An Authorization Model for Federated Systems

Sabrina De Capitani di Vimercati Pierangela Samarati

Dipartimento di Scienze dell'Informazione
Università di Milano
via Comelico 39/41 Milano 20135, Italy
Phone: (+39) 2-55006227
Fax: (+39) 2-55006253
{*decapita,samarati*}*@dsi.unimi.it*

Abstract. We present an authorization model for federated systems based on a tightly coupled architecture. The model supports authorizations to build and maintain the federation as well as authorizations to access the federated data. At each component site owners declare the objects they wish to export and the access modes executable on them by users of the federation. Inclusion of objects into the federation requires their subsequent import by the federation administrator. Different degrees of authorization autonomy are supported, whereby users can retain or delegate the federation administrator the task of specifying authorizations. A site can require to authenticate the user at each access or accept his identity as communicated by the federation. An access control algorithm describing controls to be enforced at the federation and at each local site under the different authentication and administrative options is presented.

Keywords: Federated systems, access control, authorization administration, authorization autonomy.

1 Introduction

A federated system is a collection of cooperating but autonomous component systems [17]. Autonomy of a component means that the local administrator maintains some form of control over its system. Four different kinds of autonomy can be supported: *design autonomy*, *communication autonomy*, *execution autonomy*, and *association autonomy*.

Enforcing protection of information in federated systems requires the investigation of several issues which traditional access control models, proposed for centralized or distributed systems [3, 15], do not address. A major issue that becomes very important in this context is the establishment of administrative policies regulating grant and revocation of authorizations on federated data. While in a centralized or distributed system ownership or centralized administration may be satisfactory solutions, federated systems call for more flexible approaches. On one side, enforcing ownership would require data owner to specify authorizations for federated users and therefore to maintain information of

how the federation is composed. On the other side, applying a centralized administration approach may imply a loss of control, and therefore of autonomy of the data owner. Moreover, even traditional problems, such as authentication or user identities, require careful reconsideration in a federated context, where single components may specify accesses for federated subjects (users, groups, or whole sites).

Although recent research has addressed the problem of protecting federated systems [2, 7, 8, 12, 14, 20, 21, 22] and few federated systems, like Mermaid [18], Orion-2 [10], or the one proposed by Heimbigner and McLeod [5] support some form of authorization specification and access control, several issues still remain to be investigated. In this paper we investigate these issues and focus on two particular subtypes of association autonomy, meaning *authentication* and *authorization* autonomy. By authentication autonomy we mean the ability of a site to decide how the identity of each user accessing data stored at the site is established. By authorization autonomy we mean the ability of a site to specify which accesses are to be allowed or denied on objects stored at the site.[1]

We propose an authorization model for the protection of information in a tightly coupled federated system. The reason for considering a tightly coupled federation is to avoid the users the burden of explicitly maintaining the relationships with each single site participating in the federation. A tightly coupled architecture allows to rely on a central authority for defining and maintaining the federation. Even with this centralized administration we allow a good degree of autonomy to each single site participating in the federation. In particular, our model allows users to selectively share their objects by specifying, for each object, the access modes that are available to the federation. Different kinds of administrative policies are also supported whereby users can retain the privilege of specifying authorizations on objects, can delegate the federation administrator the task of specifying authorization, or can require that accesses be authorized by both the federation administrator and by them in order to be allowed. Access authorizations are specified at both the global level (on federated objects) and the local level (on exported objects). Local authorizations can also be negative. The possibility of specifying negative authorizations at the local level allows owners to retain control over their objects even when delegating administration to the federation administrator. Although in the paper we do not deal with authentication issues, the model assumes that two different authentication options are possible. A site can require to authenticate every user requiring access (even if the request comes through the federation) or can accept the identity of the user as communicated by the federation.

The remainder of the paper is organized as follows. Section 2 illustrates previous work on authorization models in federated systems. Section 3 presents the reference architecture of the federation and introducing some notations. Sec-

[1] The need to explicitly consider security issues in a separate category has been also pointed out in [8] where the notion of authorization autonomy is introduced. Note, however, that the notion introduced in [8] corresponds to our notion of authentication autonomy.

tion 4 illustrates authorizations and operations for constructing the federated data. Section 5 describes how authorizations can be specified at both the federation and the local level. Section 6 illustrates access controls enforced, globally and locally, upon submissions of access requests to the federation. Finally, Section 7 presents the conclusions and outlines future work.

2 Related work

Some federated database systems proposed in the literature present some form, often very limited, of authorization specification and access control.

Wang and Spooner [22] propose an approach to enforce content-dependent access control in a heterogeneous federated system where authorizations can be specified at both the local and the global level. The approach is based on views and ownership paradigm. Content-dependent access control is enforced by materializing views and treating them as protection objects.

In Mermaid [18], a front-end system to integrate multiple homogeneous DBMSs, an authorization model enforcing access control at both global and local level is presented. In order to use Mermaid a user must be registered for it. Access authorizations are specified both at the global level, in the Mermaid system, and at the local level, at each site. Access control at a site is always carried out with respect to the identity of the user at the site.

Another model allowing specification of authorizations at both the local and global level is proposed by Jonscher and Dittrich in [8]. In this model a global security administrator specifies the local identities corresponding to each global identifier. Authorizations can be positive or negative. The grantor of an authorization at the global level can require consistency of the authorizations. Consistency means that a request permitted according to the global authorizations cannot fail due to access rejection at the local level. Consistency is enforced by propagation of authorizations: every time a global authorization is granted, local sites are required to grant the corresponding necessary authorizations. The global authorization is inserted only if all the corresponding local grants can be enforced.

In the work of Heimbigner and McLeod [5] a loosely coupled federated architecture supporting a very primitive form of access control is presented. The data model is based on the concept of types and maps. Types are collections of objects that share common properties while maps are functions that map objects of some types into object instances. Authorizations on types and maps defined at a site are specified in terms of whole components and not to specific users identity. All types and maps defined at a given site s_1 for which another site s_2 is authorized constitute the export schema of s_1 for s_2. A site can import all types and maps contained in export schemas defined for it at other sites. Export and import operations are the result of negotiation between the two sites. Once types and maps are imported at a site no further negotiation is required for their access, which is completely analogous to access to a local type except that data is transferred over the network.

In ORION-2 [10], a federated loosely coupled object-oriented system is presented based on a shared database and a number of private databases. The shared database contains data accessible to all authorized users, while each private database can be accessed only by the user who owns it. Check in and check out operations allow users to get (copy) objects from the shared databse and returning them back. Updates to different copies checked out, which may be checked in again at a later time, are managed through the use of versions.

Blaustein et al. [2] propose an approach to control access in federated database systems based on agreements established among the different sites of the federation. Agreements are rules regulating the access by users at the different sites to the cooperating database systems. Two kinds of agreements are considered: action agreements and access agreements. Action agreements describe the action to be taken in response to database requests, while access agreements allow to enforce exceptions to prohibitions otherwise in effect. The identity of users at the remote site from which they submit the request is used in access control.

Other proposals have been presented based on the enforcement of mandatory policies [12, 14, 20, 21].

3 System's Architecture and Basic Concepts

In this section we illustrate the architecture of the federated system and give the basic assumptions on our model.

The system is essentially structured on two levels: at the global level there is the federation and at the local level the different sites participating in it. At the federation level a privileged user, called *federation administrator* is responsible for creating and maintaining the federated schema and establishing authorizations for users to access the objects in the federation.

At the local level there are the different sites taking part in the federation. We distinguish three categories of participants:

- *customers*. These are sites whose users can be authorized to connect to the federation and access its objects.
- *providers*. These are sites that take part in the construction of the federated data, i.e., that can export their local objects for the population of the federation.
- *customers-providers*. These are sites that fall in both the categories above.

At each site a privileged user, called *site administrator* is responsible for the relationship with the federations in which the site participates.[2] Registration of a site in a federation is the result of a negotiation between the administrator of the

[2] For the sake of simplicity we consider a single site administrator to be responsible for the relationship with all the federations in which a site can participate. The approach can be easily extended to the case where several administrators, each responsible for the relationship with a specific federation, exist at a site.

site and the administrator of the federation. In the following, given a federation f notation *providers(f)* and *customers(f)* indicate the sites registered as provider and customer of f, respectively. Sites in the third category belong to both sets.

A good user's authentication is a prerequisite for a successful access control. In this paper we do not deal with the problem of authenticating the users but assume different authentication options we allow the federation to support. Each site registered as a provider of a federation can, during the negotiation phase, choose between two different authentication policies according to which identity of federated users needing to access local objects through the federation can be established: *global* and *local* authentication. In the *global* authentication, federation's users do not need to identify themselves at the site, their identity as communicated to the site by the federation will be used for access control (of course the federation will have to authenticate itself). In the *local* authentication, before processing each access request coming by a user through the federation the participating site will require the user to identify himself at the site. This identity will be used in the access control process. Note that communication of identities between federation and sites requires some form of trust in both the federation/site that enforced the authentication and in the communication system. Different certification forms can be used to provide this, such as those illustrated in [1, 13, 23].

3.1 Subjects and Objects

At each site a set of local users is assumed and a set of local objects is stored. We do not make any assumption on the data model used at the federation or at each specific site. Given a site s, we will simply refer to the objects stored in it as O_s. Moreover, given an object $o \in O_s$, we denote with M_o the set of access modes executable on it. The specific data types and access modes applicable on the objects will depend on the data model considered. For instance, in a relational model, M_o will be composed of the elementary select, insert, and update operations plus all applications defined for a relation. In an object-oriented model, M_o will be composed of all methods, elementary and non, executable on object o. To make our model independent from the specific administrative policy applied at the local sites, we assume each object o is associated with a set of administrators. The set of administrators associated with an object o will contain: the object's owner, if ownership is applied; the system's administrator, if a centralized policy is applied; and all users owning an administrative authorization on the object if decentralized administration is applied.

At the federation site two kinds of subjects are considered: users and groups. Users are entities allowed to connect to the federation and submit requests on its data. Groups, which are sets of users, can be defined with reference to the users identities at the federation or at the site from which their connection originates (for instance a group can be defined as containing all users connecting to the federation from a specific site). Each user can belong to one or more groups.

At the federation level, three kinds of objects are supported:

- *Global* objects. These are objects created directly in the federation by the federation administrator or by users explicitly authorized for that.
- *Imported* objects. These are objects defined and stored at some site taking part in the federation and imported "as-is" by the federation.
- *Composite* objects. These are objects defined as aggregation of other global or imported objects.

3.2 Notations

Before proceeding further to illustrate our model we introduce some notations that will be used in the paper. In the following, we will use letters U, G, O, and A to denote respectively a set of users, groups, objects, and authorizations. Subscripts will be used to discriminate the site or federation to which they refer. For instance, O_s denotes the set of objects at site s and G_f denotes the set of user groups defined at federation f.

To take distribution into consideration we suppose each user/group identifier to have associated the site at which it has been defined. User identifiers at site s will therefore have the form **user**@s. Analogously, subject (i.e., user or group) identifiers at a federation f will have the form **subject**@f. We will omit the specification of the site when it is not needed in the explanation or it is clear from the text. For instance, the site specification in subjects of authorizations will be omitted when the site is the same as the one at which the authorization is stored.

In authorizations we will also allow subject patterns to be specified. Subject patterns can be specified by using the wild character in place of a specific identifier. We allow two kinds of pattern. Pattern "*@s", which indicates any identifier at site s, and pattern "*", which indicates any identifier at any site. In the following, notation U_s will be used to indicate the set of all the identifiers of the form **user**@s together with *@s. Moreover, given a collection $s_1, \ldots, s_n$ of sites participating as customers in a federation f, we will use notation $U_{customers(f)}$ to indicate the set $U_{s_1} \cup \ldots \cup U_{s_n} \cup \{*\}$. A pattern satisfies all identifiers to which it can be reduced by appropriately instantiating the wild character. In other words, a patterns p *covers* an identifier **u@s**, denoted as $p \rhd$ **u@s**, if either *i)* $p =$ **u@s**, *ii)* $p =$ *@**s**, or *iii)* $p = *$.

Finally, given an object o, we will use notation *site(o)* to denote the site at which the object is stored.

3.3 Working of the System

Users of sites registered as customer of a federation and wishing to access the federation must explicitly open a working session by connecting to the federation site (see Figure 1). Connection requires identification of the user (1) and corresponding authentication of this identity by the federation. This identity will be used by the federation for enforcing access control. Note that this assumption does not rule out the possibility of anonymous connection. Anonymous connection may be treated with a special user identifier called **anonymous**. Besides the

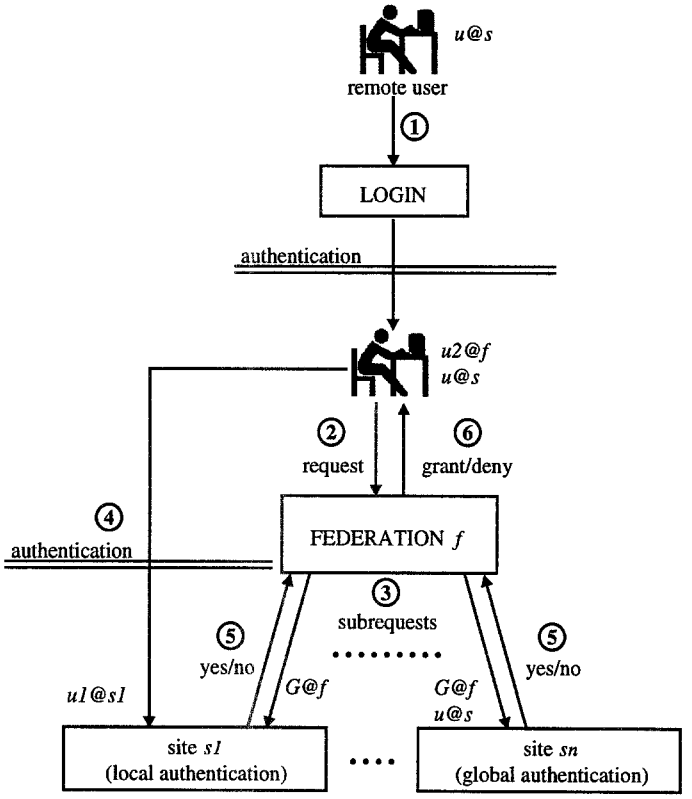

Fig. 1. Working of the system

identity with which he connected to the federation, a user has also associated an attribute containing his identity at the local site of origin.[3] Once the user has been authenticated by the federation he can submit requests to access the federated objects (2). Accessing a federated object may require to access the corresponding underlying local objects at the different sites. Each user request must therefore be split, for both access control and data retrieval in a set of requests on the underlying local objects (or a subset of them). Then, the federation sends each site storing a local object involved in the transaction an access request for the groups to which the user belongs together with the remote identity of the user (3). In case of local authentication the user will need to re-authenticate himself at the local site (4). Then, each local site controls whether the access

[3] Note that to allow non disclosure of identities, the case of attribute only partially specified, i.e., with indication only of the site, may be considered.

request is allowed and returns the result to the federation (5), which will take the access decision (6) on the basis of the replies of the different sites.

Access control will be illustrated in details in Section 6.

4 Population of the Federation

Populating a federation means specifying the objects part of the federated database. The federation can be populated directly, by explicitly creating global objects in the federation or indirectly, by importing objects from the component sites.

In the remainder of this section we discuss the authorizations required and the operations that must be executed to populate the federation.

4.1 Authorizations for the Population

Creation of global objects can be done by any user explicitly authorized for that. Authorizations allowing users to create objects in the federation, specified by the federation administrator, have the following form.

Definition 1 Create authorization. Let f be a federation, a create authorization $a \in A_f$ is a triple of the form $\langle$*fed-subject*,`create`,*remote-id*$\rangle$, with *fed-subject* $\in U_f \cup G_f$ and *remote-id* $\in U_{customers(f)}$.

Authorization $\langle$*fed-subject*,`create`,*remote-id*$\rangle$ states that federation's user *fed-subject* (its member in case *fed-subject* is a group), connected to the federation through some customer site with a login covered by *remote-id*, can create objects in the federation. For instance, authorization $\langle$`Tom`,`create`,$*\rangle \in A_{\mathtt{f}}$ states that federated user `Tom` can create objects in the federation `f`. Authorization $\langle$`senior-members`,`create`,`*@site1`$\rangle \in A_{\mathtt{f}}$ states that all users members of group `senior-members` and which are connected to the federation from `site1` can create objects in federation `f`.

Population of the federation by getting objects from sites participating in it requires agreement of both the local object's administrator on one side and the federation administrator on the other side. At the local site any of the object's administrators must be willing to export their objects to the federation. At the federation site, the federation administrator must be willing to import the objects in the federation.

We allow objects to be exported with reference to specific access modes. For instance, a user can decide to export one of his objects, i.e., allow access to the federation's users, only for reading and another one of his objects for both reading and writing. The set of objects stored at site s that can be exported to federation f, together with the specification of access modes and additional administrative information is called the export schema of the site for the federation, denoted as $ES_{s,f}$. In order to export their objects, i.e., to include them in a given export schema, users must be explicitly authorized for that by the site administrator. Export authorizations are defined as follows.

Definition 2 Export authorization. Let s and f be a site and a federation respectively with $s \in$ *providers(f)*. An export authorization $a \in A_s$ is a triple of the form $\langle$*loc-user*,`export`,$f\rangle$ with *loc-user* $\in U_s$.

Authorization $\langle$*loc-user*,`export`,$f\rangle$ states that user *loc-user* can export to federation f the objects he administers.

Users can also delegate the site administrator to export their objects, with reference to specific access modes. Reasons for delegation can be various. On one hand, users may not want to worry about federations in which the site participates and about authorizations for users of the federation. On the other hand, the administrator himself may wish to not allow direct export of objects by users thus retaining the control of what the site exports (a kind of centralized administration). Authorizations that allow site administrators to export objects are as follows.

Definition 3 Del_export authorization. Let s be a site. An export delegation authorization $a \in A_s$ is a 4-tuple $\langle$`lsa`,`del_export`,*loc-object*,*modes*$\rangle$ where `lsa` denotes the site administrator[4], *loc-object* $\in O_s$ is the object whose export is delegated, and *modes* $\subseteq M_{loc\text{-}object}$ is the set of access modes that can be exported.

Authorization $\langle$`lsa`,`del_export`,*loc-object*,*modes*$\rangle \in A_s$ states that the administrator of site s can export object *loc-object* with access modes *modes* in any federations for which the site is registered as provider.

For instance, authorization $\langle$`lsa`,`del_export`,`myfile`,{`read`}$\rangle$ granted by `Tom` at `site1` states that the local administrator of `site1` can make object `myfile` of `Tom` available for reading to all the federations in which the site participates.

4.2 Populating the Federation

In this section we discuss the operations to be executed in order to populate the federation with objects stored at the local sites.

When exporting an object, users must also decide the administrative policy establishing who can specify authorizations to access the object once imported in the federation. Three kinds of policies are supported:

- *site retained* - **SR** - Access authorizations can be specified only by the local administrators of the object;
- *federation controlled* - **FC** - The object is freely available to the federation. Access authorizations defined by the federation administrator establish who can access the object.
- *cooperative* - **C** - Access authorizations are granted by both the local administrator and the federation administrator.

[4] We note that, since the administrator is unique, its indication could have been omitted. However, to make the model easily extendable to the case of multiple administrators we prefer to explicitly indicate it.

The export operation, which can be executed by either any of the object's administrators or the site administrator, will be successful if the corresponding necessary authorization is present in A_s. Execution of the export operation for access modes *modes* on object *loc-object* with administrative policy *adm-policy* results in the addition of 4-tuple ⟨*loc-object*,*modes*,*adm-policy*,*exporter*⟩ in $ES_{s,f}$, where *exporter* is the user who required the operation. In the following, given an object *loc-object* and a federation f, notation *modes*(*loc-object*,f) denotes the set of access modes exported for *loc-object* to federation f.

In order for an object to be accessible to the federated users, the object must then be imported by the federation administrator. The import operation in a federation can be executed only by the federation administrator. For the operation to be successful, the object must be contained in the export schema of the site for the federation. Upon import, the object is included in the federated schema and can therefore be made available, according to the access modes specified, to the federated users.

When an object is imported in a federation, the specification of the access modes with which it can be accessed and the administrative policy are also registered at the federation. For the purpose of this paper we consider the federated schema as a collection of triples of the form ⟨*fed-object*, *modes*, *policy*⟩ where *modes* is the set of access modes that can be exercised on the object and *policy* is the administrative policy to which *fed-object* is subject. In case of imported objects, *modes* and *policy* are exactly those indicated in the export schema. In case of global objects, *policy* has value `global` (**G**) and *modes* is equal to all the access modes that can be exercised according to the object type. In case of composite objects, the set of access modes is the set of all the access modes defined by the administrator according to the object type. The administrative policy is the one specified for the component objects, if all the component objects have the same administrative policy. It has value `undefined` (**U**) otherwise. We assume that composite objects are always defined on objects present in the federated schema. This implies that an object must be imported before being used in the definition of a composite object. In the following, given an imported object o we will denote with *loc*(o) the corresponding local object.

4.3 Object's Withdrawal and Revocation of Authorizations for the Population

Objects exported by a site can be withdrawn. A user/administrator can withdraw only the objects he has exported. The withdrawal operation of an object from a federation has the effect of removing the object from the corresponding export schema. If the object had been imported in the federation, the withdraw operation will have effect on the federated schema, where it should also be deleted. Different strategies can be used for this, such as immediate propagation, periodic propagation, or propagation at access time. In the first strategy, whenever an object is withdrawn from an export schema a withdraw request is sent to the federation. In the second strategy, export schema changes are periodically sent to the federation. In the third strategy, no propagation of export schema

changes is sent by the site to the federation. The fact that an object is not available anymore for the federation will be find out at the time the federation will try to access it. The specific strategy adopted depends on the underlying data model and is outside the scope of this paper.

We notice that the need may arise to temporarily suspend access to an object by users of the federation. If the only possibility of enforcing this were either revoking access authorizations or withdrawing the object, the need of re-specifying the authorizations and/or re-exporting the object would arise. To avoid this destructive effect, we allow users to temporarily isolate their objects from the federation. The *isolate* operation results in a temporarily suspension of accesses to the object through the federation but it does not affect in any way the export schema, the federated schema, or the authorizations specified on the objects.

During the life time of a federation, authorizations for exporting objects may change, i.e., new authorizations be granted and existing authorizations revoked. Revocation introduces the problem of dealing with objects exported thank you to the authorizations being revoked. Two different strategies, one destructive (i.e., with object withdrawal) and one more conservative (i.e., without object withdrawal), can be adopted with respect to this. In the destructive strategy, whenever a user is revoked the export authorization for a federation, all the objects exported by the user into that federation are removed from the export schema. Analogously, if the site administrator is revoked the del-export privilege on an object, the object will be removed from the export schemas in which it was included by the site administrator. In the conservative strategy, upon revocation of a export/del-export authorization, only this authorization is deleted and the export schema remains unchanged.

5 Access Authorizations

In our model we allow the specification of authorizations at both the global level, on the federated objects, and at the local level, on the objects stored at each site.

Authorizations at the global level can be specified only by the federation administrator in case of imported or composite objects. They can be specified by the federation administrator as well as by the creator in case of global objects.

Authorizations on federated objects can be granted to single users as well as to groups. Authorizations can also contain restrictions on the remote user's identity or on the remote site from which the user is connected. Subjects and remote identities can be specified completely by indicating user/group identifier and site, or by using subject patterns. Authorizations at the global level are defined as follows.

Definition 4 Global access authorization. Let f be a federation, an access authorization $a \in A_f$ is a 4-tuple $\langle$*fed-subject*,*mode*,*fed-object*,*remote-id*$\rangle$ where: *fed-subject* $\in U_f \cup G_f$, *mode* $\in M_{\textit{fed-object}}$, *fed-object* $\in O_f$, *remote-id* $\in U_{customers(f)}$.

Authorization ⟨*fed-subject*,*mode*,*fed-object*,*remote-id*⟩ states that *fed-subject* (any of its members in case it is a group), connected to the federation through some customer site with a login covered by *remote-id*, can exercise privilege *mode* on *fed-object*. For instance, authorization ⟨`Tom`,`read`,`object1`,*⟩ states that federated user `Tom` can `read` `object1`. Authorization ⟨*,`read`,`object2`,*`@site1`⟩ states that all users connected from `site1` can `read` `object2`.

At the local level, users can specify access authorizations over the objects they have exported. Subjects of authorizations at the local level are groups defined by the federation. The reason why groups are considered for authorizations at the local level is to avoid requiring each site to know about single users of the federation. Indeed, although it is reasonable to assume some knowledge of each site about global identifiers, it is quite improbable that each site can retain track of identifiers of each single user of the federation. The consideration of groups overcome this burden. Each federation declares a set of groups into which federated users are organized and for which users at local sites can specify authorizations. Changes to groups (i.e., addition or removal of members) will not therefore have any effect on authorizations specified at local sites.

Authorizations at the local level can also put constraints on the specific identity of the user requiring the access. In case of global authentication, the constraints will refer to the remote user's identity at the site from which the user connected to the federation. In case of local authentication, the constraints will refer to local identity of the user as authenticated by the site.

Access authorizations at local level can be positive or negative. Positive authorizations state accesses that must be granted, whereas negative authorizations state accesses that must be denied. Possible conflicts are solved according to the denials take precedence policy. Hence, if both a positive and a negative authorization applies for an access, the access is denied.

Local authorizations are defined as follows.

Definition 5 Local access authorization. Let s be a site in a federation f, an access authorization $a \in A_s$ is a 5-tuple ⟨*fed-group*,*mode*,*sign*,*loc-object*,*id*⟩ with *fed-group* $\in G_f$, *mode* $\in$ *modes(loc-object,f)*, *sign* $\in \{+,-\}$, *id* $\in U_s \cup U_{customers(f)}$.

Authorization ⟨*fed-group*,*mode*,*sign*,*loc-object*,*id*⟩ states that all members of *fed-group*, with a login (remote if authentication is global, and local otherwise) covered by *id*, can (cannot if *sign* $= -$) exercise privilege *mode* on *loc-object*.

For instance, authorization ⟨`senior-members@fed3`,`read`,+,`object1`,*⟩ $\in A_{\mathtt{s}}$ states that group `senior-members` of federation `fed3` can read `object1` stored at site `s`.

Authorization ⟨`senior-members@fed3`,`read`,+,`object2`,*`@site2`⟩ $\in A_{\mathtt{s}}$ states that the members of group `senior-members` who connected to federation `fed3` from `site2` can `read` `object2` stored at site `s`.

Authorization ⟨*,`read`,−,`object1`,`bob`⟩ $\in A_{\mathtt{s}}$ states that user `bob@s` cannot read `object1` stored at site `s`.

The reason for considering both positive and negative authorizations at the local level is to give exporters a means of retaining control on accesses allowed

on their objects. This characteristic is very important for two main reasons. First, authorizations are specified for groups of users. However, an exporter may wish to grant a whole group an access but at the same time make sure that a specific user will not be able to exercise it. Since user's groups are defined at the federation site, and therefore the exporter has no means of controlling their configuration (for instance by excluding the specific user), the specification of negative authorizations may be the only means to express this requirement. Second, in the case of federation controlled administration, the exporter delegates the federation administrator the task of specifying access authorizations on his object once imported. This means that federated users will not need to have local privileges in order to access the object. However, by specifying negative authorizations, the exporter can restrict the accesses that can be exercised on the object. As a matter of fact no access, even if authorized by the administrator, will be allowed if a negative authorization for it exists at the local level.

Subject patterns, together with positive and negative authorizations can be used to express different protection requirements, as illustrated by the following example.

Example 1. Consider a site $\mathtt{s}_1$ enforcing local authentication and consider a user who exports an object o'_1 to federation `f`. Suppose the object is exported for the write access mode and under the **`site retained`** administrative policy. Suppose now that the user wishes to grant the write access to group **`student`** of the federation. However, he does not want local user **`jimmy`** to read the object, even in the case where **`jimmy`** is a member of **`student`**. This requirement can be expressed by specifying the following two local authorizations: $\langle$**`student@f`**,**`read`**,+,o'_1,$*\rangle$ and $\langle *$,**`read`**,−,o'_1,**`jimmy`**$\rangle$.

As another example consider a site $\mathtt{s}_2$ enforcing global authentication. Suppose a user exports an object o'_2 to federation `f`. Suppose the object is exported for the read access mode and under the **`federation controlled`** policy. Since the policy is federation controlled, in order to access the object, users need to have the authorization for the access at the global level. By contrast, no authorization is necessary at the local level. Suppose now that the exporter wants to make sure that users from site `s1` will not read the object. He can do so by specifying the following negative authorization: $\langle *$,**`read`**,−,o'_2,$*@\mathtt{s}_1\rangle$. □

6 Access Control

Users connected to the federation can submit requests to access federated objects. In order to determine whether to grant or deny the access, authorizations specified at the federation as well as at the local sites involved in the access must be controlled. Specific controls and additional authentication processes required depend on: the type of object (global versus imported or composite), the kind of administrative policy of the component object(s), and the authentication policy required by each site involved. Each request on an imported or composite object is translated into a request, or a set of requests, on the corresponding

/* **Input:** Request (u, m, o, rid) of federated user u, remotely connected as *rid*, to exercise access mode m on federated object o */

At the federation site f

1. Determine the groups G_u to which the user belongs.
2. /* Access control at global level*/
   If *adm-policy*$(o) = \mathbf{G}$
   then if $\exists a \in A_f$ such that $a \rhd \langle u, m, o, rid\rangle$ or $a \rhd \langle G_u, m, o, rid\rangle$
   then *grant access*
   else *deny access*
   else if *adm-policy*$(o) \neq \mathbf{SR}$
   then if $\nexists a \in A_f, a \rhd \langle u, m, o, rid\rangle$ or $a \rhd \langle G_u, m, o, rid\rangle$
   then *deny access*
3. /* Split into requests on component objects */
   Decompose access (m, o) into a set of accesses $S = \{(m_1, o_1), \ldots, (m_n, o_n)\}$.
4. /* Check locally all accesses for which component object is global */
   If there exists $(m_i, o_i) \in S$ such that *adm-policy*$(o_i) = \mathbf{G}$ and $\nexists a \in A_f$,
   $a \rhd \langle u, m, o, rid\rangle$ or $a \rhd \langle G_u, m, o, rid\rangle$
   then *deny access*
   else if there do not exists $(m_i, o_i) \in S$ such that *adm-policy*$(o_i) \neq \mathbf{G}$
   then *grant access*
5. /* Send requests on imported objects to local */
   For each pair $(m_i, o_i) \in S$ such that *adm-policy*$(o_i) \neq \mathbf{G}$ **do**
   $o_i' = loc(o_i)$
   send request $r_i = (G_u, m_i, o_i', rid)$ to the site $site(o_i')$ from which o_i' was imported
6. /* Collect replies from site and and return response to the user */
   Collect replies from all sites
   If the reply has value **true** for all requests r_i **then** *grant access* **else** *deny access*

At each local site $s = site(o_i')$
/* upon reception of request $r_i = (G_u, m_i, o_i', rid)$ from federation f */

1. **if** $\exists\langle o_i', modes, policy\rangle \in E_{s,f}$ with $m_i \in modes$
   then *adm-policy* := *policy*
   else *reply* := **false**, go to step 4
2. **if** authentication is global
   then $id = rid$
   else Authenticate user. Assign *id* the local user identity.
3. **case** *adm-policy* **of**
   SR or **C**: **if** $\exists a \in A_s, a \rhd \langle G_u, m_i, +, o_i', id\rangle$ and $\nexists a \in A_s, a \rhd \langle G_u, m_i, -, o_i', id\rangle$
   then *reply* := **true**
   else *reply* := **false**
   FC: **if** $\nexists a \in A_s, a \rhd \langle G_u, m_i, -, o_i', id\rangle$
   then *reply* := **true**
   else *reply* := **false**
4. Send *reply* to f

Fig. 2. Access control algorithm

local object(s). Each of these requests must be communicated to the appropriate site for both access control, since local authorizations must be verified to determine whether access must be allowed or denied, and data retrieval, since the data stored at the different sites are not replicated at the federation. The mapping of access operations on federated objects onto access operations on the corresponding local objects is enforced by the data management system of the federation. It is outside of the scope of this paper to discuss this mapping. We therefore assume that each operation to access a federated object o with access mode m is mapped by the data management system onto a set of operations $\langle m_1, o_1\rangle, \ldots, \langle m_k, o_k\rangle$, where $o_1, \ldots, o_k$ are local objects corresponding to o and $m_1, \ldots, m_k$ are the access modes to be exercised on these objects.

In the following, we say that a global authorization $a = \langle$*fed-subject*,*mode*,*fed-object*,*remote-id*$\rangle$ covers a tuple $t = \langle u,m,o,id\rangle$, written $a \rhd t$ iff: *fed-subject* $\rhd$ u, *mode* $= m$, *fed-object* $= o$, and *remote-id* $\rhd$ *id*. The cover relationship for local authorizations requires in addition, the sign in the authorization and in the tuple to be the same.

Figure 2 illustrates the different controls to be executed at the federation site and at each local site involved. In the figure, given a group G_r, we use notation $\langle G_r,m,o,rid\rangle$ as an abbreviation for the set $\{\langle g_i,m,o,rid\rangle \mid g_i \in G_r\}$. Moreover we use $a \rhd \langle G_r,m,o,rid\rangle$ as an abbreviation for $a \rhd \langle g_1,m,o,rid\rangle \vee a \rhd \langle g_2,m,o,rid\rangle \vee \ldots \vee a \rhd \langle g_k,m,o,rid\rangle$. Let f be a federation and u be a federated user connected to the federation with some remote login *rid*. Consider the request submitted to the federation by this user to exercise access mode m on object o. In step 1, the set of groups G_u to which the user belongs are determined. In step 2, access control on global authorizations is enforced.

If the object is global (administrative policy **G**) this is the only control that must be executed. Then, access is granted or authorized according to whether an authorization covering the request exists or not. Otherwise, the controls to be executed depend on the administrative policy to which the object is subject. In particular, if the administrative policy is site retained, no authorization is needed for the access at the global level and therefore no authorization check is performed. In any other case (i.e., administrative policy equal to **FC**, **C**, or **U**), the system looks for an authorization covering the request. If no such authorization exists the access is denied, otherwise the process continues. Hence, the access on the federated object is split into the corresponding accesses on component objects. (Note that access splitting is meaningful only for accesses on composite objects, it has not effect otherwise). As a result a set of accesses to be controlled is obtained. If any of these accesses is on a global object and no authorization exists for it, the request is denied. Then, if no access to local objects is required the request is granted. Otherwise, the algorithm proceeds by sending, for each access on an imported object, a request to the corresponding site for the set of subjects G_u with remote identity *rid*. The access is granted if all local sites accept the request, it is denied otherwise.

Access control at each local site s works as follows. Upon reception of a request by a federation, the export schema of the site for the federation is con-

Constraints on authorizations		Administrative policy: Site retained	Federation Controlled	Cooperative	Global
At federation f (in A_f)			$\exists a: a \triangleright (u,m,o,rid)$ or $a \triangleright (G_u,m,o,rid)$	$\exists a: a \triangleright (u,m,o,rid)$ or $a \triangleright (G_u,m,o,rid)$	$\exists a: a \triangleright (u,m,o,rid)$ or $a \triangleright (G_u,m,o,rid)$
At site s of o (in A_s)	global auth.	$\exists a: a \triangleright (G_u,m,+,o',rid)$ $\nexists a: a \triangleright (G_u,m,-,o',rid)$	$\nexists a: a \triangleright (G_u,m,-,o',rid)$	$\exists a: a \triangleright (G_u,m,+,o',rid)$ $\nexists a: a \triangleright (G_u,m,-,o',rid)$	
	local auth.	$\exists a: a \triangleright (G_u,m,+,o',lid)$ $\nexists a: a \triangleright (G_u,m,-,o',lid)$	$\nexists a: a \triangleright (G_u,m,-,o',lid)$	$\exists a: a \triangleright (G_u,m,+,o',lid)$ $\nexists a: a \triangleright (G_u,m,-,o',lid)$	

Legenda: $o' = loc(o)$
rid remote identity of user u
lid identity of user u at site s

Table 1. Conditions necessary to grant request by user u of groups G_u with remote location *rid* to exercise mode m on federated object o

trolled. If there is no triple in the export schema for the object with access modes including the one required, a false reply is immediately returned to the federation. Otherwise the control proceeds. The identity to be considered in the access control depends on the authentication policy established for the site. If the authentication is global, the remote identity of the user as communicated by the federation is considered. If the authentication is local, the user will be asked to identify himself. This local identity, once authenticated, will be used in place of the remote identifier in the access control. Hence the local authorizations are checked. In case of **site-retained** or **cooperative** administrative policy a **true** is returned only if there exists a positive authorization and there does not exist any negative authorizations for the access. In case of **federation controlled** administrative policy a **true** is returned if no negative authorization exists for denying the access.

Table 1 summarizes the authorizations necessary at global and local level in order for a request on a global or imported object to be granted. In case of composite objects, at the local level the conditions must be satisfied, at the corresponding site, with reference to each access (m_i, o_i) in which the global operation is decomposed.

Example 2. Let $\mathtt{s}_1$, $\mathtt{s}_2$, $\mathtt{s}_3$ be three sites in a federation $\mathtt{f}$. Let the export schema and authorizations at sites $\mathtt{s}_1$ and $\mathtt{s}_2$ be as follows.

$$ES_{\mathtt{s}_1,\mathtt{f}} = \{\langle o'_1,\{\mathtt{read},\mathtt{write}\},\mathbf{SR},u_1\rangle\};\ A_{\mathtt{s}_1} = \{\langle \mathtt{student@f},\mathtt{read},+,o'_1,*\ \rangle, \langle *,\mathtt{read},-,o'_1,\mathtt{jimmy}\rangle\}$$

$$ES_{\mathtt{s}_2,\mathtt{f}} = \langle o'_2,\{\mathtt{read}\},\mathbf{FC},\mathtt{lsa}\rangle;\qquad A_{\mathtt{s}_2} = \{\langle *,\mathtt{read},-,o'_2,*@\mathtt{s}_1\rangle\}$$

Suppose objects o'_1 and o'_2 have been imported in the federation as o_1 and o_2, respectively. Let the federated schema and the authorizations at the federation be as follows:

$$FS_{\mathtt{f}} = \{\langle o_1,\{\mathtt{read},\mathtt{write}\},\mathbf{SR}\rangle, \langle o_2,\{\mathtt{read}\},\mathbf{FC}\rangle\};\ A_{\mathtt{f}} = \{\langle \mathtt{jeremy},\mathtt{read},o_2,*\rangle\},$$

where **jeremy** is a user belonging to groups **student** and **us-citizens**.

Consider now user **jim@s**$_3$, connected to the federation as **jeremy**, that submits a request to **read** o_1. Since the administrative policy of o_1 is site retained no control is executed at the federation and the request for the set of groups {**student, us-citizens**} (the groups to which **jeremy** belongs) to read o'_1 is sent to site s_1. Since s_1 enforces local authentication, **jeremy** will need to identify himself. Let **jimmy** be his identity at s_1. Access control for request ⟨{**student, us-citizens**}**@f**,**read**,o'_1,**jimmy**⟩ returns false due to the negative authorization stored at s_1 and therefore the access will be denied.

Consider now a request by the same user to read o_2. Access control at the federation level is successful thank you to the authorization for **jeremy**. Since the administrative policy is federation controlled no authorization is needed at local level, however, local control must be executed to make sure that no negative authorizations exist for the access. Hence, a request for the set of groups {**student, us-citizens**} with remote login **jim@s**$_3$ to access o_2 is sent to site s_2. Since site s_2 enforces global authentication, no further authentication is necessary. Since no negative authorization denying the access exists (the only negative authorization applies only to user remotely connected from s_1) the reply of site s_2 is positive and the access will be granted. □

7 Conclusions and Future Research

We have presented an authorization model for the protection of information in a tightly coupled federated system. The model allows users to decide which of their objects to share with the federation and how to share them. In particular they can specify the specific access modes allowed by federated users and the administrative policy establishing how authorizations on objects are to be specified. Inclusion of objects in the federated schema is a two step-process requiring agreement of both the administrator of the object at one side and the administrator of the federation at the other. Authorizations can be specified both at the global level, on federated objects, and at the local level on local objects exported to the federation. An algorithm describing the controls to be executed at the federation and at each local site to determine whether to grant or deny access to federated data has also been presented. The model results flexible and able to support different kinds of protection requirements users may have.

Our paper leaves space for further work. A first issue we plan to investigate is the relationship between global and local authorizations. In the paper we have assumed that global and local authorizations are specified independently. The approach could be extended to the consideration of different strategies for the specification of authorizations. A first possible strategy is that upon specification of a global authorization, the local authorizations needed for the access globally granted to be allowed are derived and inserted at the local sites. Another strategy could be that of deriving global authorizations on federated objects from the authorizations specified locally. A further issue to be investigated concerns the communication of identities or group memberships between the federation and the sites. In this paper we have assumed that local access control on requests by

a federated user is always carried out with reference to the groups to which the user belongs, and that the federation communicates these groups to the local sites. Alternative approaches can be considered such as the use of a call back mechanism, where a site explicitly asks whether a user belongs to a group, or the use of credentials [6]. Moreover, our model could be made more precise by considering at the global level specific data model. In this case, all the issues related to the translation of the export schemas into this data model should be taken into consideration. Finally, a further issue that can be investigated concerns the consideration of component sites with heterogeneous data models or heterogeneous security policies, and of different cooperation rules that can be established among sites to regulate access to the shared data.

References

1. M. Abadi, M. Burrow, B. Lampson, and G. Plotkin. A Calculus for Access Control in Distributed Systems. Technical Report 70, DEC, System Research Center, Palo Alto, February 1991.
2. Barbara T. Blaustein, Catherine D. McCollum, Amon Rosenthal, and Kenneth P. Smith. Autonomy and Confidentiality: Secure Federated Data Management. In *Proceeding of the 2nd International Workshop on Next generation Information Technologies and Systems*, Naharia, Israel, June 1995.
3. S. Castano, M.G. Fugini, G. Martella, and P. Samarati. *Database Security*. Addison-Wesley, 1995.
4. M. L. Goyal and G. V. Singh. Access Control in Distributed Heterogeneous Database Management Systems. *Computers & Security*, 10:661–669, 1991.
5. D. Heimbigner and D. McLeod. A Federated Architecture for Information Management. *ACM Transactions on Office Information Systems*, 3(3):253–278, 1985.
6. V. E. Jones, N. Ching, and M. Winslett. Credentials for Privacy and Interoperation. In *Proc. New Security Paradigms Workshop*, pages 93–100, La Jolla, California, U.S.A, August 1995.
7. Dirk Jonscher and Klaus R. Dittrich. Access Control for Database Federations a discussion of the state-of-the-art. In *Proceeding DBTA Workshop on Interoperability of DBSs and DB Applications*, October 1993.
8. Dirk Jonscher and Klaus R. Dittrich. An Approach for Building Secure Database Federations. In *Proceedings of the 20th VLDB Conference, Santiago, Chile*, 1994.
9. Dirk Jonscher and Klaus R. Dittrich. Argos — A Configurable Access Control Subsystem Which Can Propagate Access Rights. In *Proc. 9th IFIP Working Conference on Database Security*, Rensselaerville, New York, U.S.A, August 1995.
10. Wom Kim, Nat Ballou, Jorge F. Garza, and Darrel Woelk. A Distributed Object-Oriented Database System Supporting Shared and Private Databases. *ACM Transactions on Office Information Systems*, 9(1):31–51, January 1991.
11. Witold Litwin, Leo Mark, and Nick Roussopoulos. Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys*, 22(3):267–293, 1990.
12. J. McHugh and B. Thuraisingham. Multilevel Security Issues in Distributed Database Management Systems. *Computers & Security*, 7:387–396, 1988.
13. B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications Magazine*, 32(9):33–38, 1994.

14. Martin S. Olivier. A Multilevel Secure Federated Database. In *Proc. 9th IFIP Working Conference on Database Security, Rensselaerville*, pages 23–38, New York, U.S.A, August 1995.
15. R.S. Sandhu and P. Samarati. Access Control: Principles and Practice. *IEEE Communications*, pages 2–10, September 1994.
16. M. Satyanarayanan. Integrating Security in a Large Distributed System. *ACM Transactions on Computer Systems*, 7(3):247–280, August 1989.
17. Amit P. Sheth and James A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
18. M. Templeton, E. Lund, and P. Ward. Pragmatics of Access Control in Mermaid. In *IEEE-CS TC Data Engineering*, pages 33–38, September 1987.
19. Gomer Thomas, Glenn R. Thompson, Chin-Wan Chung, Edward Barkmeyer, Fred Carter, Marjorie Templeton, Stephen Fox, and Berl Hartman. Heterogeneous Distributed Database Systems for Production Use. *ACM Computing Surveys*, 22(3):237–266, 1990.
20. B. Thuraisingham. Multilevel Security Issues in Distributed Database Management Systems II. *Computers & Security*, 10:727–747, 1991.
21. B. Thuraisingham and Harvey H. Rubinovitz. Multilevel Security Issues in Distributed Database Management Systems III. *Computers & Security*, 11:661–674, 1992.
22. Ching-Yi Wang and David L. Spooner. Access Control in a Heterogeneous Distributed Database Management System. In *IEEE 6th Symp. on Reliability in Distributed Software and Database Systems, Williamsburg*, pages 84–92, 1987.
23. Edward Wobber, Martin Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos Operating System. *ACM Transactions on Computer Systems*, 12(1):3–32, 1994.