Hierarchical domains for decentralized administration of spatially-aware RBAC systems

Maria Luisa Damiani
University of Milan, Italy
EPFL, Switzerland
maria.damiani@dico.unimi.it

Claudio Silvestri
University of Milan, Italy
silvestri@dico.unimi.it

Elisa Bertino
Purdue University, USA
bertino@cs.purdue.edu

Abstract

Emerging models for context-aware role-based access control pose challenging requirements over policy administration. In this paper we address the issues raised by the decentralized administration of a spatially-aware access control model in a mobile setting. We present GEO-RBAC Admin, the administration model for the GEO-RBAC model. The model is based on the notion of hierarchy of spatial domains; a spatial domain is an entity grouping objects based on organizational and spatial proximity criteria. In the paper we formally define the model and introduce and prove relevant properties.

1 Introduction

The administration of RBAC (Role Based Access Control) policies in large organizations is an important research issue, which is gaining new momentum under the push of new emerging paradigms like context-aware access control models.

Typically access control policies are administered in accordance with an *administrative model*. An administrative model defines a number of functions for the policies administration and which subjects can enter these policies; it is thus the basis on which the policy specification language is defined. For example, the administration model of the RBAC standard defines a set of functions comprising *administrative functions* for the creation and maintenance of the element sets and relations of the RBAC system, and *review functions* to review the effects of the administrative actions. In most cases administration is *centralized*, that is, the policy is administered by a unique administrator [5].

Unfortunately RBAC, when applied to large organizations consisting of thousands of roles and users [12], does not scale well because it lacks mechanisms for the modular organization of roles and the decentralized management of policies. Note that we use the terms administration decentralization and administration delegation as synonyms to mean that *someone has the authority of assigning administrative functions to somebody else* and thus of “distributing” the administration tasks.

To overcome such limitation, a promising approach is to extend RBAC with the notion of *administrative domain* (simply *domain* hereinafter). A domain basically denotes a portion of the overall policy which is administered by one or more autonomous administrators. Unfortunately, there is no consensus on the meaning of domain in RBAC. Even less investigated is the notion of domain in the various context-aware extensions of RBAC.

An important category of context-aware RBAC models is represented by the *location-based models*. A model is location-based if the authorization to access depends on the position of the user in a reference space. For example, one can state that a doctor is authorized to access the patients' records only when inside the hospital. Location-based policy enforcement is motivated by the need of strong control over information access in a mobile context for privacy purposes, protection of highly sensitive data or marketing reasons. A number of location-based access control models have been recently proposed [1, 8, 4].

Now consider a large organization, for example a health organization consisting of several clinics, willing to adopt a location-based access control policy for protecting very sensitive health records in clinics. A central management of such policy may be extremely complex. To address such issue, we introduce the notion of domain in location-based policies. To make the discussion more concrete, we refer to a specific location-based model, GEO-RBAC [4]. Like RBAC, GEO-RBAC is a family of models. Moreover, roles in GEO-RBAC have a complex structure comprising attributes, role types and spatial objects.

The geographical location represents a natural element of cohesion inside a community or an organization, Therefore, according to the approach by Kern & al. [7], we define a GEO-RBAC domain as *an entity which collects ob-*

jects according to some characteristic, such as the organizational structure, **and spatial proximity**. We highlight the property of *spatial proximity* to point out that a domain represents some sector of an organization which can be associated with a location and in which objects and users are physically close to each other. From this ontological perspective, it follows that the domain model must have a spatial connotation. Further we envision a scenario in which domains can be dynamically decomposed in smaller domains (*sub-domains*), so to enable a more flexible administration. Important questions in the definition of such model are: which kind of relationship exists between a domain and its sub-domains? Which is the degree of autonomy of sub-domain administrators? To address these issues, we define *GEO-RBAC Admin* a decentralized administration model for the Core GEO-RBAC model.

The key concept of *GEO-RBAC Admin* is that of *spatial domain*. A spatial domain is a first class entity which is associated with a reference space and a *owner role*; the owner role is the unique role having an authority over the domain. GEO-RBAC Admin combines the concept of ownership typical of discretionary access control models with the concepts of domains and roles to support decentralized administration. Despite of the focus on GEO-RBAC, we believe that the proposed approach is of more general applicability, and can be generalized for use in different authorization administration models.

This paper is organized as follows. In the next section we overviews Core GEO-RBAC and related work; then we present the key design choices in *GEO-RBAC Admin*; the model is formally described in the subsequent section. Final remarks conclude the paper.

2 Background and related work

2.1 GEO-RBAC

The model is based on the central concept of *spatial role*. A spatial role is a spatially confined organizational function. A spatial roles becomes effective only when the user who has been assigned that role is located in the *role extent*. In such case the role is said to be enabled. Both the role extent and the user positions are described by spatial objects (i.e. *spatial features* compliant with geo-spatial standards) of the type specified in the *role schema*. The spatial entities which are defined in a GEO-RBAC policy are assumed to have a geometric extent which falls inside a (possibly bounded) *reference space*. Such reference space defines the spatial scope of the organization.

The role schema is the template for spatial roles. For example the role schema for doctor, specified as *doctor(HospitalType, LocationType, Lmf)*, is read as follows: *doctor* is the role name, *HospitalType* is the role extent type; *LocationType* is the type of *logical position* which describes the position of the user at a certain granularity, for example building and room; *Lmf* is the *location mapping function* which maps the actual position acquired through some location sensing technology to the logical and abstract location. We refer the reader to [4] for more details.

2.2 Related work

The spatial dimension of domains is a novel concept that has not been investigated before. Related work is thus about the decentralization of administration in classical access control models. In this section we overview significant approaches focusing, in particular, on the following two questions:

- How can domains be represented in RBAC systems?
- Who can delegate the administration of what to whom?

How can domains be represented?

In ARBAC97 [11] a domain is indirectly represented through the notion of *role range*, that is, a pair of roles confining the portion of role hierarchy, namely the set of roles, that an administrator can administer. Role hierarchy characterizes also the administrative model proposed by Crampton [3]. Oh and Sandhu [10] propose an administration model, referred to as ARBAC02. ARBAC02 retains the main features of ARBAC97, and adds the concept of organization unit, which represents a group of individuals involved in related tasks. The organization unit is indirectly modeled through the concepts of user pool and permission pool which however have limited generality. A different approach is adopted by UARBAC model by Li and Mao [9]. UARBAC is a family of administrative models for RBAC which consists of a basic model and one extension $UARBAC^P$. $UARBAC^P$ adds constraint-based administrative domains; the basic idea is to assign one or more attributes to each object in the RBAC system and then define administrative domains using constraints on these attributes. The notion of domain is, however, not explicit, in that it can be represented by the value of some arbitrary attributes of objects or even by a combination of values of attributes. Furthermore this approach makes strong assumptions on the nature of the RBAC model, in that domains can only be added if objects are parametrized.

The administration of RBAC-based context-aware access control has been addressed by X-GTRBAC Admin [2]. A salient feature of this model is that domains are first class objects, that is, they have an identifier and can be related to other objects, in much the same way an Internet domain is defined by a name. A policy is then explicitly associated with a domain. Despite its simplicity, this notion of domain is fairly powerful, since it is actually independent from the concepts of the underlying RBAC and thus can be naturally added to a variety of models.

Who can delegate the administration of what to whom? A major mechanism for administration delegation is provided by discretionary access control (DAC) models [6], in that a subject with a certain access permission is able to pass this permission to another subject.

In the decentralized administration models based on RBAC, domains are administered by members of roles. The roles which are enabled to exercise administration functions may be either *administration roles* like in ARBAC97 and X-GTRBAC, or application-dependent roles like in UARBAC. Separating administrative roles from non-administrative roles ensures a stronger security control and thus is preferable in contexts in which strong security is requested. In ARBAC97 and X-GTRBAC the Security System Officer is the only subject allowed to create administrative roles and thus decentralize administration. The delegation hierarchy thus consists of only two levels, the Security System Officer and the application-dependent administrative roles. In large organizations, however, a delegation hierarchy organized according to two levels may be not sufficiently flexible because of the complexity of the organizational structure.

3 Baseline of the approach

Current approaches provide only partial solutions to the requirements posed by the the GEO-RBAC administration. *GEO-RBAC Admin* tries to rigorously introduce the spatial dimension in the notion of domain. We now present the key design choices of *GEO-RBAC Admin*.

3.1 The representation of domains

Following the approach proposed for X-GTRBAC, in our model domains are first class objects. However, unlike X-GTRBAC, a domain, besides a name, has attributes. One attribute is mandatory, namely the *reference space* of the domain. The reference space of domain d is the (possibly bounded) space, which confines the geometric extent of the spatial entities in d. The reference spaces of two domains may overlap or be disjoint, while, as we will see later on, the relationship of containment between the references spaces of two domains is a necessary condition for the two domains to be one the sub-domain of the other. Following the common intuition, a domain can be seen as "container" of objects. Objects can be application-dependent objects or system-defined objects. Notice that we do not make any assumption about the meaning of application-dependent objects, and in particular whether objects are spatial or not.

A domain d is represented by the tuple $d \equiv < id, s >$ where id is the univocal domain identifier, and s the spatial feature of a system-defined type representing the reference space. We denote with *TopDomain* the system-defined domain: $TopDomain \equiv < id_0, TopF >$, where $TopF$ is the system-defined feature which covers the whole Earth. D denotes the set of all domains.

Example 1 *Suppose Italy and Switzerland be the identifiers of two system-defined spatial features. We want to define two domains, one for each country, say MobileI and MobileCH. The two domains take the form:* $< MobileI, Italy >$ *and* $< MobileCH, Switzerland >$

3.2 Domain administration

A domain is administered by the members of one or more administration roles, referred to as *admin roles*. The set of admin roles is disjoint from the set of *regular roles* where *regular* stands for non-administrative. The members of admin and regular roles are *users*. An admin role is uniformly represented as a GEO-RBAC role [1] and conventionally the admin role is a role defined over the extent representing the reference space of the domain. *TopAdmin* is the unique admin role in the Top Domain and it has the entire set of permissions.

Admin roles are instances of *admin schemas*. Admin schemas are used for a specific purpose, that is, to define templates for the specification of admin roles in sub-domains. For example one can define the role schema $AdminUser$, assign it a set of permissions and then create admin roles in sub-domains as instances of this schema, say $AdminUser(Hosp_1), AdminUser(Hosp_2)$, where $Hosp_1$ and $Hosp_2$ are reference spaces of sub-domains. The concept of admin schema supports the modular organization of administrative roles and permissions in sub-domains.

3.3 Administration delegation

Domains are created and administered by admin roles. Let d_2 be a domain created by an admin role ar of a domain d_1; we call d_1 the *parent* domain of d_2 (conversely d_2 is sub-domain of d_1), and role ar the *owner* of d_2. By definition the owner of a domain is unique. The owner role is the unique role that can create and confer and revoke authorizations from admin roles and users in the owned domain. Therefore, it cannot occur that the admin roles in a domain are created by different roles; neither it can occur that the admin role of a domain is assigned permissions by different roles.

[1]Specifically an admin role is a GEO-RBAC *non-spatial* role, because the role extent and the logical position coincide with the reference space and that matches with the definition of non-spatial role.

An admin role can create a domain and delegate its administration if it has been assigned the *delegation permission*. Moreover the member of the creator role can only delegate a subset of the role permissions; therefore it cannot occur that one assigns a permission that one does not hold. The delegation permission, if assigned to the admin role r of a sub-domain, authorizes r to further delegate administration. It can be observed that such model is similar to the model underlying the *grant option* mechanism of discretionary models, in that the subject who has the delegation permission is authorized to further propagate the administration delegation.

3.4 Domain scoping rules

An important issue is to define the rules which govern the visibility and thus accessibility of objects across domains. Since this problem resembles the definition of scoping rules in programming languages, we refer to these criteria of visibility as *domain scoping rules*. Note that we only consider the issue of visibility for system objects. We have identified three possible approaches.

- *Global objects.* A first approach is to have a pool of objects, accessible from all the application domains, and only administered by the members of Top Admin. The task of the domain administrator (i.e. the administrator of a domain different from TopDomain) is basically to define the user-role and role-permission relationships. The drawback of this approach is the limited autonomy of the domain administrator in the management of the domain policy.

- *Local objects.* Objects are *local* to each domain, that is, they can be accessed and administered only inside a unique domain. This approach ensures the maximum autonomy; the drawback is that the objects which are of common interest to many sectors of the organization must be replicated in each domain.

- *Hybrid solution.* A subset of the system objects are global while the remaining system objects are local. The practical effect of this domain scoping rule is the following: when the user connects to the system and specifies the domain in which he or she is registered, the objects which become available are those local to the domain and those which are globally defined. Because of its flexibility, this is the solution adopted in GEO-RBAC.

A problem related with the choice of the hybrid approach is how to establish what is global and what instead is local. In principle, such a choice could exclusively depend on the application needs. However, it can be also driven by usability concerns. Such consideration results from the experiments we have carried out with the prototype of the administration system which enables the creation of various elements of the domain policy which are then stored in a spatial DBMS. We have observed that some objects may be too complex to specify for a "normal" administrator: for example the specification of a *location mapping function*, which ideally can be whatever function returning a logical position out of the real position, entails programming capabilities that administrators do not necessarily have. It seems thus more convenient to let the Top Admin administer these objects, under the assumption that the member of such a role has the needed competences. In conclusion, domain scoping should be defined based on the desired trade-off between simplicity of use and administrative autonomy. In the current model, the domain scoping rules are built-in in the model. Specifically the components of role schemas are global while the other objects are local. A more flexible management of domain scoping rules, however, would be desirable and this aspect will be investigated as part of future researcg.

4 Specification of GEO-RBAC Admin

4.1 Objects and permissions

For the representation of objects and permissions we adapt the formal notation by Li and Mao [9].

System object class	Meaning
C_R	Regular roles class
C_{RS}	Regular role schemas class
C_{AR}	Administrative roles class
C_{ARS}	Administrative role schemas class
C_{EXT}	Role extent types and instances
C_{LPS}	Logical position types and instances
C_{LMF}	Logical mapping functions
C_U	Users class

Table 1. The set SC of system classes

Object classes. Objects describe entities of different nature. It is thus convenient to group them in classes. The set C of classes comprise:

- *System classes.* System classes are those of interest for the administration. The set SC of classes is reported in Table 1. The choice of the granularity of objects, in particular composite objects, that is objects consisting of parts like role schemas is an important design issue. In that respect, for the sake of flexibility, we have defined a class not only for the whole, namely the role schema but also for each part, namely each schema component.

- *Application classes.* The set $AC = \{C_{AP_1}, ..., C_{AP_n}\}$ of application classes is application dependent.

Access modes. Access modes define the type of operations which can be authorized over objects and are specified based on the desired control granularity. We define access modes based on the principle of *reversibility* [9]. Such a principle states that a permission should enable the reversibility of an action; for example the permission for creating on object should also authorize the deletion of the object. The set of access modes over system objects is: $AM = \{admin, assignPrm, assignUser, delegate\}$, where: *admin* allows one to create and delete an object; *assignPrm* is to assign and revoke permissions to roles and role schemas; *assignUser* is to assign and revoke roles to users; *delegate* is to authorize and revoke administration delegation.

The set of access mode over application objects is application dependent. We only assume the access mode $admin$ for creating and deleting an application object.

Permissions. A permission takes the form $[c, a]$ where c is a class of objects and a an access mode. From the previous classification, it follows the distinction between the set $PRMS_S$ of *system permissions* in Table 2 and the set $PRMS_A$ of *application permissions*. In general, each access mode authorizes two or more operations. In particular the *delegate* access mode authorizes the whole set of operations which are needed to delegate (or revoke) administration to the admin roles of a sub-domain.

Permission	Meaning
$[C_R, admin]$	- to create and delete a regular role
$[C_{RS}, admin]$	- to create and delete a regular role schema
$[C_R, assignPrm]$	- to assign and revoke regular permissions to regular roles
$[C_{RS}, assignPrm]$	- to assign and revoke regular permissions to regular role schemas
$[C_R, assignUser]$	- to assign and revoke users to regular roles
$[C_{EXT}, admin]$	- to create and delete role extent types and instances
$[C_{LPS}, admin]$	- to create and delete logical position types and instances
$[C_{LMF}, admin]$	- to create and delete logical mapping functions
$[C_U, admin]$	- to create and delete users
$[C_{AR}, delegate]$	- to create and delete a sub-domain, an admin schema and an admin role for the sub-domain; to assign and revoke a user to the admin role and a permissions to an admin role of sub-domain

Table 2. The set $PRMS_S$ of system permissions

4.2 State of a domain

Each domain has a *state*. The state defines the sets of objects and relations which are accessible in a domain. The state is described at intensional level by the *state schema* and at extensional level by the *state instance* (simply *state*). We introduce the following notation: O denotes the set of all possible objects; function $OBJS(c)$ returns the set of possible objects in O of class c.

Definition 1 (Schema of domain state) *The state schema of a domain is defined by the following tuple of functions:*

- $OB_A : AC \times D \rightarrow 2^O$ *such that* $OB_A(c, d) \subseteq OBJS(c)$. *The function returns the set of application objects of a certain classes in a domain.*
- $OB_S : SC \times D \rightarrow 2^O$ *such that* $OB_S(c, d) \subseteq OBJS(c)$. *As above, but the function returns the set of system objects.*
- $P_A : D \rightarrow 2^{PRMS_A}$ *returns the set of application permissions in a domain.*
- $P_S : D \rightarrow 2^{PRMS_S}$ *returns the set of system permissions in* d *in a domain.*
- $SUA : D \rightarrow 2^{OBJS(C_U)\times(OBJS(C_R) \bigcup OBJS(C_{AR}))}$. *The function returns the set of pairs* (user, role) *in a domain. The role can be regular or administrative.*
- $SPA_I : D \rightarrow 2^{OBJS(C_R)\times PRMS_A}$. *The function returns the set of pairs* (regular role, permission) *in a domain.*
- $SPA_S : D \rightarrow 2^{OBJS(C_{RS})\times PRMS_A}$. *The function returns the set of pairs* (regular role schema, permission) *in a domain.*
- $ASPA_I : D \rightarrow 2^{OBJS(C_{RS})\times(PRMS_S \bigcup PRMS_A)}$. *The function returns the set of pairs* (admin role, permission) *in a domain.*
- $ASPA_S : D \rightarrow 2^{OBJS(C_{ARS})\times(PRMS_S \bigcup PRMS_A)}$. *The function returns the set of pairs* (admin role schema, permission) *in a domain.*

A *state*, denoted as $\mathcal{I}(d)$, is the state schema applied to domain d. Conventionally the admin roles in a domain d have as extent the reference space of d. $\mathcal{I}$ denotes the *state of the domain set*, defined a $\mathcal{I} = \{\mathcal{I}(d), d \in D\}$

The state $\mathcal{I}(TopDomain)$ has the properties described below.

Definition 2 (State of the TopDomain) *The state of the TopDomain satisfies the following properties:*

- $P_A(TopDomain) = PRMS_A$
- $P_S(TopDomain) = PRMS_S$
- $OB_S(C_{AR}, TopDomain) = \{TopAdmin\}$

- $OB_S(C_U, TopDomain) \supseteq \{TopUser\}$
- $ASPA_I(TopDomain) = \{TopAdmin\} \times (PRMS_A \bigcup PRMS_S)$
- $SUA(TopDomain) \supseteq \{[TopUser, TopAdmin]\}$

4.3 Admin hierarchy

The owner of a domain is the only admin role that can administer that domain and the corresponding admin roles and admin users.

The relationship *to-be-owner-of* is captured by the *Admin Hierarchy*. The Admin Hierarchy consists of a partially ordered set of *nodes* where nodes take the form $[d, ar]$ with d a domain and ar an admin role in d or $\perp$ (undefined).

Definition 3 (Admin hierarchy) *Let $\mathcal{I}$ be the state of the domain set. The Admin Hierarchy $AH = (T, \prec)$ in $\mathcal{I}$ verifies the following conditions:*

(1) $T \subseteq \{[d, ar] \mid d \in D,\ ar \in OB_S(C_{AR}, d) \bigcup \{\perp\})$

(2) $\prec \ \subseteq T \times T$ *is a partial order. The ordering* $[d_m, r_m] \prec [d_n, r_n]$ *holds iff* $d_m \neq d_n$, $r_m \neq \perp$ *and one of the following two conditions is true:*

- r_m *is the owner of* d_n
- $\exists [d, r] \in T$ *such that* $[d_m, r_m] \prec [d, r] \prec [d_n, r_n]$

(3) $\forall\ [d, ar] \in T,\ [TopDomain, TopAdmin] \prec [d, ar]$. *[TopDomain,TopAdmin] is the root of AH.*

It can be easily shown that the Admin Hierarchy is a tree. In fact each node represents an admin role for a domain. Since by definition, an admin role in a domain can be only owned by a unique admin role in another domain, it follows that every node, different from the root, has a unique parent node. Therefore since the set of ancestors of a node is totally ordered and has a bottom element (i.e. the root), the admin hierarchy is a tree.

Example 2 *Assume that the member of the TopAdmin has created two domains, named $hosp1\_dom$ and $hosp2\_dom$, representing two hospitals. The rectangles in Figure 1.a correspond to domains $hosp1\_dom$, and $hosp2\_dom$ with reference space $hosp1\_s$ and $hosp2\_s$ respectively. Moreover the TopAdmin has created two admin roles, one for each of the newly created domains, named respectively $admin(hosp1\_s)$ and $admin(hosp2\_s)$. Notice that these two admin roles are instances of the same schema $admin$. The member of the former role has created two sub-domains, $card\_dom$ and er_dom, corresponding to the cardiology and emergency room divisions, over the spaces $card\_s$ and er_s respectively. The former is administered by two admin roles: $sec\_officier(card\_s)$ and $user\_admin(card\_s)$; the latter by a unique role $sec\_officier(er\_s)$. The resulting admin hierarcy is illustrated in Figure 1.b.*

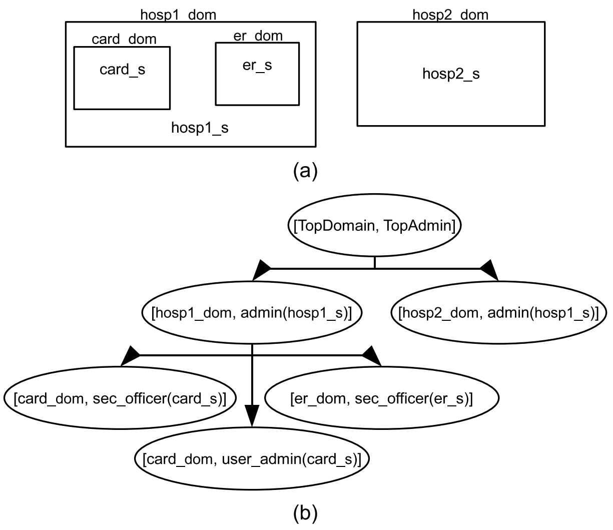
hosp1_dom
card_dom
card_s
er_dom
er_s
hosp1_s
hosp2_dom
hosp2_s
(a)
[TopDomain, TopAdmin]
[hosp1_dom, admin(hosp1_s)]
[hosp2_dom, admin(hosp1_s)]
[card_dom, sec_officer(card_s)]
[er_dom, sec_officer(er_s)]
[card_dom, user_admin(card_s)]
(b)

Figure 1. Admin hierarchy

Finally, we introduce the notion of *policy state*, defined by the pair $\mathcal{I}^* =< \mathcal{I}, AH >$.

4.4 Domain scoping rules

An object obj in domain d is *global* only if it represents a schema component in the state of the TopDomain, that is, $obj \in OB_S(C_o, d) \rightarrow obj \in OB_S(C_o, TopDomain)$ with $C_o \in \{C_{LPS}, C_{LMF}, C_{EXT}\}$. Object obj is *local* to d when obj is only in the state of d and not in the state of any other domain.

Let $Cm = \{C_{LPS}, C_{LMF}, C_{EXT}\}$ and consider the utility function $d\_Prms(r, d)$ returning the set of permissions assigned to admin role r in domain d comprising the permissions both assigned to the role schema and directly to r. The following proposition states some relevant properties of administration hierarchies with respect to scoping rules.

Proposition 1 *The following properties hold:*

- *Global objects cannot be administered by the administrators of application-dependent domains, that is:* $\forall d \in D - \{TopDomain\}, \forall C_o \in Cm,\ \nexists ar \in OB_S(C_{AR}, d), [C_o, admin] \in d\_Perms(ar, d)$.
- *The sets of local objects in the state of the domains are disjoint, that is:* $\bigcap_{d \in D} OB_x(C_o, d) = \emptyset$ *for* $C_o \in C - Cm$ *where* OB_x *stands for* OB_A *and* OB_S

4.5 Constraints on states

Administrative operations determine a change of state. Operations can lead, however, to an improper state. For example, if a role schema is removed from a domain while instances of that schema still exist, then the enforcement of those roles would be impossible or fail in any case. In such case we say that the state of the domain is *unsound*.

It may also occur that after an operation the domain is in a transitory state. For example, if in a domain there is a unique admin role which is authorized to administrate regular roles and that admin role is removed, then the regular roles which still exist cannot be administered any longer until a new administrator is appointed. In that case, we say that the state of the domain is *administratively incomplete*. The notions of domain soundness and administration completeness are important to enable the check of domains and the proper specification of administrative operations. We now define more precisely the meaning of these concepts.

Auxiliary functions: $Owner(d)$ returns the admin role which is owner of d; $Parent(d)$ returns the domain of the owner role; *Sk(r)* returns the schema of role r.

Proposition 2 (Soundness of domains) *Let $\mathcal{I}^* = < \mathcal{I}, AH >$ be a policy state with $AH = (T, \prec)$. We say that $\mathcal{I}^*$ is sound iff for each domain $d \in D - \{TopDomain\}$ the following conditions hold:*

- *Regular and admin roles in d have an extent contained in the reference space of d.*
- *Each regular role is instance of a regular role schema in d, that is, $\forall d \in D, \forall r \in OB_S(C_R, d), \exists rs \in OB_S(C_{RS}, d),\ rs = Sk(r)$.*
- *Each admin role is instance of an admin role schema in the parent domain, that is, $\forall d \in D, \forall ar \in OB_S(C_{AR}, d), \exists ars \in OB_S(C_{ARS}, Parent(d)),\ ars = Sk(ar)$.*
- *d is present in some node of the Admin Hierarchy, that is: $\exists r \in OB_S(C_{AR}, d) \bigcup \perp, [d, r] \in T$.*
- *The owner role of d has the delegation permission, that is: $(Owner(d), [C_{AR}, delegate]) \in d\_Perm(Owner(d), Parent(d))$.*
- *The reference space of d is spatially contained in the reference space of the parent domain.*
- *Permissions assigned to an admin role ar in d are included in the set of permissions assigned to the owner of d, that is: $\forall p \in d\_Perm(ar, d) \rightarrow p \in d\_Perm(Owner(d), Parent(d))$.*

If $\mathcal{I}^*$ is sound, it is easy to show that the following properties hold.

- Let d_1 and d_2 be two domains. If d_1 and d_2 are not comparable, in that one domain is not the ancestor of the other, then the respective reference spaces $d_1.s$ and $d_2.s$ are spatially disjoint or overlap.
- The value of the admin role in node $[d, ar] \in T$ may be undefined, i.e. $\perp$ only if the node is a leaf in AH.

Proposition 3 (Administration completeness of domains) *Let $\mathcal{I}^* =< \mathcal{I}, AH >$ be a sound policy state. $\mathcal{I}^*$ is administratively complete iff for each domain $d \in D - \{TopDomain\}$ the following conditions hold:*

- *Each admin role is assigned at least one user, that is $\forall ar \in OB_S(C_{AR}, d), \exists au \in OB_S(C_U, d)$ such that $(ar, au) \in SUA(d)$.*
- *The owner role is assigned at least one user that is: $\exists u \in OB_S(C_U, Parent(d)),\ (u, Owner(d)) \in SUA(Parent(d))$.*
- *If there is at least one regular role instance, regular schema or user, there must be at least one admin role $ar \in OB_S(C_{AR}, d)$ which is authorized to administer regular roles instances, schemas and users.*
- *For each application object $o \in OB_A(C_{AP_i}, d)$ there must be an admin role with permission $[C_{AP_i}, admin]$.*

It can be easily shown that if the above conditions are satisfied then each object in the policy state is administered by someone.

Admin object class	meaning
$CreateObj(c, o, d)$	Create an object o of class c
$CreateDomain(d, s)$	Create a domain d with reference space s as sub-domain of current domain
$GrantAdmPrm(c, r, p, d)$	Assign permission p to admin object r of class $c \in \{C_{AR}, C_{ARS}\}$ in d
$GrantUser(u, r, d)$	Assign user u to role r

Table 3. Administration operations

4.6 Administration operations

In this section, for the sake of space, we only describe the process of delegating administration to a sub-domain using the operations of *CreateDomain, CreateObj, GrantAdmPrm, GrantUser*, reported in Table 3.

Assume that the member of an admin role ar in the current domain cd wants to create a sub-domain sd. 1) As first step ar creates a sub-domain through the operation *Create-Domain*. A child of node $[ar, cd]$ which is only partially filled is created and added to the admin hierarchy. 2) If not already existing, ar creates an admin schema ars' in the current domain and possibly assigns permissions to that schema through the operation *GrantAdmPrm*. 3) ar creates an admin role ar' as instance of schema ars' through the operation *CreateObj* and possibly assigns permissions through *GrantAdmPrm*: ar' is thus automatically added to the set of admin roles of sub-domain sd. The admin hierarchy is updated. 4) Finally ar creates an user au' through *CreateObj* and assigns it to the admin role ar' through *GrantUser*.

Notice that if the delegation permission is assigned to the admin role of the newly created sub-domain, the administration can be further delegated to nested sub-domains. Further, observe that the creator of a sub-domain can also add new admin roles after the domain is created.

5 Conclusions

In this paper we have presented a model for the decentralized administration of GEO-RBAC based on a hierarchy of spatial domains. The approach is quite flexible. Moreover, although nested domains can be freely created, the administration still remains indirectly under the control of the Chief Security Officer, which is important in contexts demanding strong access control. A first prototype of the GEO-RBAC system has been developed as Web application in which spatial-aware policies are stored in a Spatial DBMS. Some important issues are still open and will be investigated as part of the future activity:

- **User sharing.** In GEO-RBAC Admin, a users u who is registered in domain d is not visible in a sub-domain d'. The question is thus how to enable a more flexible management of users, so to allow a sort of automatic registration in sub-domains. To that purpose, a possible approach is to *share users* between domains.
- **Administrative object sharing.** An extension of the previous functionality is to enable the sharing of additional administrative objects, in particular role schemas. That would allow the specification of a role ontology for the organization.
- **Mobility.** The issue is how to support mobility of users across domains, that is, (*roaming*). As a scenario imagine an access control system for regulating the access to services, like traffic information services in different cities, and assume that a domain and thus a local policy is specified for each city. Assume also that the user is assigned a home domain. The user however is mobile, thus while driving the user may move outside the extent of the home domain and enter a new domain and this in/out operation can occur several times before the car definitely stops. This scenario raises a number of important requirements, such as the need of transparent connection to domains and roles interoperability.

Acknowledgements

This work has been partially funded by the European Commission project IST-6FP-014915 "GeoPKDD: Geographic Privacy-aware Knowledge Discovery and Delivery (GeoPKDD)" (web site: http://www.geopkdd.eu), and by the US National Science Foundation grant 0712846 "IPS: Security Services for Healthcare Applications".

References

[1] C. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati. Supporting Location-based Conditions in Access Control Policies. In *Proc. of the 2006 ACM Symp. on Information, Computer and Comm. Security*, 2006.

[2] R. Bhatti, J. B. D. Joshi, E. Bertino, and A. Ghafoor. X-GTRBAC Admin: A Decentralized Administration Model for Enterprise-Wide Access Control. *ACM Trans. Inf. Syst. Secur.*, 4, 2005.

[3] J. Crampton and G. Loizou. Administrative scope: A foundation for role-based administrative models. *ACM Trans. Inf. Syst. Secur.*, 6(2):201–231, 2003.

[4] M. Damiani, E. Bertino, B. Catania, and P. Perlasca. GEO-RBAC: A Spatially Aware RBAC. *ACM Trans. Inf. Syst. Secur.*, 10(1), 2007.

[5] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[6] P. P. Griffiths and B. W. Wade. An authorization mechanism for a relational database system. *ACM Trans. Database Syst.*, 1(3):242–255, 1976.

[7] A. Kern, A. Schaad, and J. Moffet. An Adminstration Concept for the Enterprise Role-based Access Control Model. In *Proc. of the 8th ACM Symposium on Access Control Models and Technologies*, 2003.

[8] M. Kumar and R. Newman. STRBAC - An approach towards spatio-temporal role-based access control. In *Proc. of Communication, Network, and Information Security*, 2006.

[9] N. Li and Z. Mao. Administration in role-based access control. In *ASIACCS '07: Proc. of the 2nd ACM symposium on Information, computer and communications security*, USA, 2007.

[10] S. Oh, R. Sandhu, and X. Zhang. An effective role administration model using organization structure. *ACM Trans. Inf. Syst. Secur.*, 9(2):113–137, 2006.

[11] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Trans. Inf. Syst. Secur.*, 2(1):105–135, 1999.

[12] A. Schaad, J. Moffett, and J. Jacob. The role-based access control system of a European bank: a case study and discussion. In *Proc. of the 6th ACM symposium on Access Control Models and Technologies*, USA, 2001.