Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems

D. Richard Kuhn
National Institute of Standards and Technology
Gaithersburg, Maryland 20899

Abstract

Role based access control (RBAC) is attracting increasing attention as a security mechanism for both commercial and many military systems. Much of RBAC is fundamentally different from multi-level security (MLS) systems, and the properties of RBAC systems have not been explored formally to the extent that MLS system properties have. This paper explores some aspects of mutual exclusion of roles as a means of implementing separation of duty policies, including a safety property for separation of duty; relationships between different types of exclusion rules; properties of mutual exclusion for roles; constraints on the role hierarchy introduced by mutual exclusion rules; and necessary and sufficient conditions for the safety property to hold. Results have implications for implementing separation of duty controls through mutual exclusion of roles, and for comparing mutual exclusion with other means of implementing separation of duty policies.

1 Introduction

Role based access control (RBAC) is an alternative to traditional discretionary (DAC) and mandatory access control (MAC) policies that is attracting increasing attention [1], particularly for commercial applications. The principle motivation behind RBAC is the desire to specify and enforce enterprise-specific security policies in a way that maps naturally to an organization's structure. With RBAC, security is managed at a level that corresponds closely to the organization's structure. Each user is assigned one or more *roles*, and each *role* is assigned one or more *operations* that are permitted to users in that role. The RBAC emphasis on controlling who has access to operations is fundamentally different from information flow security in multi-level secure systems.

Role based security has been used in a variety of forms for computer system security for at least 20 years, and several proposals for incorporating roles into existing access control mechanisms have been published [2], [3], [4]. More recently, formal definitions for general-purpose RBAC notions have been proposed [5], [6], [7]. While the properties of traditional lattice-based security have been examined extensively in the literature, relatively little has been done with RBAC beyond the formal description of a variety of models. As RBAC becomes supported on increasing numbers of systems, it will be necessary to understand the implications of security mechanisms associated with RBAC.

In particular, separation of duty is an important requirement in many commercial systems, and one of the most desired features of an RBAC system [1]. One means of implementing separation of duty policies is with mutual exclusion of roles [6], [7], [8]. This paper explores some of the properties of mutual exclusion of roles in RBAC systems. (Other means of implementing separation of duty, such as transaction sequencing [9], [10], are not considered here.) The results presented in the paper are useful in comparing mutual exclusion of roles with other potential mechanisms for implementing separation of duty controls, and for understanding the implementation implications of various types of mutual exclusion.

2 Formal Description

This section summarizes the basic rules of RBAC. A number of different flavors of RBAC have been described by various authors. Although it is not identical to any of them, the abstract model defined in this paper is intended to capture the essential features of RBAC as described in models such as those presented in [5], [6], and [7]. (Because of constraints in the treatment of the active role set, the definition of role hierarchy in this paper is in fact more strict than that in [7].) RBAC is a mechanism that can implement a variety of policies, but separation of duty policies are often closely tied to RBAC models, because separation of duty is critical in many commercial applications, and because RBAC is a natural mechanism for implementing separation of duty. The core RBAC mechanisms are summarized first, followed by a discussion of various aspects of mutual exclusion rules.

2.1 Basic Model

Variables used are shown with their types below:

$s : subject$
$i, j, k : role$

p, q : *privilege*
u : *user*

The following definitions are used:

Subjects:
$U : subject \rightarrow user$
$U[s]$ = user u associated with subject s

$R : subject \rightarrow 2^{role}$
$R[s]$ = the set of roles for which subject s is authorized

$A : subject \rightarrow 2^{role}$
$A[s]$ = the current set of active roles for subject s

Roles:
$M : role \rightarrow 2^{user}$
$M[i]$ = the users authorized for role i

$P : role \rightarrow 2^{privilege}$
$P[i]$ = the privileges that are authorized for role i

$E : role \times role$ = the set of role pairs (i, j) that are mutually exclusive with each other

Access to privileges:
$X : subject \times privilege \rightarrow boolean$
$X[s, p]$ = true if and only if subject s can execute privilege p

The following invariants must be maintained by the RBAC system.

Consistent subject: relates human users to subjects executing on the users' behalf. For any subject s associated with user u, a role i is included in the authorized role set $R[s]$ if and only if the user is authorized for role i.

$$(\forall s)(\forall u)(\forall i)|U[s] = u : u \in M[i] \Leftrightarrow i \in R[s] \qquad (1)$$

Role assignment: a subject can execute a privilege only if the subject has selected or been assigned an active role:

$$(\forall s)(\forall p) : X[s, p] \Rightarrow A[s] \neq \emptyset \qquad (2)$$

Role authorization: a subject's active role must be in the set of authorized roles for the subject:

$$(\forall s) : i \in A[s] \Rightarrow i \in R[s] \qquad (3)$$

Privilege authorization: a subject can execute a privilege only if the privilege is authorized for a role in which the subject is currently active:

$$(\forall s)(\forall p)(\exists i) : X[s, p] \Rightarrow i \in A[s] \wedge p \in P[i] \qquad (4)$$

With (2) and (3), this rule guarantees that a subject can execute a privilege only if the privilege is authorized for that active role.

Role Hierarchy: Roles are organized into a partially ordered set (poset) so that if a role is included in the authorized or active role sets, roles below it in the poset are included also:

$$\begin{aligned}&(\forall i, j)(\forall s) : \\ &\quad (i \in A[s] \wedge i \succeq j \Rightarrow j \in A[s]) \\ &\quad \wedge (i \in R[s] \wedge i \succeq j \Rightarrow j \in R[s]) \qquad (5)\end{aligned}$$

2.2 Separation of Duty through Role Exclusion

When implemented using role exclusion rules, separation of duty can be analyzed along at least two dimensions: *when mutual exclusion is applied*, and the *operations to which it is applied.* Two types of mutual exclusion are considered, *authorization-time exclusion* and *run-time exclusion*, that depend on whether the mutual exclusion rule is applied at role authorization time, or at run time, during a user session. These forms of exclusion have been termed *static* and *dynamic* in [6], and *association conflict* and *activation conflict* in [8]. Two additional attributes – *complete exclusion* and *partial exclusion* – indicate whether mutually exclusive roles share no privilege or share some, but not all, privileges.

Safety Condition The purpose of separation of duty rules is to prevent one person from doing all parts of a task that should require two or more, in order to prevent collusion or fraud. For example, many organizations require that the request and approval of a major expenditure be done by two separate people. If there are only two privileges to such a task, then each privilege can be assigned to separate roles and the roles made mutually exclusive. If more than two privileges are involved, then they can be split among two or more roles.

We define a *safety condition* that must be met to ensure that separation of duty requirements are not violated. Let $C[t] : task \rightarrow 2^{privilege}$ be a mapping from tasks requiring separation of duty to sets of privileges required for those tasks (as in the previous example). Then to ensure that no one person can accomplish all parts of a task t, no user can have access to all privileges in $C[t]$. The safety condition for separation of duty is thus as below.

$$(\forall u)(\forall t)(\forall i) : (C[t] \not\subseteq \bigcup_{i \in role | u \in M[i]} P[i]) \tag{6}$$

The term "mutual exclusion" has an intuitive meaning, but some complications can arise when exploring the implications of role exclusion in an RBAC system. Does the mutual exclusion of roles occur when roles are authorized for users, or only for a given user session? Further complications arise if the privileges are made available to other roles that may not be designated as mutually exclusive. Care must be taken that some combination of roles does not allow a user to have access to privileges that should be mutually exclusive. For example, suppose there are two roles, P and Q, which are mutually exclusive, and role Q has access to privileges b and c. Assume that role R has privilege b, role S has privilege c. Then a user in role P could gain access to the same capability provided by role Q through role R and S. Therefore any discussion of role exclusion must consider whether all, or only some, privileges in a role are denied to a mutually exclusive role. Conceivably, the privileges in a role that has been designated as mutually exclusive with another role could be made available to other roles not in the mutually exclusive pair. So another consideration is whether privileges can be shared by roles outside the pair of mutually exclusive roles. To simplify analysis, privileges for which separation of duty is required could be assgned to unique roles, then these roles can be inherited by others. This section discusses some of the alternatives for privilege sharing.

Authorization-time/Run-time Exclusion We define authorization-time exclusion to mean that roles which have been specified as mutually exclusive cannot both be included in a user's set of authorized roles. With run-time exclusion, users may be authorized for two roles that are mutually exclusive, but cannot have both roles active at the same time in a session. In other words, authorization-time exclusion enforces the mutual exclusion rule at the time an administrator sets up role authorizations, while run-time exclusion enforces the rule at the time a user selects roles for a session. A system may be configured to enforce authorization-time or run-time exclusion.

Authorization-time exclusion:

$$\begin{aligned} &(\forall u)(\forall i, j) | i \neq j : \\ &\quad (i, j) \in E \Rightarrow u \in M[i] \Rightarrow u \notin M[j] \end{aligned} \tag{7}$$

Authorization-time exclusion says that if two roles are mutually exclusive, then a user can be authorized for one role only if user is not authorized for the other role. A similar but weaker form of exclusion is to allow users to be authorized for roles that are mutually exclusive, but allow them to be active in only one role at a time. This will be referred to as run-time exclusion.

Run-time exclusion :

$$\begin{aligned} &(\forall u)(\forall s)(\forall i, j) | i \neq j; U[s] = u : \\ &\quad (i, j) \in E \Rightarrow u \in M[i] \wedge u \in M[j] \\ &\quad \Rightarrow i \in A[s] \Rightarrow j \notin A[s] \end{aligned} \tag{8}$$

In addition to the time at which mutual exclusion is applied, the degree to which privileges are shared by mutually exclusive roles and by other roles must also be considered. This dimension is independent of authorization-time or run-time exclusion. Four possibilities can be considered:

- Disjoint/disjoint (D/D): Privilege sets for mutually exclusive roles are disjoint. That is, if two roles are designated as mutually exclusive, then each privilege is assigned to only one of them. In addition, if a privilege is assigned to a role that has been designated as mutually exclusive with another role, then it is not assigned to any other role.

  $(\forall i, j, k)(\forall p) | i \neq j, i \neq k :$
  $(i, j) \in E \Rightarrow p \in P[i] \Rightarrow p \notin P[j] \wedge p \notin P[k]$

- Disjoint/shared (D/S): Privilege sets for mutually exclusive roles are disjoint, but if a privilege is assigned to a role that has been designated as mutually exclusive with another role, then it may also be assigned to roles outside of the mutual exclusion relationship.

  $(\forall i, j)(\forall p) | i \neq j : (i, j) \in E \Rightarrow p \in P[i] \Rightarrow p \notin P[j]$

- Shared/disjoint (S/D): Privileges may be shared between roles that are mutually exclusive, with the provision that each must have at least one privilege not available to the other. In addition, if a privilege is assigned to a role that has been designated as mutually exclusive with another role, then it is not assigned to any other role.

  $(\forall i, j, k)(\exists p)(\forall q) | i \neq j, i \neq k, j \neq k :$
  $(i, j) \in E \Rightarrow (p \in P[i] \Rightarrow p \notin P[j])$
  $\wedge (q \in P[i] \vee q \in P[j] \Rightarrow q \notin P[k])$

- Shared/shared (S/S): Privileges may be shared between roles that are mutually exclusive, with the provision that each must have at least one privilege not available to the other. If a privilege is assigned to a role that has been designated as mutually exclusive with another role, then it may also

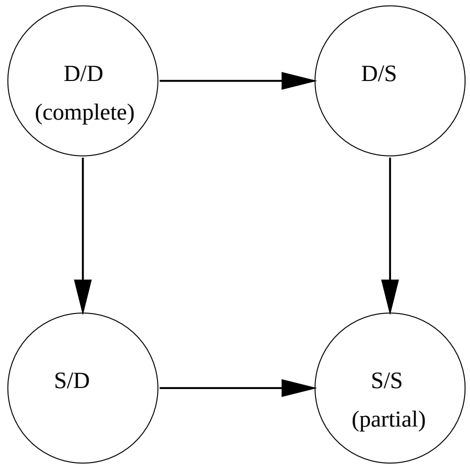

Figure 1: Mutual Exclusion Rule Relationships

be assigned to roles outside of the mutual exclusion relationship.

$$(\forall i, j)(\exists p) | i \neq j : \\ (i, j) \in E \Rightarrow (p \in P[i] \Rightarrow p \notin P[j])$$

From the definitions above, it can be seen that the mutual exclusion rules have the relationships shown in Figure 1. (Arrows in Figure 1 indicate logical implication.) Two of the above rule combinations are of particular interest: disjoint/disjoint and shared/shared. These will be referred to as complete exclusion and partial exclusion respectively.

Complete/Partial Exclusion

The set of privileges accessible by roles to which mutual exclusion is applied is a significant consideration in ensuring separation of duty. In the earlier example, it may be desirable to prevent access to all privileges available to role Q or only to some. The restriction of all privileges available to a mutually exclusive role will be referred to as *complete* exclusion, and restriction of only some privileges available to a mutually exclusive role as *partial* exclusion. A system may be configured to enforce complete or partial exclusion.

Complete exclusion (disjoint/disjoint):

$$(\forall i, j, k)(\forall p) | i \neq j, i \neq k : \\ (i, j) \in E \Rightarrow p \in P[i] \Rightarrow p \notin P[j] \wedge p \notin P[k] \quad (9)$$

Complete exclusion says that if any role i is mutually exclusive with another role, then no privilege in i is assigned to any other role

Partial exclusion (shared/shared):

$$(\forall i, j)(\exists p) | i \neq j : \\ (i, j) \in E \Rightarrow p \in P[i] \Rightarrow p \notin P[j] \quad (10)$$

Partial exclusion says that if any role i is mutually exclusive with another role, then at least one other privilege in i is not assigned role j.

3 Separation of Duty Properties

The mutual exclusion rules (7) through (10) can be coupled with invariants (1) through (5) to produce a variety of RBAC systems with separation of duty. This section explores some properties of mutual exclusion rules in a role based access control system, including:

- desired properties of mutual exclusion for roles;
- relationships between authorization-time and run-time, and between complete and partial exclusion rules;
- constraints on the role hierarchy introduced by the mutual exclusion rules, including the non-existence of a "root" role that contains all roles;
- necessary and sufficient conditions for the safety condition to hold.

3.1 Basic Properties

The first two results are obvious from definitions. From these results and definitions of rules (7) through (10), it can be seen that combinations – authorization-time complete (A/C), run-time complete (R/C), authorization-time partial (A/P), and run-time partial (R/P) – have the relationships shown in Figure 2.

Theorem 3.1 *If authorization-time exclusion holds, then run-time exclusion is maintained.*

Theorem 3.2 *If complete exclusion holds, then partial exclusion is maintained.*

An essential property of any separation of duty implementation is that roles designated as mutually exclusive cannot be brought into the active set simultaneously. The next result establishes this property for authorization-time and run-time exclusion in an RBAC system.

Theorem 3.3 *Mutually exclusive roles cannot be brought into the active set A.*

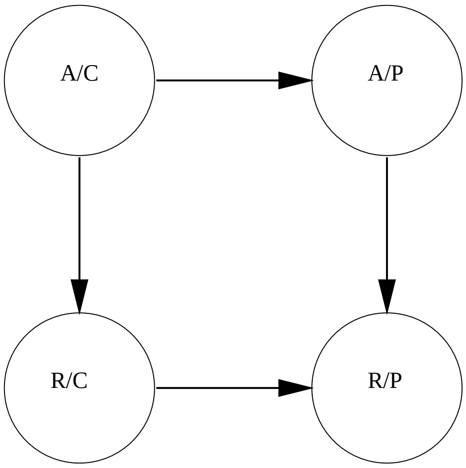

Figure 2: Separation of Duty Relationships

Proof:
Case I - Authorization-time exclusion. Assume that roles i and j are mutually exclusive but are both part of the active set for some subject s of user u (i.e., $U[s] = u$):

$(i, j) \in E \wedge i \in A[s] \wedge j \in A[s]$,

Then by (3), we have

$i \in R[s] \wedge j \in R[s]$,

and by (1),

$u \in M[i] \wedge u \in M[j]$.

So from (6), we have $(i, j) \notin E$, which contradicts the assumption.

Case II - Run-time exclusion. Again, assume

$(i, j) \in E \wedge i \in A[s] \wedge j \in A[s]$,

Then from (7) we have

$u \notin M[i] \vee u \notin M[j]$.

But by (3),

$i \in R[s] \wedge j \in R[s]$,

so by (1)

$u \in M[i] \wedge u \in M[j]$,

which is a contradiction. Q.E.D.

The next result is an interesting consequence of the run-time exclusion rule. The practical significance of this result is that a role cannot inherit another role that has been designated as mutually exclusive with it. This is clearly a desirable property, and this result shows that the rules are sufficient to ensure it.

Theorem 3.4 *Two roles i and j can be mutually exclusive only if they are incomparable within the role hierarchy poset:* $(i, j) \in E \Rightarrow \neg(i \succeq j \vee j \succeq i)$.

Proof: Suppose $(i, j) \in E \wedge (i \succeq j \vee j \succeq i)$. (The proof will be shown for run-time exclusion only; it should be clear that it can be shown for authorization-time exclusion as above.) Arbitrarily choose i as the role which is in the active role set, with $i \succeq j$, i.e.,

$i \in A[s]$.

Then by (5),

$j \in A[s]$,

so by (8),

$(i, j) \notin E$,

which contradicts the assumption. Q.E.D.

An immediate corollary is that if there are any mutually exclusive roles, then a role cannot be mutually exclusive with itself. This might have been required as one of the basic rules, but as it happens, it is a consequence of them.

Corollary 3.5 *A role cannot be mutually exclusive with itself:* $\forall i : (i, i) \notin E$

Proof: By the theorem above,

$i \succeq j \vee j \succeq i \Rightarrow (i, j) \notin E$.

Substituting $j := i$ gives

$i \succeq i \Rightarrow (i, i) \notin E$

By definition, $i \succeq i$, so for all i,

$(i, i) \notin E$. Q.E.D.

3.2 Constraints Introduced by Mutual Exclusion

If there are any mutually exclusive roles, then those roles cannot have a common upper bound.

Theorem 3.6 *If there is any pair $(i, j) \in E$, then there can be no role k such that $k \succeq i \wedge k \succeq j$.*

Proof: Suppose there is some role k, and mutually exclusive roles i, j such that

$k \succeq i \wedge k \succeq j \wedge (i, j) \in E$.

Then because

$k \succeq i \wedge k \succeq j$,

the role hierarchy rule (5) requires that

$i \in A[s] \wedge j \in A[s]$,

so by rule (7),

$(i, j) \notin E$,

which contradicts the assumption. Q.E.D.

An immediate corollary is that the rules also prohibit the existence of a "superuser" or "root" role that contains all other roles on the system.

Corollary 3.7 *For any pair $(i, j) \in E$, then there can be no role r such that for all i, $r \succeq i$.*

The implication of this result is that a system enforcing authorization-time exclusion can have a "root" user only if no roles are designated as mutually exclusive, i.e., if separation of duty is not used. Note that this holds even if run-time exclusion is used. Under run-time exclusion, a single user could be authorized for all roles, but they could not be active simultaneously. But because the "root" role inherits all other roles, "root" could never be activated. If the system is configured for authorization-time exclusion, then the system would prevent any individual user from being authorized for all roles.

However, note that the rules do not prevent two mutually exclusive roles from having a common lower bound. This is important since many systems will have a basic role that all other roles inherit. For example, a hospital system may have an "employee" role that is used to allow other roles to inherit the basic privileges that are available to all employees.

The separation of duty rules also have implications for the cardinality of privilege sets.

Theorem 3.8 *If $(i, j) \in E$ then either $P[i]$ and $P[j]$ are disjoint sets or else $\#P[i] \geq 2$ and $\#P[j] \geq 2$.*

Proof: If complete exclusion is in effect, then $P[i]$ and $P[j]$ are disjoint sets by (9). If partial exclusion is in effect, then either $P[i]$ and $P[j]$ are disjoint sets or else $P[i]$ and $P[j]$ overlap. If they overlap, they have at least one privilege in common. But (10) requires that role i has at least one privilege in $P[i]$ that is not in $P[j]$, and role j has at least one privilege that is not in $P[i]$. Therefore neither $P[i] \subseteq P[j]$ nor $P[j] \subseteq P[i]$. Therefore each must have at least one additional privilege in addition to the one or more privileges that they have in common. Q.E.D.

3.3 Maintenance of Safety Condition

For any reasonable implementation, the separation of duty rules must maintain the safety condition. This section develops the relationships between the safety condition and the various forms of mutual exclusion.

Theorem 3.9 *If there are no empty privilege sets, then authorization-time/complete exclusion is sufficient to ensure the safety condition.*

Proof: Assume that complete exclusion holds but the safety condition does not. Then there must be some user that is authorized for all privileges $C[t]$ that are part of some task t:

$$(\exists u)(\exists t) : (C[t] \subseteq \bigcup_{i \in role | u \in M[i]} P[i])$$

Since $C[t]$ represents a set of privileges that are required for some critical task t, $C[t]$ must be split up among at least two mutually exclusive roles. (see Section 4.4) Because $C[t]$ has been split among 2 or more mutually exclusive roles,

$$C[t] = P[i\prime] \cup P[i\prime\prime] \cup \ldots$$

To simplify the presentation we will consider only the case where $C[t]$ is split between two roles $i\prime$ and $i\prime\prime$, where $(i\prime, i\prime\prime) \in E$.

Let $Ru[u]$ represent the set of all roles authorized for a user u, i.e., $Ru[u] = \bigcup_{i \in role | u \in M[i]} P[i]$. So the above formula can be written as:

$$(\exists u)(\exists t) : (P[i\prime] \cup P[i\prime\prime] \subseteq \bigcup_{i \in Ru[u]} P[i])$$

Then all the mutually exclusive role privileges in $i\prime$ and $i\prime\prime$ must be in $\bigcup_{i \in Ru[u]} P[i]$, which is the set of all privileges available to the user u. So by definition of $\subseteq$,

$$(\exists u)(\exists t) : p \in P[i\prime] \vee p \in P[i\prime\prime] \Rightarrow p \in \bigcup_{i \in Ru[u]} P[i]$$

This is equivalent to:

$$(\exists u)(\exists t) : (p \in P[i\prime] \Rightarrow p \in \bigcup_{i \in Ru[u]} P[i])$$
$$\wedge (p \in P[i\prime\prime] \Rightarrow p \in \bigcup_{i \in Ru[u]} P[i])$$

But complete exclusion (9) ensures that privileges in any role that is designated as mutually exclusive with some other role are contained in only one privilege set, so a privilege can be in $P[i\prime]$ or $P[i\prime\prime]$, but not both. Therefore the only way that privileges in $P[i\prime]$ and $P[i\prime\prime]$ could be available to a user is if both $i\prime$ and $i\prime\prime$ are in the user's authorized role set, but this is impossible by rule (7). Q.E.D.

Authorization-time/complete exclusion thus ensures the safety condition across all RBAC users. It is easy to see also that run-time/complete exclusion will ensure a sort of "run-time safety condition" within a single user session. This form of exclusion could be appropriate for organizations where it is impractical to rigidly divide privileges among employees ([11] and [12] have discussed this problem).

The weakest of the four possible ways of assigning privileges (shared/shared), or partial exclusion, is a necessary (but not sufficient) condition for safety.

Theorem 3.10 *The safety condition can be met for a subject s only if (at least) partial exclusion is maintained.*

Proof: We show this directly by showing that, given subject s and role $i \in R[s]$, formula (6) implies formula (10).

Rewriting (6), we have

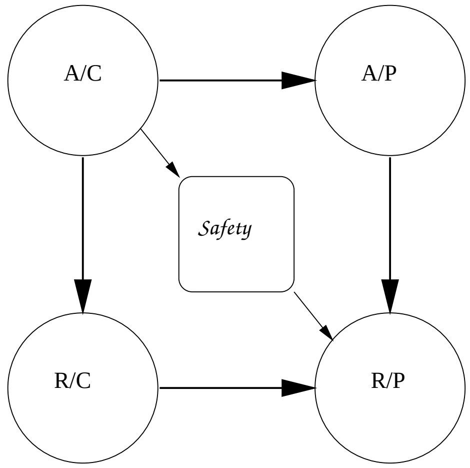

Figure 3: Safety and Mutual Exclusion

$(\forall s)(\forall i, j, k)|i \neq j, k \neq j, k \in R[s]$:
$i \in R[s] \wedge (i, j) \in E$
$\Rightarrow \neg((\forall p) p \in P[j] \Rightarrow p \in \bigcup P[k])$

This is equivalent to
$(\forall s)(\forall i, j, k)|i \neq j, k \neq j, k \in R[s]$:
$i \in R[s] \wedge (i, j) \in E \Rightarrow ((\exists p) p \in P[j] \wedge p \notin \bigcup P[k])$

Because the only constraint on k is that $k \neq j$, there is some k such that $k = i$, so this can be rewritten as
$(\forall s)(\forall i, j, k)|i \neq j, k \neq j, k \in R[s]$:
$i \in R[s] \wedge (i, j) \in E$
$\Rightarrow ((\exists p) p \in P[j] \wedge p \notin P[i] \wedge p \notin \bigcup P[k])$

This immediately implies (10):
$(\forall i, j)(\exists p)|i \neq j : (i, j) \in E \Rightarrow p \notin P[i] \vee p \notin P[j]$
Q.E.D.

The significance of this result is that any system that does not provide at least partial exclusion cannot ensure the safety condition through mutual exclusion. Since partial exclusion is not sufficient for safety, if a system supports only partial exclusion, then other mechanisms must be provided to ensure safety.

4 Conclusions

The results regarding relationships between various types of mutual exclusion rules presented in this paper can be summarized in Figure 3. Authorization-time/complete exclusion is sufficient to ensure safety. As noted earlier however, run-time/complete exclusion will ensure separation of duty safety within a single user session. This provides some flexibility in implementing RBAC for organizations which are too small or otherwise find it impractical to rigidly separate privileges. In this case, audit mechanisms can be used in concert with RBAC mechanisms to ensure (post-hoc) that separation of duty requirements are being followed.

There does not seem to be any obvious problem with implementing task sequencing mechansisms layered on top of a role exclusion mechanism in an RBAC system. Work-flow or other sequencing mechanisms might be added as additional components on a basic system implementing role exclusion.

5 Acknowledgements

I am very grateful to Serban Gavrilla and David Ferraiolo for pointing out some important clarifications.

References

[1] R. Sandhu, E.J. Coyne, and C.E. Youman, editors. *Proceedings of the First ACM Workshop on Role Based Access Control.* ACM, 1996.

[2] R.W. Baldwin. Naming and grouping privileges to simplify security management in large databases. In *Proceedings, IEEE Computer Society Symposium on Research in Security and Privacy.* IEEE Computer Society, 1990.

[3] D.J. Thomsen. Role-based application design and enforcement. In *Database Security IV: Status and Prospects.* North-Holland, 1991.

[4] D.F. Sterne. A TCB subset for integrity and role-based access control. In *15th National Computer Security Conference.* NIST/NSA, 1992.

[5] D. Ferraiolo and D.R. Kuhn. Role based access control. In *15th National Computer Security Conference.* NIST/NSA, 1992.

[6] D. Ferraiolo, J. Cugini, and D.R. Kuhn. Role based access control: Features and motivations. In *Annual Computer Security Applications Conference.* IEEE Computer Society Press, 1995.

[7] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role based access control models. *IEEE Computer*, 29(2), February 1996.

[8] D. Jonscher. Extending access control with duties: realized by active mechanisms. In *Workshop on Database security.* IFIP WG 11.3, 1992.

[9] R. Sandhu. Transaction control expressions and separation of duties. In *Proceedings, IEEE Computer Society Symposium on Research in Security and Privacy.* IEEE Computer Society, 1988.

[10] L. Notargiacomo, B.T. Blaustein, and C.D. McCollum. A model of integrity and dynamic separation fo duty for a trusted DBMS. In *Database Security VII Workshop on Database security*. IFIP WG 11.3, 1992.

[11] D. Clark and D.R. Wilson. Evolution of a model for computer integrity. In *Proceedings, 11th National Computer Security Conference: a postscript.* NIST/NSA, 1988.

[12] M.J. Nash and K.R. Poland. Some conundrums concerning separation of duty. In *Proceedings, 1990 IEEE symposium on computer security and privacy*. IEEE, 1990.