Supporting Authorization Query and Inter-domain Role Mapping in Presence of Hybrid Role Hierarchy

Siqing Du James B. D. Joshi
LERSAIS & Department of Information Science and Telecommunications, University of Pittsburgh, Pittsburgh, USA
{sdu, jjoshi}@mail.sis.pitt.edu

ABSTRACT

The role hierarchy is one of the most distinguished features of an RBAC approach to securing large systems as it facilitates efficient administration of permissions. However, the role hierarchy as defined in the currently standardized RBAC model has limitations in capturing generic policy requirements such as separation of duty, time-based and cardinality constraints. To address such limitations, permission inheritance and activation inheritance semantics have been introduced to define three different types of role hierarchies. In presence of a hybrid hierarchy that allows all the three types of hierarchies to coexist, the overall hierarchy administration problem becomes quite complex. A key problem is to efficiently handle authorization queries to decide whether a user's request to activate a set of roles should be granted. A hybrid hierarchy also makes the problem of mapping a request for a set of permissions to a minimal set of roles difficult. Such a mapping is crucial in multidomain environments where different security domains have to establish and engage in secure interoperation by first mapping their security policies. In this paper, we investigate these two problems and present solutions that are efficient and practical.

Categories and Subject Descriptors

D.4.6 [Security and Protection]: Access controls;
G.4 [Mathematical Software]: Algorithm, design and analysis.

General Terms

Algorithms, Security, Theory.

Keywords

Role based access control, hierarchy, role mapping, multidomain, secure interoperation.

1. INTRODUCTION

Role based access control (RBAC) has become the predominant approach for advanced access control in large, distributed systems [7, 12]. It not only encompasses traditional *discretionary* and *mandatory* access control (DAC/MAC) models, but also offers many attractive features, such as policy neutrality, support for least privilege and efficient access control management [6, 15 11].

One of the most distinguished characteristics of the RBAC approach is the role hierarchy. A role hierarchy defines inheritance semantics related to permission acquisitions and role activations through role-role relationships that can be utilized to efficiently and effectively structure functional roles of an organization having related access control needs. Joshi *et al.* have established a clear distinction among the following three types of role hierarchies in the context of the generalized temporal RBAC (GTRBAC) model [11]: *permission-inheritance*-only hierarchy (*I*-hierarchy, $\geq_i$), *activation-inheritance*-only hierarchy (*A*-hierarchy, $\geq_a$), and the combined *permission-inheritance* and activation hierarchy (*IA*-hierarchy, $\geq$). It has been shown that such a fine-grained hierarchical semantics facilitates capturing a wide range of security requirements, including the specification of fine-grained separation of duty (SoD) and temporal constraints on hierarchically related roles, and user-centric as well as permission-centric cardinality constraints on roles in the hierarchy [12, 17]. Furthermore, hybrid hierarchies where all three types of hierarchical relations can coexist, allow flexible ways to express a given policy [14], and provide better support for policy integration in multidomain environments [19].

Given a role hierarchy, the system must maintain knowledge of the set of roles that a user is allowed to activate in a single session in order to process a user's request to activate a set of roles. In the presence of a hybrid hierarchy, maintaining permission acquisition and role activation semantics can become quite challenging. Joshi *et al.* introduce the concept of *uniquely activable set* (UAS) to facilitate the analysis of hybrid hierarchies and simplify the process of determining the activation and permission-acquisition sets [11]. A UAS is a set each element of which is a set of roles that can be activated in a single session by a specific user. In other words, a user assigned to a role in a hierarchy can activate only those set of roles that occur in a UAS associated with him. Computing UAS has been shown to be a complex task [3]. Chandran *et al.* have presented two approaches for computing the UAS of a hierarchy [3]: the *decomposition* based (DB) approach that constructs the UAS by computing UASs of the sub-hierarchies, and the *derived relations* based (DRB) approach that uses a set of implication rules, introduced in [13], to derive hierarchical relations between every pairs of roles in the hierarchy and then compute the UAS from them [3]. While these approaches are useful for carrying out exhaustive analysis of policies, they have non-polynomial time complexities.

The presence of hybrid hierarchies in RBAC-based policies also makes the issue of secure interoperation between two domains very difficult. In particular, one of the key requirements when two systems interoperate is to make sure that their policies are appropriately mapped, including mapping of role hierarchies, so that the principles of *security* and *autonomy* are preserved [12, 19]. In a loosely coupled environment, when two domains that are not known to each other decide to interact, we assume the process starts with the specification of service requirements of each domain. Hence, the underlying issue related to secure interoperation is that of appropriately mapping policies to satisfy each request of a domain by the other domain, where each request is represented as a set of permissions required. These requested set of permissions in turn need to be satisfied by identifying the set of local roles that may be hierarchically related. Ideally, we want to identify the minimal number of roles that are associated with the requested set of permissions. This approach to inter-domain mapping has been introduced in [16].

In this paper, we focus on these two crucial problems centered on the use of hybrid hierarchies. For the first problem, referred to as the *authorization query* (AQ) problem, we present an approach simpler than the DB or the DRB approaches to determine whether a user's request to activate a set of roles can be granted. We define a special matrix called *assistant matrix* (AM) that encodes some relationships among the hierarchically related roles. For the second problem, referred to as the *inter-domain role mapping* (IDRM) problem, we show that finding the minimal set of roles that satisfy an external domain's request, represented as a set of permissions, is *NP-complete*. We, therefore, present a heuristic greedy search approach to find a sub-optimal solution and enhance it by including a probability parameter to get a better result. The novelties of our contributions are as follows:

1. To the best of our knowledge, work related to UAS and hybrid hierarchy, and hence the AQ problem is new and has not been carried out by other researchers. The proposed solution is efficient and can also be extended to address policy analysis. Note that the AM does not completely encode all the UAS elements or the derived relations as is done by the DB and DRB approaches, respectively. However, this approach can be used to identify undesirable UAS elements (thus indicating undesirable hierarchy design) by querying whether the undesirable set of roles are allowed by the hybrid hierarchy to be activated in a single session. As the DB and DRB approaches have non-polynomial time complexities, this approach will support for the practical use of the hybrid hierarchy.
2. We show that, during inter-domain mapping in presence of hybrid hierarchy, finding the minimal set of roles that match a set of permissions is *NP-complete*. We extend an existing heuristic greedy approach to get better sub-optimal solution to this problem. The extended algorithm has the flavor of both the simulated annealing and genetic algorithms. To the best of our knowledge, the IDRM problem in this form has not been addressed by other researchers, let alone the problem in presence of the hybrid hierarchy.

The rest of the paper is organized as follows. In Section 2, we provide an overview of the concept of hybrid hierarchy and relevant background material on the GTRBAC. In Section 3, we discuss the AQ problem and present our proposed solution. In Section 4, we investigate the IDRM problem and show that it is *NP-complete*. We then present an enhanced greedy search algorithm to find a sub-optimal solution for the IDRM problem. We present brief overview of implementation in Section 5, the related work in Section 6 and conclusions and future work in Section 7.

2. GTRBAC and HYBRID HIERARCHY

The GTRBAC model introduces the separate notion of role enabling and role activation, and provides constraints and event expressions associated with both [12]. In the GTRBAC model, an *enabled* role indicates that a valid user can activate it, whereas an *activated* role indicates that at the least one user has activated the role. The basic GTRBAC model proposed in [12], allows specification of the following set of constraints:

1. *temporal constraints on role enabling/disabling* that allow specification of intervals and durations in which a role is enabled; (*ii*)
2. *temporal constraints on user-role and role-permission assignments* that allow specifying intervals and durations in which a user or permission is assigned to a role; (*iii*)
3. *activation constraints*: These constraints allow specification of restrictions on the activation of a role. These include, for example, specifying the total duration for which a user may activate a role, or the number of concurrent activations of the role at a particular time;
4. *run-time events* that allow an administrator and users to dynamically initiate the various role events, or enable the duration or activation constraints;
5. *constraint enabling* that events that enable or disable duration and role activation constraints mentioned earlier; and
6. *triggers* that allow expressing dependencies among events and conditions

Semantically, a role hierarchy expands the scope of the permission-acquisition and role-activation semantics beyond the explicit user-role and role-permission assignments to inheritance through the hierarchical relations defined among roles. Within the GTRBAC framework, the following three hierarchy types have been identified: *permission-inheritance-only* hierarchy (*I*-hierarchy), *role-activation-only* hierarchy (*A*-hierarchy) and the combined *inheritance-activation* hierarchy (*IA*-hierarchy) [12]. Table 1 captures the predicate notations used in defining the semantics of these hierarchies [12]. Predicates *enabled*(*r, t*), *assigned*(*u, r, t*) and *assigned*(*p, r, t*) refer to the status of roles, user-role and role-permission assignments at time *t*. Predicate *can_activate*(*u, r, t*) indicates that user *u* can activate role *r* at time *t*, implying that user *u* is implicitly or explicitly assigned to role *r* at time *t*. *active*(*u, r, s, t*) indicates that role *r* is active in user *u*'s session *s* at time *t*, whereas, *acquires*(*u, p, s, t*) implies that *u* acquires permission *p* at time *t* in session *s*. The axioms in Table 1 capture the key relationships among these predicates and identify precisely the permission-acquisitions and role-activations allowed in GTRBAC [12]. Axiom (1) states that if a permission is assigned to a role, then it *can be acquired* through that role. Axiom (2) states that all users assigned to a role *can activate* that role. Axiom (3) states that if a user *u can activate* a role *r*, then all the permissions that *can be acquired* through *r can be acquired*

Table 1. Status predicates

Predicate	Meaning	Axioms
enabled(*r*, *t*)	*Role r is enabled at time t*	*For all* $r \in$ *Roles*, $u \in$ *Users*, $p \in$ *Permissions*, $s \in$ *Sessions*, *and time instant* $t \geq 0$, *the following implications hold*:
u_assigned(*u*, *r*, *t*)	*User u is assigned to role r at time t*	
p_assigned(*p*, *r*, *t*)	*Permission p is assigned to role r at time t*	1. *p_assigned*(*p*, *r*, *t*)$\rightarrow$ *can_be_acquired*(*p*, *r*, *t*)
can_activate (*u*, *r*, *t*)	*User u can activate role r at time t*	2. *u_assigned*(*u*, *r*, *t*) $\rightarrow$ *can_activate* (*u*, *r*, *t*)
can_acquire (*u*, *p*, *t*)	*User u can acquire permission p at time t*	3. *can_activate* (*u*, *r*, *t*) $\wedge$ *can_be_acquired*(*p*, *r*, *t*) $\rightarrow$ *can_acquire* (*u*, *p*,*t*)
can_be_acquired(*p*, *r*, *t*)	*Permission p can be acquired through role r at time t*	
active(*u*, *r*, *s*, *t*)	*Role r is active in user u's session s at time t*	4. *active*(*u*, *r*, *s*, *t*) $\wedge$ *can_be_acquired*(*p*, *r*, *t*) $\rightarrow$ *acquires*(*u*, *p*, *s*, *t*)
acquires(*u*, *p*, *s*, *t*)	*User u' acquires permission p in session s at time t*	

Table 2. Role hierarchies in GTRBAC

Category	Short form	Notation	The following implication rule holds
Unrestricted hierarchies *(No effect of timing constraints on role)*	*I-hierarchy*	$(x \geq_i y)$	$\forall p$, $(x \geq_i y) \wedge$ *can_be_acquired*(*p*, *y*, *t*)$\rightarrow$ *can_be_acquired*(*p*, *x*, *t*)
	A-hierarchy	$(x \geq_a y)$	$\forall u$, $(x \geq_a y) \wedge$ *can_activate* (*u*, *x*, *t*) $\rightarrow$ *can_activate* (*u*, *y*, *t*)
	IA-hierarchy	$(x \geq y)$	$(x \geq y) \leftrightarrow (x \geq_i y) \wedge (x \geq_a y)$
Consistency Property: *Let* $<f_1><f_2> \in \{\geq_i, \geq_a, \geq\}$. *Let x and y be distinct roles such that* $(x<f_1>y)$; *then the condition* $\neg(y <f_2> x)$ *must hold.*			

by *u*. Similarly, axiom (4) states that if there is a user session in which a user *u* has activated a role *r* then *u acquires* all the permissions that *can be acquired* through role *r*. We note that axioms (1) and (2) indicate that permission-acquisition and role-activation semantics are governed by explicit user-role and role-permission assignments. We ignore the time parameter for the rest of the paper.

In Table 2, the semantics of each hierarchy type is defined by its corresponding implication rule in the last column. The rule for the *I*-hierarchy, $(x \geq_i y)$, implies that if $(x \geq_i y)$ holds, then the permissions that can be acquired through role *x* include all the permissions that *can be acquired through* role *y*. In other words, permissions of the junior roles are inherited by the senior role. Similarly, the rule for the *A*-hierarchy implies that if user *u can activate* role *x*, and $x \geq_a y$ is defined, then user *u can* also *activate* role *y* even if he is not explicitly assigned to *y*. Note that it does not imply that user *u* can acquire *y*'s permissions by merely activating *x*. In other words, permission-inheritance is not implied in an *A*-hierarchy. The *IA*-hierarchy is the most general form and includes both permission-inheritance and role-activation semantics. In the remaining sections we do not use the time parameter *t* in any expression.

We represent a hybrid hierarchy as $H = (R, F)$, where *R* is a set of roles, $R = \{r_1, r_2, \ldots, r_n\}$; *F* is a set of hierarchy relations defined on *R*, $F \subseteq \{\geq_i, \geq_a, \geq\}$. If $F = \{<f>\}$ is a singleton set with hierarchy relation $<f>$, then we call *H a monotype* hierarchy and write $(R, <f>)$. We use notation $P_{au}(r)$ to refer to the set of *authorized permissions* of role *r*, which is a set of permissions that *can be acquired* by activating *r. That is,*

- $P_{au}(r)$ = *can_be_acquired*(*r*, *p*), and
- $P_{au}(R) = \bigcup_{r \in R} P_{au}(r)$.

Formally, the UAS can be defined as follows [13].

Definition 1: *Let* $H = (R, F)$ *be a rooted hybrid hierarchy. Then,* $UAS(H) = \{Y_1, Y_2, \ldots, Y_m\}$, *where* $\varnothing \subset Y_i \subseteq R$ *for each* $i \in \{1, 2, \ldots, m\}$, *is the Uniquely Activable Set* (***UAS***) *of role sets for a user assigned to the senior-most role* S_H *of H if the following conditions hold:*

- $\forall i, j \in \{1, 2, \ldots, m\}$ *and* $i \neq j$, $P(Y_i) \neq P(Y_j)$, *and*
- $\forall Z \subseteq R$ s.t. $Z \notin UAS(H)$, *if* $P(Y) = P(Z)$ *for a* $Y \in UAS(H)$, *then* $(|Y| < |Z|)$; *where* $|A|$ *is the cardinality of set A.*

Example 1: An example hybrid hierarchy is illustrated in Figure 1. *I*, *A* and *IA*-hierarchies are represented by a simple line, a dotted line and a line with arrows on both ends, respectively. For this hierarchy, the UAS for a user assigned to r_1 is:

$\{ \{r_1\}, \{r_2\}, \{r_3\}, \{r_4\}, \{r_5\}, \{r_6\}, \{r_7\} \{r_1, r_3\}, \{r_1,r_5\}, \{r_1,r_6\}, \{r_1,r_7\}, \{r_2, r_3\}, \{r_2,r_5\}, \{r_2,r_6\}, \{r_2,r_7\}, \{r_3,r_4\}, \{r_3,r_6\}, \{r_3,r_7\}, \{r_4,r_5\}, \{r_4,r_6\}, \{r_4,r_7\}, \{r_5,r_6\},\{r_5,r_7\}, \{r_1,r_3,r_6\}, \{r_1,r_3,r_7\}, \{r_1,r_5,r_6\},\{r_1,r_5,r_7\}, \{r_2,r_3,r_6\}, \{r_2,r_5,r_6\}, \{r_2,r_5,r_7\}, \{r_4,r_3,r_6\}, \{r_4,r_3,r_7\}, \{r_4,r_5,r_6\}, \{r_4,r_5,r_7\} \}$

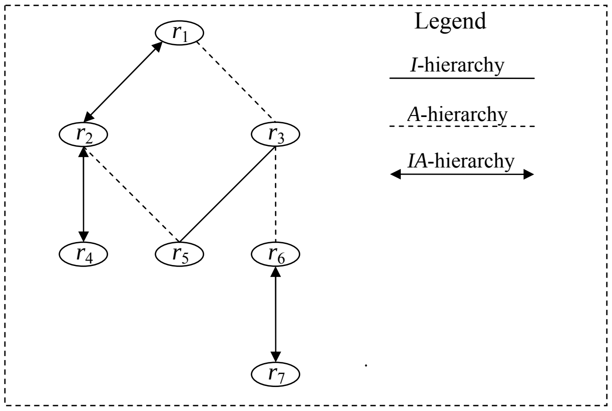


Figure 1. An illustration of hybrid hierarchy and UAS

3. AUTHORIZATION QUERY

Efficient techniques are needed to maintain the permission inheritance and role activation semantics to support efficient administration and management of hybrid hierarchy. One of the most challenging tasks for the administration of the security of hybrid hierarchy is to determine whether to grant a user's request for activating a set of roles. Earlier work has tried to address this problem by building UAS of a hybrid hierarchy by using the DB and DRB approaches mentioned earlier, and presented in [3]. However, both of these approaches are computation intensive. Besides, even with a UAS available, to answer whether a request is in the UAS or not is prohibitively expensive, because the number of elements in a UAS can range over from one to the power set (associated with an *A*-hierarchy) of roles. Neither computing nor storing the whole UAS for a large hierarchy is viable for practical systems.

In this section, we investigate a polynomial time approach by first proving an important property of UAS, and utilizing an *assistant matrix* (AM) that only keeps pair wise UAS information, to support authorization query in a very efficient way. In our approach, we do not actually build the UAS, but only keep the pair-wise UAS information.

3.1 UAS Checking Criterion

We use the pair-wise UAS sets to infer the actual UAS set. The following theorem establishes the required basis for using the AM to make the authorization decision.

Theorem 1: *Let H = (R, F) be a hybrid hierarchy; where R =* $\{r_1, r_2,\ldots r_n\}$ *is a set of roles, F* $\subseteq \{\geq_i, \geq_a, \geq\}$ *is a set of hierarchy relations over R.* Let $R_1 = \{r_1, r_2,\ldots,r_m\}$, *then the following condition holds,*

- $R_1 \in \text{UAS}(H) \Leftrightarrow \{r_i, r_j\} \in \text{UAS}(H)$ $(0 \leq i, j \leq m)$.

The proof for the theorem is as follows.

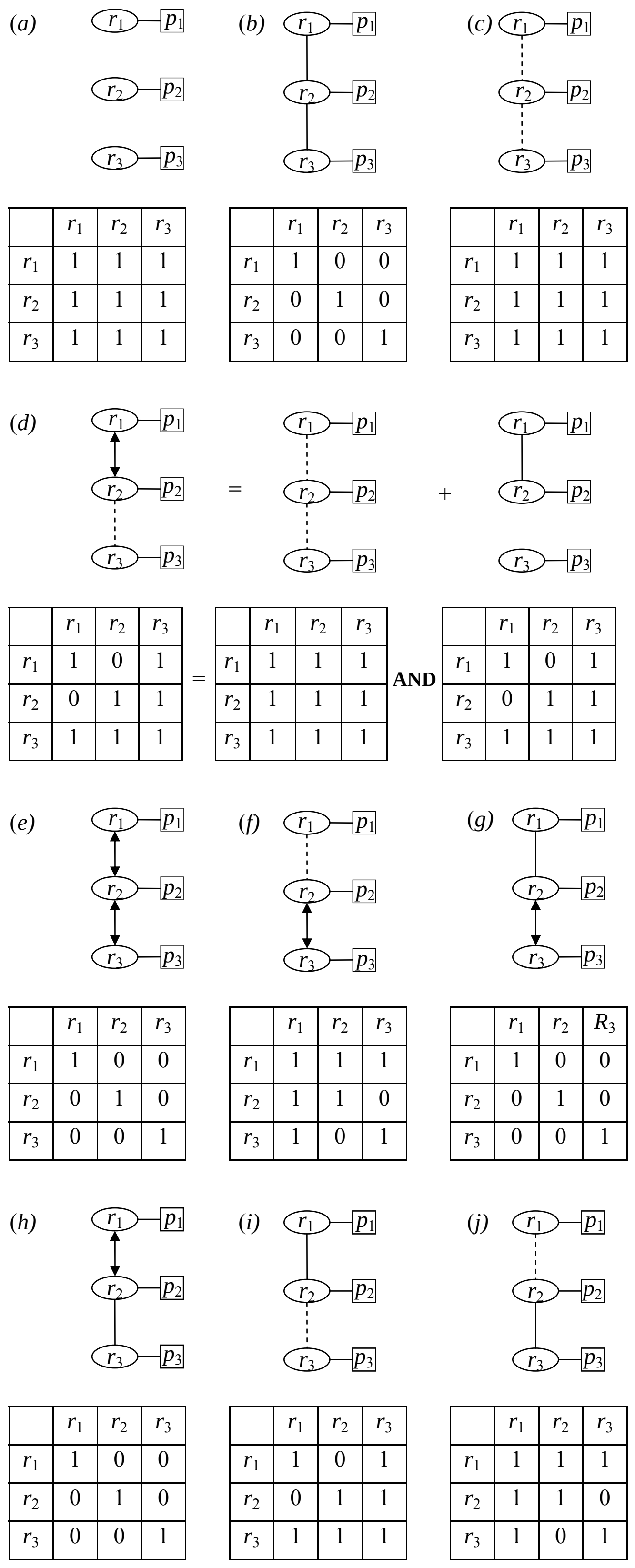
(*Proof for* $\Rightarrow$): If $R_1 \in \text{UAS}(H)$, then each role $r_i \in \{r_1, r_2,\ldots, r_m\}$ is activable. Obviously, each role pair $\{r_i, r_j\}$ for $0 \leq i, j \leq m$, is also activable in a single session. Now, we prove the permission set for each $\{r_i, r_j\}$ for $0 \leq i, j \leq m$, is also unique. If we assume there is one $\{r_i, r_j\}$ for which the permission set is not unique, then a subset of $\{r_i, r_j\}$ should have the same set of permissions as that of the $\{r_i, r_j\}$. Let us assume it to be $\{r_i\}$, then we can replace $\{r_i, r_j\}$ in R_1 with $\{r_i\}$, the resulting set $R'_1 = (r_1, r_2, \ldots, r_m\}\backslash r_j$ should have the same set of permissions same as $P(R_1)$. That means R'_1 is a smaller set than R_1 and hence, $R_1 \notin \text{USA}(H)$, which results in a contradiction that $R_1 \in \text{UAS}(H)$. Hence,

$$R_1 \in \text{UAS}(H) \Rightarrow \{r_i, r_j\} \in \text{UAS}(H)\ (0 \leq i, j \leq m)$$

(*Proof for* $\Leftarrow$): For a given set $\{r_1, r_2,\ldots, r_m\}$, if $\{r_i, r_j\} \in \text{UAS}(H)$ $(0 \leq i, j \leq m)$ holds, then each role $r_i \in \{r_1, r_2,\ldots, r_m\}$ is activable. Now we prove uniqueness. If $\{r_i, r_j\} \in \text{UAS}(H)$ $(0 \leq i, j \leq m)$, assume the corresponding permission set for each $\{r_i, r_j\}$ is p_i, which is minimized by the definition of UAS, then the corresponding permission set for $\{r_1, r_2,\ldots, r_m\}$ is $\bigcup p_i$, which is also minimized. So, $\{r_1,\ldots r_m\} \in \text{UAS}(H)$. Hence,

$$\{r_i, r_j\} \in \text{UAS}\ (H) \Rightarrow R_1 \in \text{UAS}(H)\ (0 \leq i, j \leq m)$$

Therefore, $R_1 \in \text{UAS}(H) \Leftrightarrow \{r_i, r_j\} \in \text{UAS}\ (H)\ (0 \leq i, j \leq m)$.



(a)

	r_1	r_2	r_3
r_1	1	1	1
r_2	1	1	1
r_3	1	1	1

(b)

	r_1	r_2	r_3
r_1	1	0	0
r_2	0	1	0
r_3	0	0	1

(c)

	r_1	r_2	r_3
r_1	1	1	1
r_2	1	1	1
r_3	1	1	1

(d)

	r_1	r_2	r_3
r_1	1	0	1
r_2	0	1	1
r_3	1	1	1

=

	r_1	r_2	r_3
r_1	1	1	1
r_2	1	1	1
r_3	1	1	1

AND

	r_1	r_2	r_3
r_1	1	0	1
r_2	0	1	1
r_3	1	1	1

(e)

	r_1	r_2	r_3
r_1	1	0	0
r_2	0	1	0
r_3	0	0	1

(f)

	r_1	r_2	r_3
r_1	1	1	1
r_2	1	1	0
r_3	1	0	1

(g)

	r_1	r_2	R_3
r_1	1	0	0
r_2	0	1	0
r_3	0	0	1

(h)

	r_1	r_2	r_3
r_1	1	0	0
r_2	0	1	0
r_3	0	0	1

(i)

	r_1	r_2	r_3
r_1	1	0	1
r_2	0	1	1
r_3	1	1	1

(j)

	r_1	r_2	r_3
r_1	1	1	1
r_2	1	1	0
r_3	1	0	1

Figure 2. Relationship between hierarchy graph and AM

3.2 Assistant Matrix

UAS is defined as the uniquely activable set for a user assigned to the senior most role. Here, we shall extend the definition for a generic hierarchy. We construct an AM to maintain the pair-wise UAS information. We formally characterize the AM as per definition 2.

Definition 2: ***Assistant Matrix* (AM)**: *For a given hybrid hierarchy $H = (R, F)$, where $R = \{r_1, r_2, \ldots r_n\}$ is a set of roles*, and $F \subseteq \{\geq_i, \geq_a, \geq\}$ *is a set of hierarchy relations*, AM *is defined as follows,*

$$\forall role\ r_i, r_j, \quad \begin{array}{l} if\ r_i = r_j,\ \mathrm{AM}[i][j] \leftarrow 1 \\ if\ \{r_i, r_j\} \subseteq \mathrm{UAS}(H) \Rightarrow \mathrm{AM}[i][j] = 1 \\ if\ \{r_i, r_j\} \not\subset \mathrm{UAS}(H) \Rightarrow \mathrm{AM}[i][j] = 0 \end{array}$$

Note that AM is a symmetric matrix. Now, we look at the relationship between hierarchical structure and the AM as illustrated in Figure 2. Figure 2(*a*) depicts a case that has no hierarchical structure. We use it for construction purpose. If there is no relationship between two roles, both of them can be in the UAS. So, the AM matrix for this case is essentially a unit matrix. Figure 2(*b*) shows a monotype *I*-hierarchy. In this kind of hierarchy, if a user is assigned to a role r_1, then the corresponding UAS is $\{r_1\}$ with permissions $\{p_1, p_2, p_3\}$. If a user is assigned to role r_2, then the corresponding UAS is $\{r_2\}$ with permissions $\{p_2, p_3\}$, and similarly, we get another UAS = $\{r_3\}$ with permission $\{p_3\}$ when a user is assigned to role r_3. The corresponding AM is an identity matrix as shown under the graph.

Figure 2(*c*) shows a monotype *A*-hierarchy. In this kind of hierarchy, the UAS for user assigned to r_1 is the power set of the roles, $\{\{r_1\},\{r_2\}, \{r_3\}, \{r_1, r_2\}, \{r_1, r_3\}, \{r_2, r_3\},\{r_1, r_2, r_3\}\}$. For a user assigned to r_2, the corresponding UAS is $\{\{r_2\}, \{r_3\}, \{r_2, r_3\}\}$. For a user assigned to r_3, the corresponding UAS is $\{r_3\}$.

Figure 2(*d*) illustrates a hybrid hierarchy that can be constructed from the three basic structures in figures 2(*a*), 2(*b*) and 2(*c*). In [13], it has been proved that a hybrid hierarchy can be split into an *I*-hierarchy and an *A*-hierarchy because of the fact that an *IA* relation represents the presence of both the *A* and *I* relations. This is captured by the fact that the AM of a hybrid hierarchy is equal to the logical *and* of the AM of the corresponding *I*-hierarchy and the corresponding *A*-hierarchy.

Figures 2(*e*)-(*j*) illustrate all other possible basic linear hybrid hierarchies for three roles and their corresponding AMs.

Based on the relationship between hierarchy graph and AM, we next present algorithm **BUILD-AM()** that produces an AM from a hybrid hierarchy graph. The algorithm divides role relationships into four categories corresponding to four categories of cells in the AM.

(a) The diagonal element of the AM is "1" by definition.

(b) If there is a direct *A*-hierarchy relationship ($r_i \geq_a r_j$) between two roles, the AM cell value is set to "1". If there is a direct *I*-hierarchy or *IA*-hierarchy ($r_i \geq_i r_j$ or $r_i \geq r_j$) relationship between two roles, the AM cell value is set to "0".

(c) Note that there may be multiple paths between two nodes. In such a case, the *derived* relation is the combination of the two relations derived through each path [13]. For instance, if one path derives an *I*-relation and another path derives an *A*-relation between the same two roles, then the derived relation between them is actually an *IA*-relation; hence, the AM entry will correspond to *IA*-relation.

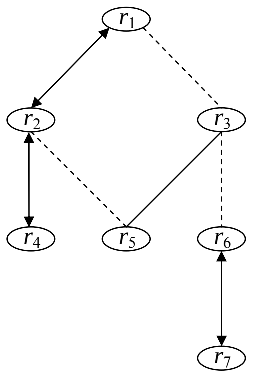
(d) All other roles are not related to each other, which means they can always be activated by different user assignments, so the AM cell is set to "1".

The time complexity of **BUILD-AM()** is $O(n^2)$, where n is the total number of roles in a given hybrid hierarchy.

```
BUILD-AM (Hierarchy_Graph g)
Input: g — the graph representation of the hybrid hierarchy structure
Output: Assistant Matrix, am
initialize am[n][n] ={-1}
foreach role $r_i$ in g
    foreach role $r_j$ in g
        if $i = j$
            $am[i][j] \leftarrow 1$
        if $r_i \geq_a r_j$
            $am[i][j] \leftarrow 1$
            $am[j][i] \leftarrow 1$
        else if ($r_i \geq_i r_j$) or ($r_i \geq r_j$)
            $am[i][j] \leftarrow 0$
            $am[j][i] \leftarrow 0$
        if $r_j$ is_non_direct_offspring_of ($r_i$)
            if there is $r_i \geq_a r_j$ on the path
                if $am[i][j]$ unfilled
                    $am[i][j] \leftarrow 1$
                    $am[j][i] \leftarrow 1$
            else
                $am[i][j] \leftarrow 0$
                $am[j][i] \leftarrow 0$
        if $am[i][j] = -1$
            $am[i][j] \leftarrow 1$
            $am[j][i] \leftarrow 1$
return am
```

Figure 3. Assistant matrix building algorithm

`Example 2`: Figure 4 shows the application of **BUILD-AM()** to build an AM from a hybrid hierarchy. On the left hand side is the graphic representation of a hybrid hierarchy with 7 roles. On the right hand side is the corresponding AM built from the graphic representation according to **BUILD-AM()**.



	r_1	r_2	r_3	r_4	r_5	r_6	r_7
r_1	1	0	1	0	1	1	1
r_2	0	1	1	0	1	1	1
r_3	1	1	1	1	0	1	1
r_4	0	0	1	1	1	1	1
r_5	1	1	0	1	1	1	1
r_6	1	1	1	1	1	1	0
r_7	1	1	1	1	1	0	1

Figure 4. An example of AM building

3.3 Authorization Query Algorithm

We use the UAS checking criterion shown in Theorem 1 and the AM proposed earlier to develop a simple and efficient algorithm **AUTH_QUERY()** to determine whether a set of roles can be uniquely activated or not. The essential part of the algorithm is to check the pair-wise relations which have been stored in the AM by applying algorithm **BUILD_AM()**. For a request to activate a set of roles $RQ = \{r_1, r_2,\ldots, r_m\}$, if at least any pair of roles, $\{r_i, r_j\}(0 \leq i, j \leq m)$ is not in the UAS then the requested set RQ can not be in UAS. The requested role set RQ can be in a UAS if and only if all pairs of roles $\{r_i, r_j\} \subseteq RQ$ $(0 \leq i, j \leq m)$ are in the UAS.

The time complexity of **AUTH_QUERY()** is $O(n^2)$, and n is the total number of roles in a given hybrid hierarchy.

```
AUTH-QUERY(RQ, AM am)
Input: RQ = {r_1, r_2,…, r_m}— the query, a set of roles roles to
                                 be activated in a session
       am — the AM built from algorithm Build-AM()
Output: TRUE if RQ uniquely activable
        FALSE if RQ not uniquely activable
1     $S_{rest} \leftarrow RQ$
2        foreach $r_i \in RQ$
3           $S_{rest} \leftarrow S_{rest} \setminus \{r_i\}$
4           foreach $r_j \in S_{rest}$
5              if $em[i][j] = 0$
6                 return FALSE
7     return TRUE
```

Figure 5. Authorization query algorithm

4. INTER-DOMAIN ROLE MAPPING

In this section, we address the inter-domain policy mapping (IDRM) problem. We first show that finding a minimal set of roles that matches a given set of permissions is *NP-complete*. Then we propose some enhanced heuristics to get a sub-optimal solution.

4.1 The IDRM Problem

An important requirement of emerging system is to be able to share information with other systems [1, 8, 10]. When a system needs to allow previously unknown entities to access its resources, mechanisms should be in place to ensure that the accesses granted are limited to pre-defined sharing requirements. We emphasize that a requirements-driven interoperation is needed in a loosely coupled environment, and there should be an efficient mechanism to facilitate an external entity to access a local domain's resources by mapping external entities to local entities [16]. We assume use of RBAC-based policies in such interacting domains.

Figure 6 illustrates an inter-domain interoperation scenario [16]. Assuming two domains interoperate, each domain first sends the access requirements to the other. Once the requirements have been received, the requests are fulfilled by identifying the set of roles that can satisfy the requested permissions. The goal is to find a minimal set of roles that match the requested set of permissions. In an earlier work, we have proposed designating special external roles that are mapped to exported roles through an A-hierarchy relation – this semantically means that the external entity has to activate the specified exported roles in the provider domain. The exported roles are themselves made I-seniors of other local roles that satisfy the requested accesses to ensure that the external entities can not activate other local roles. By using these *A* and *I* hierarchy structures, we can prevent the transitivity of the activation semantics that is usually the underlying problem in inter-domain access [8]. For more details on this methodology, we refer the readers to [16].

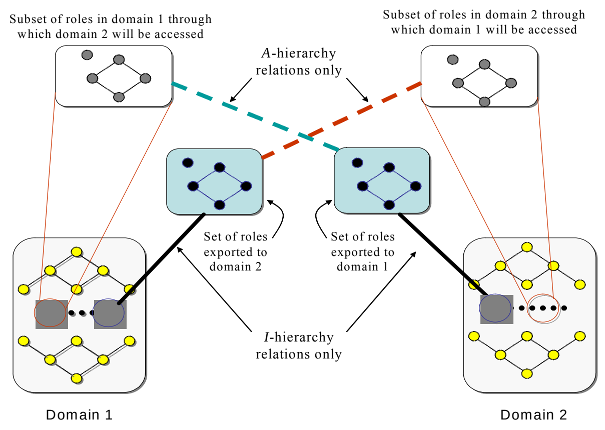

Subset of roles in domain 1 through which domain 2 will be accessed
Subset of roles in domain 2 through which domain 1 will be accessed
A-hierarchy relations only
Set of roles exported to domain 2
Set of roles exported to domain 1
I-hierarchy relations only
Domain 1
Domain 2

Figure 6. Inter-domain interoperation scenario

We represent a domain requested permission set as RQ. Our goal is to find the minimal set of roles that can provide the RQ. Formally,

Inter-domain Role Mapping (IDRM) Problem: *For a request* $RQ = \{p_1,p_2,\ldots,p_m\}$ *and a given hybrid hierarchy* $H = (R, F)$, *where* $R = \{r_1, r_2,\ldots r_n\}$ *is a set of roles and* $F \subseteq \{\geq_i, \geq_a, \geq\}$ *is a set of hierarchy relations, find the minimal set of roles* $R' \subseteq R$, *such that* $P_{au}(R') = RQ$.

4.2 IDRM is NP-Complete

In a monotype *I*-hierarchy, as the permissions of a junior role *can_be_acquired* by senior roles, the role hierarchy can facilitate a top-down scan to solve the IDRM problem. However, the presence of a hybrid hierarchy presents a more complicated and realistic model, in which a senior role may not have more permissions than a junior role (as illustrated in the Example 3). Here, we take a set-based approach to solve the IDRM problem.

Without loss of generality, we can simplify the IDRM problem by assuming $R_1 \subseteq R$, such that, $\forall r, r \in R_1 \Rightarrow P_{au}(r) \subseteq RQ$, and $P_{au}(R_1) = RQ$. This can be done in linear time by removing all the roles r in R such that $P_{au}(r) \not\subset RQ$. Then the problem is how can we find the minimal set of roles $R' \subseteq R_1$, such that $P_{au}(R') = RQ$.

Obviously, without the condition of finding the minimal set of roles, the simplest way to solve this problem is to first select one role from R_1 arbitrarily (there are total n choices, assuming $|R_1| = n$), then selecting another role from the rest of R_1, and so on, until the selected set R' satisfies $P_{au}(R') = RQ$. However, with the minimal set of roles condition, the question becomes hard.

When given a set of roles R, we can quickly check to determine whether $P_{au}(R) = RQ$ or not. This can all be done in linear time. Hence, Q_1 is in NP.

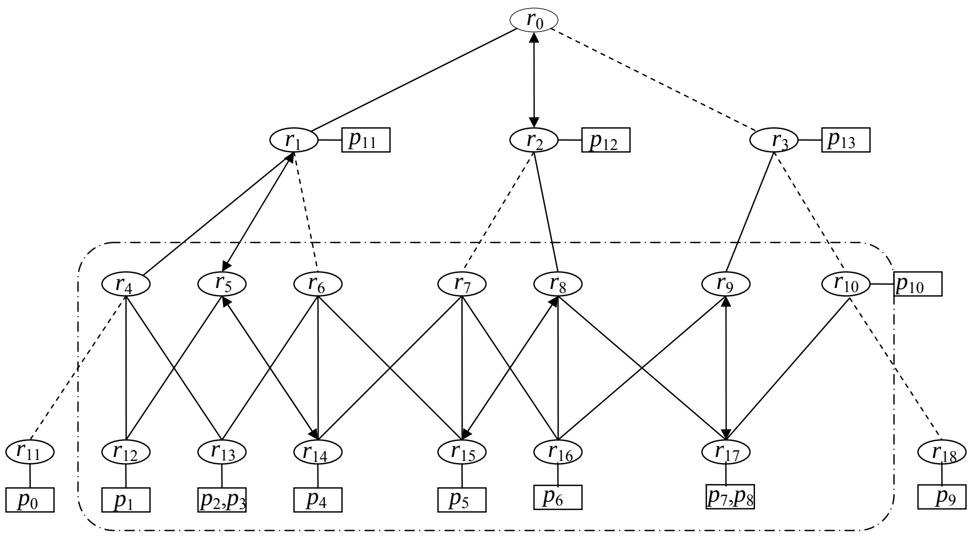


Figure 7. An example of greedy search in hybrid hierarchy

However, there is no polynomial time solution to find the minimal set of roles that exactly cover the requested set of permissions. In essence, the IDRM problem is *NP-complete*. To prove this, we reduce a classic *NP-complete* problem *Minimal Set Cover* (MSC) with uniform cost to the IDRM problem.

Minimal Set Cover (MSC) [4]: *Given a collection C of subsets of a finite set S such that every element in S belongs to at least one member of C* ($S = \bigcup_{c_i \in C} c_i$), *the MSC problem is to find* $C' \subseteq C$ *that satisfies the following:*

- *C' is a set cover for S, i.e.,* $S = \bigcup_{c_i \in C} c_i$ *, and*
- *the cardinality of C', i.e.,* $|C'|$ *is minimized.*

We do the following simple construction. Let each e_i in S correspond to a p_i in RQ. Then $C = \{c_1, c_2, \ldots, c_m\}$ maps to $R_1 = \{r_1, r_2, \ldots r_m\}$, such that c_i corresponds to $P(r_i)$ (Note that, $\forall r, r \in R_1 \Rightarrow P_{au}(r) \subseteq RQ$, and $\bigcup P_{au}(r_i) = RQ$). The mapping can be easily shown in polynomial time. Hence, the set R' discussed earlier now corresponds to C'. Hence, if we can find R' in polynomial time, we also solve the MSC problem in polynomial time.

```
GREEDY-SEARCH (R, RQ)
Input: R -- a set of roles,
Output: R* -- set of roles, such that $P_{au}(R^*) = RQ$, ($R^* \subseteq R$)
1   foreach r in R
2       if $P_{au}(r) \subseteq RQ$
3           $R_1 \leftarrow r$
4   $R^* \leftarrow \varnothing$
5   while $RQ \neq \varnothing$ do
6       Find set $V \in R_1 \setminus R^*$ that maximize $P_{au}(V) \cap RQ$
7       $R^* \leftarrow R^* \cup V$
8       $RQ \leftarrow RQ \setminus V$
9   return R*
```

Figure 8. Greedy search algorithm

There are well-known approximation algorithms with time complexity within $1+ln|S|$ for MSC problem [9], which we have adopted in the greedy search algorithm shown in Figure 8 for our IDRM problem. The algorithm does not guarantee to find the optimal solution R'. However, it has been proved that GREEDY-SEARCH algorithm is an H_n-approximation algorithm for the MSC problem. That is,

$$\frac{|R^*|}{|R'|} \leq H(\max\{|V| : V \in R_1\})$$

and $H(d)$ is the d^{th} *harmonic number* [4], which is equal to

$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{d} = \sum_{i=1}^{d}\frac{1}{i} = \log(d)+O(1).$$

`Example 3`: Consider the hybrid hierarchy H shown in Figure 8. Here $R = \{r_1, r_2, \ldots, r_{18}\}$, and

$P_{au}(r_0) = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_{11}, p_{12}\}$,	
$P_{au}(r_1) = \{p_1, p_2, p_3, p_4, p_{11}\}$;	$P_{au}(r_2) = \{p_5, p_6, p_7, p_8, p_{12}\}$,
$P_{au}(r_3) = \{p_6, p_7, p_8, p_{13}\}$,	$P_{au}(r_4) = \{p_1, p_2, p_3\}$,
$P_{au}(r_5) = \{p_1, p_4\}$,	$P_{au}(r_6) = \{p_2, p_3, p_4, p_5\}$,
$P_{au}(r_7) = \{p_4, p_5, p_6\}$,	$P_{au}(r_8) = \{p_5, p_6, p_7, p_8\}$,
$P_{au}(r_9) = \{p_6, p_7, p_8\}$,	$P_{au}(r_{10}) = \{p_7, p_8, p_{10}\}$,
$P_{au}(r_{11}) = \{p_0\}$,	$P_{au}(r_{12}) = \{p_1\}$,
$P_{au}(r_{13}) = \{p_2, p_3\}$,	$P_{au}(r_{14}) = \{p_4\}$,
$P_{au}(r_{15}) = \{p_5\}$,	$P_{au}(r_{16}) = \{p_6\}$,
$P_{au}(r_{17}) = \{p_7, p_8\}$,	$P_{au}(r_{18}) = \{p_9\}$.

Table 3. Results for each step of Example 3

Step 1	Step 3	Step 5
$R^* = \varnothing$ $V = r_6$	$R^* = \{r_6, r_8\}$ $RQ = \{p_1, p_{10}\}$ $V = r_4$	$R^* = \{r_6, r_8, r_4, r_{10}\}$ $RQ = \varnothing$
Step 2	**Step 4**	
$R^* = \{r_6\}$ $RQ = \{p_1, p_6, p_7, p_8, p_{10}\}$ $V = r_8$	$R^* = \{r_6, r_8, r_4\}$ $RQ = \{p_{10}\}$ $V = r_{10}$	

Assuming $RQ = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_{10}\}$, **GREEDY-SEARCH()**algorithm first constructs $R_1 = \{r_4, r_5, r_6, r_7, r_8, r_9, r_{10}, r_{12}, r_{13}, r_{14}, r_{15}, r_{16}, r_{17}\}$. Then the results for each step in the *while* loop are as shown in Table 3.

Example 3 illustrates the application of **GREEDY-SEARCH()** algorithm in a hybrid hierarchy shown in Figure 7. There are 18 roles and the request is for 9 permissions. The solution $R^* = \{r_6, r_8, r_4, r_{10}\}$, with cardinality $|R^*|= 4$, returned by **GREEDY-SEARCH()** algorithm, is not optimal. The optimal solution is $R' = \{r_4, r_7, r_{10}\}$ with cardinality $|R'| =3$. On the other hand, we have $H(\max\{|V| : V \in X_1\}) = H(4) \approx 2.083$. So the upper-bound of cardinality of the solutions returned by **GREEDY-SEARCH()** algorithm is $|R'| * H(4) = 3 * 2.083 \approx 6.25$. Hence, the **GREEDY-SEARCH()** algorithm guarantees that at most a set of 6 roles can provide the required set of permissions.

```
PROBABILISTIC-GREEDY-SEARCH($R$ ,$RQ$)
Input: $R$ -- a set of roles,
Output: $R^*$ -- minimal set of roles, such that
          $P_{au}(R^*) = RQ, (R' \subseteq R)$
1    foreach $r$ in $R$
2        if $P_{au}(r) \subseteq RQ$
3            $R_1 \leftarrow r$
4    $R^* \leftarrow \varnothing$
5    while $RQ \neq \varnothing$ do
6        with probability $p$
7            Find set $V \in R_1 \setminus R^*$ that maximize $P_{au}(V) \cap RQ$
8        with probability 1-$p$
9            Randomly find a set $V \in R_1 \setminus R^*$
10       $R^* \leftarrow R^* \cup V$
11       $RQ \leftarrow RQ \setminus V$
12   return $R^*$
```

Figure 9. Enhanced greedy search algorithm

4.3 Enhanced Greedy Search Algorithm

Greedy search algorithm used above suffers from local maxima problem [4]. One possible approach to alleviate this problem is by introducing a random parameter, such as in the *Simulated Annealing* algorithm, which has been used for optimization problems in the literature [18].

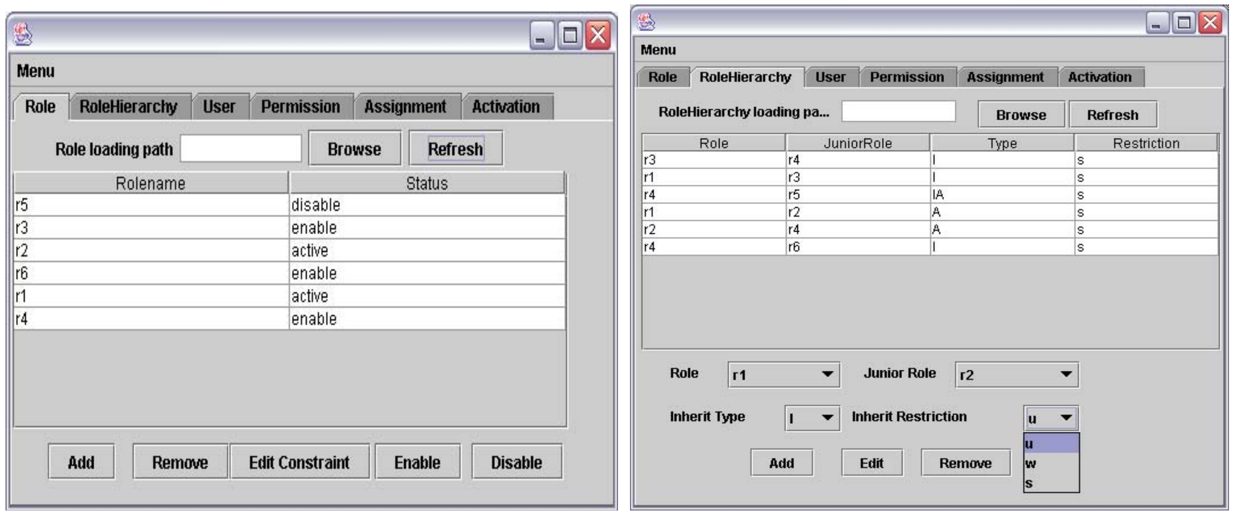
In the **PROBABILITY-GREEDY-SEARCH()** algorithm, shown in Figure 9, we enhance the earlier algorithm with a probability parameter. With probability p (usually near 1), the algorithm will execute the statement 7 just like in **GREEDY-SEARCH()** algorithm. With probability 1-p, the algorithm will randomly select a candidate set, which helps the **GREEDY-SEARCH()** algorithm avoid local maxima. The algorithm can be run multiple times in order to get the best result. **PROBABILISTIC-GREEDY-SEARCH()** algorithm has both the flavor of simulated annealing search and genetic algorithm, but with much less computational overhead.

5. IMPLEMENTATION

We have implemented our algorithms in our java based GTRBAC prototype system where we had tested our earlier proposed DB and DRB approaches [3]. The prototype has been extended to include trust-based requirements driven policy mapping in loosely coupled mobile environment. We plan to use Blackberry devices to implement secure interoperation between two mobile security domains for simple applications using the proposed algorithm.

6. RELATED WORK

Several research efforts [2, 5, 8] have been devoted to the topic of policy composition and secure interoperation in multi-domain environment. In [19], an integer programming approach has been proposed to allow policy integration between multiple RBAC policies. More relevantly, [3] has tried different approaches to facilitate the administration of role hierarchy by constructing the actual UAS set. Two techniques have been proposed for computing the UAS of a hierarchy and compared. The DB approach constructs the UAS by composing the computed UASs of sub-hierarchies. The DRB approach, on the other hand, uses a set of implication rules, (we refer to [13] for the details), to derive hierarchical relations between every pairs of roles in the hierarchy and then compute the UAS from them. While the first approach is slightly better in terms of time complexity, both these approaches are non-polynomial solutions. In [20], Shehab *et al.* proposed a



Rolename	Status
r5	disable
r3	enable
r2	active
r6	enable
r1	active
r4	enable

Role	JuniorRole	Type	Restriction
r3	r4	I	s
r1	r3	I	s
r4	r5	IA	s
r1	r2	A	s
r2	r4	A	s
r4	r6	I	s

Figure 10. Snapshot of the our hybrid hierarchy administration

distributed secure interoperability protocol that ensures secure interoperation of the multiple collaborating domains without compromising the security of collaborating domains. They use access paths and path discovery algorithms to query interoperating domains for the set of secure access paths between different domains. No requirement based mapping has been addressed. In [16], the authors propose a breadth-first-search-based algorithm for policy mapping between two loosely coupled interacting domains for sharing resources. However, the algorithm proposed does not do an exhausted search; instead, it creates new roles even if there is possible a combination of roles in the local domain that can satisfy the requested permissions. Other earlier work related to hybrid hierarchy that highlight its importance can be found in [11, 17].

7. CONCLUSION AND FUTURE WORK

In this paper, we have investigated the key problems introduced by hybrid hierarchy. While a hybrid hierarchy is important to make an RBAC approach generic enough to capture very diverse set of access requirements as well as to support flexible policy expression and inter-domain policy mapping, it introduces complexity in terms of the maintenance of the overall policy. We addressed two key problems in presence of hybrid hierarchy. The first problem relates to determining whether a user's request for activating a set of roles can be granted or not. The second problem is to find a set of roles that may belong to a hierarchy that satisfies a request for a set of permissions, which needs to be solved to address ad-hoc policy mapping in loosely coupled environments. We have shown that finding the minimal set of roles that matches the requested permission set is *NP-complete* and then presented an efficient heuristic algorithm with a flavor of simulated annealing and genetic algorithms. We plan to use these results to address (*i*) the issue of the management of an RBAC policy as it evolves – here administering hybrid hierarchies is a key challenge, and (*ii*) the problem of efficiently mapping security policies to facilitate secure interoperation in loosely coupled interactions. The result for the query problem needs to be extended and applied within the context of a generic GTRBAC policy with SoD and cardinality constraints.

Acknowledgement: This research has been supported by the US National Science Foundation award IIS-0545912. We thank the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] Biskup, J., Flegel, U., Karabulut, Y. *Secure Mediation: Requirements and Design.* Proceedings of 12th Annual IFIP WG 11.3 Working Conference on Database Security, 1998.

[2] Bonatti, P.A., Sapino, M. L., Subrahmanian, V.S. *Merging Heterogeneous Security Orderings.*Esorics'96,183-197.

[3] Chandran, S.M., Joshi, J.B.D. *Towards Administration of a Hybrid Role Hierarchy*, IEEE International Conference on Information Reuse and Integration, 2005.

[4] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. Introduction to Algorithms, Second Edition, MIT press, 1991.

[5] Dawson, S., Qian, S., Samarati, P. *Providing Security and Interoperation of Heterogeneous Systems*. International Journal of Distributed and Parallel Databases. 2000.

[6] Ferraiolo, D. F., Gilbert, D. M., Lynch, N. *An Examination of Federal and Commercial Access Control Policy Needs*, NISTNCSC National Computer Security,1993, 107-116.

[7] Ferraiolo, D.F, Kuhn, D.R. *Role Based Access Control*. 15th National Computer Security Conference,1992.

[8] Gong, L., Qian, X. *Computational Issues in Secure Interoperation*. IEEE Transaction on Software and Engineering, Vol. 22, No. 1, January 1996.

[9] Johnson, D.S., *Approximation algorithms for combinatorial problems*. J. Comput. System Sci. 9, 1974. 256-278.

[10] Joshi, J.B.D., Aref, W.G., Ghafoor, A., Spafford, E.H. *Security Models for Web-based Applications.* Communications of the ACM, 44, 2, 2001,38-72.

[11] Joshi, J.B.D., Bertino, E., Ghafoor, A. *Temporal hierarchies and inheritance semantics for GTRBAC*, Proceedings of the seventh ACM symposium on Access control models and technologies, Monterey, California, USA, 2002, 74 – 83.

[12] Joshi, J.B.D., Bertino, E., Latif, U., Ghafoor, A. *A Generalised Temporal Role Based Access Control Model*. IEEE Transactions on Knowledge and Data Engineering, 17(1), 2005, 4-23.

[13] Joshi, J.B.D., Bertino, E., Ghafoor, A. *Formal Foundation for Hybrid Hierarchies in GTRBAC*. ACM Transactions on Information and System Security. (revised submission).

[14] Joshi, J.B.D., Bertino, E., Ghafoor, A. *An Analysis of Expressiveness and Design Issues for the Generalized Temporal Role-Based Access Control Model,* IEEE Transactions on Dependable and Secure Computing, Vol. 2, No. 2, April 2005.

[15] Osborn, S., Sandhu, R., Munawer, Q. *Configuring Role-based Access Control to Enforce Mandatory and Discretionary Access Control Policies*. ACM Transactions on Information and System Security, 3(2), 2000, 85-106.

[16] Piromruen, S., Joshi, J.B.D. *An RBAC Framework for Time Constrained Secure Interoperation in Multi-domain Environment*, IEEE Workshop on Object-oriented Real-time Databases (WORDS-2005), 2005.

[17] Sandhu, R. *Role hierarchies and constraints for lattice-based access controls*. Computer Security - Esorics'96, LNCS N. 1146, 1996, 65-79.

[18] Sen, S. *Minimal cost set covering using probabilistic methods*. Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice. Indianapolis, Indiana, United States, 1993, 157-164.

[19] Shafiq, B., Joshi, J.B.D., Bertino, E., Ghafoor, A. *Secure Interoperation in a Multi-Domain Environment Employing RBAC Policies*, Knowledge and Data Engineering, 2005.

[20] Shehab, M., Bertino, E., Ghafoor, A. *SERAT: SEcure Role mApping Technique for Decentralized Secure Interoperability*, In Proceedings of the ACM Symposium on Access Control, Models and Technologies (SACMAT 05), Stockholm, Sweden, 2005.