and split (partition). Central to role management is the concept of the change (or lack of change) of path privileges, because path privilege changes have implications for roles with indirect access to these privileges.

The concept of paths in the role graph is important in that specific types of processing can be associated with specific paths. Since there is privilege sharing among roles within a path, one can impose constraints about the order of role participation in the processing as well as separation of duty requirements. Role and path independence are important for cases with conflict of interest. Two types of processing that conflict can be associated with independent paths and by ensuring that no user is authorized for roles from both paths, we can impose conflict of interest restriction to processing.

Currently, we are involved in the implementation of a role management tool which we hope will give further insight into the applications of the role graph model in access rights administration.

Acknowledgements

This work was supported in part by a grant from the Natural Science & Engineering Research Council, NSERC, of Canada. We also thank Jim Mullin for his useful comments on an earlier draft of this paper. The anonymous referees raised a number of issues that have been useful in making clear some of our ideas. Sheila Lindsay, who worked on an implementation of the role graph, pointed out some errors in the algorithms proposed earlier. We are grateful for her comments.

References

[AGU72] A. V. Aho, M. R. Garey, and J. D. Ullman. The Transitive Reduction of a Directed Graph. *SIAM Journal of Computing*, 1(2):131–137, June 1972.

[Bal90] R. W. Baldwin. Naming & Grouping Privileges to Simplify Security Management in Large Databases. In *Proc. 1990 IEEE Symposium on Research in Security and Privacy*, pages 116–132. IEEE Computer Society Press, May 1990.

[BL75] D. E. Bell and L. J. LaPadula. Secure Computer Systems: Unified Exposition & Multics Interpretation. Technical Report MTIS AD-A023588, MITRE Corporation, July 1975.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Presss, 1990.

[CW87] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Security Policies. In *Proc. 1987 IEEE Symposium on Research in Security and Privacy*, pages 184–194. IEEE Computer Society Press, April 1987.

[Den76] D. E. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM*, 19(5):236–243, May 1976.

[DM89] J. E. Dobson and J. A. McDermid. Security Models and Enterprise Models. In C. E. Landwehr, editor, *Database Security II: Status & Prospects*, pages 1–39. North-Holland, 1989.

[GMP92] J. Glasgow, G. MacEwen, and P. Panangaden. A Logic for Reasoning About Security. *ACM Transactions on Computer Systems*, 10(3):226–264, August 1992.

[KM92] E. V. Krishnamurthy and A. McGuffin. On the Design & Administration of Secure Database Transactions. *ACM SIGSAC Review*, pages 63–70, Spring/Summer 1992.

[Law93] L. G. Lawrence. The Role of Roles. *Computers & Security*, 12(1):15–21, Feb 1993.

[Man89] Udi Manber. *Introduction to Algorithms: A Creative Approach*. Addison-Wesley, 1989.

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

[NO93b] M. Nyanchama and S. L. Osborn. Role-Based Security: Pros, Cons & Some Research Directions. *ACM SIGSAC Review*, 2(2):11–17, June 1993.

[NO94] M. Nyanchama and S. L. Osborn. Information Flow Analysis in Role-Based Security Systems. *"All about nothing", Journal of Computing & Information*, 1(1), May 1994. Special Issue: Proc. of the 6th International Conference on Computing and Information (ICCI), Peterborough, Ontario, Canada.

[RBWK91] F. Rabitti, E. Bertino, D. Woelk, and W. Kim. A Model of Authorization for Next Generation Databases Systems. *ACM TODS*, 16(1):88–131, March 1991.

[RWK88] F. Rabitti, D. Woelk, and W. Kim. A Model of Authorization for Object Oriented and Semantic Databases. In *Proc. of Int'l Conference on Extending Database Technology*, March 1988.

[TDH92] T. C. Ting, S. A. Demurjian, and M. Y. Hu. Requirements Capabilities and Functionalities of User-Role Based Security for an Object-Oriented Design Model. In C. E. Landwehr and S. Jajodia, editors, *Database Security V: Status & Prospects*, pages 275–296. North-Holland, 1992.

[Tho91] D. J. Thomsen. Role-Based Application Design and Enforcement. In S. Jajodia and C. E. Landwehr, editors, *Database Security, IV: Status and Prospects*, pages 151–168. North-Holland, 1991.