4

Argos – A Configurable Access Control System for Interoperable Environments

D. Jonscher, K.R. Dittrich
Universität Zürich, Institut für Informatik, Winterthurerstr. 190
CH – 8057 Zürich
{jonscher,dittrich}@ifi.unizh.ch

Abstract

The integration of autonomous information systems causes a fundamental problem for security management. How to ensure a consistent authorisation state if several independent software components are involved, each having an access control system of its own? In other words, how to ensure an organisation-wide security policy?

Argos has been developed for the CHASSIS[1] project, where it serves as an access control system at the global layer of heterogeneous database federations. However, it can be used for any object-based system. The Argos mechanisms are very flexible; it is possible to enforce a variety of security policies in the area of identity-based access control (discretionary access control with several mandatory extensions).

Since database federations have to take the autonomy of component systems into account, Argos is able to propagate global authorisations to the involved (local) component systems. Autonomy means that the local systems are free to accept or reject these propagated authorisations. Therefore, the global system has to act as a coordinator of the involved component systems, which includes the enforcement of failure protocols.

In this paper, we focus on the propagation of global authorisations from the global to the local layers. Further, we describe the functionality of Argos, i.e. the spectrum of policies Argos is able to enforce.[2]

Keywords

access control, federated database systems, interoperability, object-oriented database systems

---

1 Configurable Heterogeneous And Safe, Secure Information System (a joint project, together with the University of Geneva and ABB Baden, funded within SPP Informatik-Forschung under project number 5003-34355)

2 This is a considerably reduced version of the paper that has been presented at the IFIP WG 11.3 conference. Due to lack of space the algorithms implementing the revocation approach (Section 3.6) and the validation of accesses (Appendix) have been omitted. Furthermore, all formulas have been removed. A complete version of this paper can be requested from "jonscher@ifi.unizh.ch".

D. L. Spooner et al. (eds.), *Database Security IX*

1 Introduction

Information is one of the most important resources of our society. Various "data repositories" are used to cope with the huge amount of data. Beginning in the sixties, database management systems (DBMS) have proved to be the most attractive, convenient and secure vehicle to manage data. However, they were mainly used for particular application domains and have evolved independently from each other, burdened with several million lines of code to serve many different applications.

Nowadays, there is an increasing need to integrate existing data repositories into (apparently) homogeneous systems that allow queries involving several independent databases, enforce global integrity constraints, etc. Unfortunately, existing application programs often prevent from moving the data into a new distributed database system. In this case, integration has to be carried out without affecting existing applications.

This led to a recent research direction in the area of database technology, the development of so-called database federations /ShLa 90/. Federations offer a global unified view on existing component database management systems (CDBMS) while preserving local autonomy. The most promising approach to realise a federated database management system (FDBMS) is to use object-oriented technology, especially if we have to cope with heterogeneous CDBMS /SaCG 91, HäDi 92, NiWM 93, Kent 93/.

FDBMS have to offer strong security mechanisms to "convince" the administrators of CDBMS that it is reasonable to join the federation. As a part of the CHASSIS project, which aims at the design and implementation of secure interoperable systems, we have developed and implemented an access control concept for database federations /JoDi 93a, JoDi 93b, JoDi 94/ (using C++ and ObjectStore, a commercial, C++ based, object-oriented database system). In this paper (which is the follow-up paper of /JoDi 94/), we describe the access control system – called Argos – that is used by the database federation to evaluate global accesses against the global security policy, as well as to propagate global authorisations to the involved component systems.

The contributions of Argos are:

- a configurable access control system which is able to enforce a variety of access control policies in the area of identity-based access control
- a new implementation technique for cascading as well as non-cascading revocations of access rights which is semantically equivalent to the approach of System R /GrWa 76, Fagi 78/ with the extensions proposed in /BeSJ 93, BeSJ 95/
- an (implemented) approach to propagate global authorisations to autonomous local systems by mapping global onto local access rights

The basic access control concept for object-oriented DBMS which is also included in Argos is not a contribution, because several similar approaches in the area of identity-based access control already exist /Ahad 92, BeJS 93, BeOS 94, Bert 92, Brüg 92, FaSp 91, FeGS 89, FeWF 94, GaGF 93, GuSF 91, HuDT 93, HuDT 94, LGSF 90, NyOs 92, NyOs 93, PfHD 88, RaWK 88, RBKW 91, Spoo 88, TiDH 92a, TiDH 92b/. Our contribution in this area is the integration of a domain concept into an object-oriented environment by using method classes. The related work in the area of access control for FDBMS is discussed in /JoDi 93a/. To our knowledge, this is the first paper where a propagation of authorisations to autonomous component systems is discussed.

The remainder of this paper is organised as follows: In Section 2 we describe the object-oriented data model that Argos assumes at the global layer. Section 3 gives an overview of the access control mechanisms of Argos, including the definition of access rights, the calculation of implicit access rights, the authorisation paradigm and the revocation approach. The access control policies that can be enforced by Argos are presented in Section 4. Afterwards, we describe in Section 5 the principle of the propagation of authorisations as well as how it has been implemented. The summary and an outlook are given in Section 6.

2 The data model

Argos has been tailored towards object-oriented database systems according to /Atki 89/, but only the very basic features of object-orientation are assumed, so that Argos can also be used for other systems.

An object-oriented database consists of a collection of cooperating, interrelated and distinguishable units, called objects. *Objects* have an identity independent from their value, a *state*, which is based on a – usually structured – value, and a behaviour. The state is *encapsulated*, i.e. it can only be observed or modified by invoking methods via a well-defined interface (*information hiding*). These methods define the *behaviour* of an object.

Every object is an instance of a *type* which defines its structure (of the value) and behaviour. This intentional interpretation of types corresponds to the database notion of a schema. The set of objects which are instances of a type is called the *extension* of a type.

Types are organised in *type hierarchies*. A subtype inherits structure and behaviour from its supertypes. The inheritance paradigm is based on *inclusion* and *substitution inheritance* /Atki 89/. An instance of a type also belongs to all extensions of the type's supertypes. Moreover, it is able to answer every request that can be sent to an instance of its supertypes. A subtype can specialise inherited methods by overriding. Conflicts due to multiple inheritance (two inherited methods have the same name) must be solved explicitly, i.e. a local method must be defined that overrides all inherited methods which are in conflict with each other (see /JoDi 93b/ for a formal definition).

3 The global access control mechanisms

3.1 Subjects

Three kinds of *subjects* (in the sense of authorisation units) are supported: users, roles and subject domains.

Users (or applications) are basic subjects issuing requests that have to be checked by the access control system. Let U be the set of users.

Roles are abstract users reflecting the organisational, functional or social position of users in the universe of discourse. Concrete users can "play" a role, thus obtaining the access rights that have been granted to a role. Users can be associated with several roles. To benefit from access rights that have been granted to a role, a user must activate the role explicitly (by executing an Argos command). Several roles can be activated concurrently by the same user. The activation of roles is persistent, i.e. roles remain activated across user sessions.

Conflict relations can be defined in order to restrict which roles cannot concurrently be activated (*activation conflict relation*), and which roles a user cannot concurrently be associated with (*association conflict relation*). The former prevents forbidden accumulations of rights, and the latter supports separation of duties principles /ClWi 87/.

Subordination relationships can be defined between roles. These relationships are used to infer implicit rights (cf. Section 3.4.2). The intended semantics of this hierarchy is that a role being higher in the role hierarchy has strictly greater power than any of its subordinated roles.

Users and roles are administered by security administrators. "Security administrator" (SA) is a predefined role, i.e. users who have activated this role are acting as SA.

Subject domains (sets of subjects) can be used to define more general authorisation units. Elements of subject domains are users and/or roles. Subject domains can be nested. The resulting algebraic structure is a poset. The semantics of this hierarchy is that all elements which belong to a subject domain *sd* also belong to all superdomains of *sd*.

3.2 Protection objects

According to the data model, type extensions are the basic protection objects[3], but for simplicity we will subsequently consider types extensionally. Whenever a type appears as a protection object, we refer to its extension.

Protection object domains (sets of protection objects) can be used to define more abstract protection objects. Elements of protection object domains are types. Analogously to subject domains, protection object domains can be nested, provided cycles are not introduced. The semantics of this hierarchy is that all elements which belong to a protection object domain *pd* also belong to all superdomains of *pd*.

3.3 Actions

Argos supports two kinds of actions, methods and method classes. *Method classes* represent sets of methods of the same type. The creator of a type can define several method classes for this type, e.g. "read_methods" and "write_methods". The following restrictions apply:

- Each method can at most belong to one method class (an exception is the predefined method class "ALL"; all methods belonging to a type are implicitly associated with ALL).
- A type inherits method classes from its supertypes. Inheritance conflicts between method classes also have to be solved by overriding.
- Name conflicts between methods and method classes are forbidden, i.e. action names have to be unique for every type.
- A method can be associated with an inherited method class. However, the association of an inherited method with a method class cannot be overridden by a subtype (but the method can be overridden and afterwards associated with a different method class).

Method classes are required to define meaningful access rights for protection object domains. Method names usually are type-specific. Thus, it makes no sense to define a right to execute a method for a set of types, because in general only few of these types will have a method with this name. Method classes, however, allow for specifying generic rights if the same name is chosen for semantically equivalent method classes of different types.

3.4 Access rights

3.4.1 Explicit access rights

An Argos access right defines a relationship between a subject, a protection object and an action (which is equivalent to a variation of the protection matrix /GrDe 72, HaRU 76, Lamp 71/). They allow or forbid a (set of) user(s) to execute a (set of) method(s) for a (set of) type(s). An access right can be represented as a six-tuple:

(subject, protection object, action name, kind of right, grantor, grant option)

"Subject" stands for the grantee. Two kinds of rights are supported, permissions and prohibitions. The grantor represents the user who has granted that access right. Grant options only apply to permissions. They allow the grantee(s) to pass on that access right to other subjects /GrWa 76, Fagi 78/. Note that "grant()" and "revoke()" are generic methods that can be executed on any type, i.e. Argos also supports authorisation rights beyond grant options (similar to SeaView /Lunt 88/).

---

[3] A later release of Argos will also support individual objects as protection objects.

3.4.2 Implicit access rights

A set of pre-defined rules is used to infer implicit access rights /FeSW 81, Brüg 92, RaWK 88, RBKW 91/ from explicit access rights. Note that the rules have to be applied recursively.

- An access right for a role implies the same right for all users who have activated this role.
- A permission for a role implies the same permission for all superior roles. A prohibition for a role implies the same prohibition for all inferior roles. This results in a system where roles have more authority than their subordinated roles.
- An access right for a subject domain (protection object domain) implies the same right for all elements of this domain.
- An access right for a method class implies the same access right for all methods that belong to the method class.[4]
- An access right for a type implies the same access right for all subtypes of the type.
- A permission with grant option implies the permission to grant the same permission to other subjects.

3.5 The authorisation state

The authorisation state determines which *requests* (of *users* to execute a *method* of a *type*) are permitted and which ones are not.

Note that even the set of implicit access rights does not yet reveal the authorisation state, because conflicts between permissions and prohibitions are possible. A *conflict* between two rights is given if they refer to the same grantee, the same protection object and the same action, but one is a permission and the other a prohibition. Argos applies a simple conflict resolution policy: prohibitions always override permissions.

Furthermore, the set of access rights (explicit and implicit ones) needs not be complete, i.e. there may be requests where neither a permission nor a prohibition applies. Argos supports both closure assumptions, the *open world assumption* (unspecified requests are always permitted) and the *closed world assumption* (unspecified requests are always forbidden).

3.6 The authorisation paradigm

Argos implements a combination of the ownership and the administration paradigm. Every protection object has an owner who is either a concrete user or a pre-defined "user" SYSTEM.

An administration paradigm is applied for all objects that are owned by SYSTEM. Security administrators act on behalf of this artificial owner (centralised authorisation).

If a concrete user is the owner of a protection object, this user is responsible for authorisations concerning her protection object.

Thus, the kind of owner determines the authorisation paradigm to be applied. The administration paradigm can be considered as a special case of the ownership paradigm. The owner can transfer her property to another user or to SYSTEM.

However, the owner only determines the original source of authority for a protection object. She is free to grant permissions with grant option, or even permissions to execute the grant method (grant permissions) to other users. Analogously, grant prohibitions, or even revoke permissions/prohibitions can be granted.

Security administrators need a way to restrict who is permitted to participate in decentralised authorisation if an ownership paradigm is applied. Decentralised authorisation means that an

---

[4] Action names are only resolved for individual types. If the protection object of an access right is a protection object domain, at first the types which belong to that domain are retrieved, and afterwards the action name is resolved. Rights where the action name does not refer to an action of the corresponding type are simply ignored.

access right should be granted where neither the grantor nor the grantee is a security administrator. Such restrictions are also based on roles.[5] Each role can get a *giveAuthority*- and/or a *getAuthority*-privilege. If a user *u* wants to grant an access right to another subject *s*, *u* must have *activated* a role having the *giveAuthority*-privilege. The grantee on the other hand needs the required privilege to get access rights from other users than security administrators:

- If the grantee is a role it must have the *getAuthority*-privilege.
- If the grantee is a user, he must be *associated* with a role having this privilege.
- In case of subject domains, every user or role being a direct or indirect element of this domain needs the required privilege.

Otherwise, the authorisation fails.

Subsequently, we assume that both, the grantor and the grantee are permitted to participate in decentralised authorisation. If so, a user *u* can grant an access right (not) to execute an action on a protection object to a grantee if:

- *u* is the owner of the corresponding protection object (or *u* is acting as a security administrator in case of system-owned objects), or
- *u* has a grant permission for this protection object (covering the action to be granted) that is not overridden by a grant prohibition, or
- *u* has the permission she wants to grant with grant option, and this permission is not overridden by a prohibition.

Note that the latter only applies to *permissions* to be granted, whereas the former two conditions allow for granting both, permissions and prohibitions.

A user *u* can revoke an access right (not) to execute an action on a protection object from a grantee if the right to be revoked exists (as an explicit access right) and one of the following conditions is fulfilled:

- *u* is the owner of the corresponding protection object (or *u* is acting as a security administrator in case of system-owned objects), or
- *u* has a revoke permission for this protection object that is not overridden by a revoke prohibition, or
- *u* has granted the access right to be revoked.

Note that the last revoke condition holds independently from the current authorisation state. If a user has got an access right with grant option and passes it on to another subject, she can even revoke this access right if she later gets a prohibition that overrides the permission which gave her the authority to pass on that access right. A different policy is described in /BeSJ 93, BeSJ 95/.

A fundamental problem in case of decentralised authorisation which is based on grant options is the semantics of the revocation of rights. Griffith and Wade /GrWa 76/ have defined a formal semantics of recursive revocations for such systems (later corrected by Fagin /Fagi 78/). Their approach is based on the assumption that the revocation of an access right *ar* should result in an authorisation state which is equivalent to a situation where *ar* has never been granted. Hence, if a permission has been granted with grant option from A to B, and B has passed on this permission to C (with or without grant option), a revocation of the permission from B requires an automatic revocation of C's permission as well (cascading revocation).

The corresponding algorithm is based on time-stamps and the information who has granted an access right (the grantor). Since this algorithm is well known and has been implemented in several commercial relational database systems (e.g., in Oracle), we omit a detailed description (see /GrWa 76, Fagi 78/ for a comprehensive discussion).

---

5 Argos mainly is a role-based system. Although access rights can also be granted to concrete users, we consider this as the exceptional case according to the Argos philosophy. In principle, however, it would also make sense to grant authorisation privileges to concrete users.

The need for non-cascading revocations has already been justified in /BeSJ 93, BeSJ 95/. If, for instance, a user changes his position in a company, his access rights have to be adjusted accordingly. If some of his access rights must be revoked, it usually makes no sense to revoke access rights he has granted (based on the authority of his position) to other users as well. Hence, the System R approach has been extended by groups, prohibitions and non-cascading revocations /BeSJ 93, BeSJ 95/.

Argos also supports cascading as well as non-cascading revocations. It is up to the revoker to decide which scheme has to be applied. However, Argos supports additional access control mechanisms beyond those mentioned in /BeSJ 93, BeSJ 95/:

- Authorisation rights are not necessarily combined with access rights (grant options). It is also possible to grant rights to execute the grant method.
- Argos supports implicit access rights beyond groups (based on role hierarchies, (nested) subject domains, type hierarchies, (nested) protection object domains, and method classes).

The latter is the reason that the time-stamp algorithm does no longer work well. Without implicit access rights, it is easy to calculate which access rights have to be revoked recursively in case of a cascading revocation of an access right. The authorisation chain can easily be determined by matching (recursively) the grantee of the right to be revoked with the grantor of rights for the same protection object and the same action. The time-stamp is required in case of multiple sources of authority (for the grantor) to check which authority has been passed to the grantee.

In Argos, however, the situation is much more complicated. Therefore, we have implemented a different algorithm that is semantically equivalent to the Griffith and Wade scheme (with the modification given in /BeSJ 93, BeSJ 95/ which is required in systems using prohibitions). Descent relationships are now explicitly managed. Each access right keeps a set of references to its ancestors and another one to its descendants. These relationships – which obviously are inverses of each other – are automatically managed by Argos, and provide for the basis of the revocation algorithm. Due to lack of space we cannot present the algorithm here, but it is included in the conference version of this paper (cf. Footnote 2).

3.7 Restrictions of the domain concept

Authorisations not only happen if a grant or revoke command is executed. An implicit authorisation takes place if a more general authorisation unit (than a single user, type or method) is changed. In particular, if an element is added to a subject domain, the new element gets the access rights which do already exist for this domain. Analogously, if an element is added to a protection object domain, the grantees of rights which are based on this domain get the corresponding access rights for the new element. Therefore, adding/removing an element to/from a domain requires several checks:

- The adding/removing user needs an access permission for the domain itself.
- The adding/removing user must be authorised to grant/revoke existing access rights which are based on this domain according to the new/old element.

The first case is very simple in Argos, since domains (subject as well as protection object domains) are protection objects of their own. Thus, accesses to domains are subject to access control like any other request.

The latter case, however, is more complicated. If many access rights based on a domain already exist, the overhead which is caused by checking the second constraint mentioned above may be tremendous. Hence, the current version of Argos enforces additional restrictions for domains:

- Any user can create a domain. He becomes the owner of this domain (if a security administrator creates a domain, SYSTEM becomes its owner).

- Only the owner of a domain can add/remove elements to/from it, and can grant access rights based on this domain.
- Domain structures have to be homogeneous with respect to ownership. This means that subdomains have to have the same owner as their superdomain. Moreover, the elements of protection object domains (types) must have the same owner as the domain itself.

These principles are not as restrictive as they may appear at first glance. Most examples of applications where domains have been used rely on an administration paradigm. In this case, all protection objects are owned by SYSTEM and are administered by security administrators, i.e. the above mentioned restrictions are trivially fulfilled. The situation becomes complicated if domains are combined with the ownership paradigm.

Now, a request to change the extent of a domain can easily be checked. The user *u* who has requested the change must be the owner of the domain. Nothing more is required in case of protection object domains, because the owner of the domain is also the owner of the elements of this domain and is thus permitted to grant/revoke access rights to/from these elements (cf. Section 3.6). Further, if access rights based on this domain already exist, *u* as well as the grantees of these rights are permitted to participate in decentralised authorisation. In case of subject domains which are not owned by SYSTEM, however, it has to be checked whether the new element (being a user, a role or another subject domain) is permitted to participate in decentralised authorisation. Furthermore, it has to be checked whether *u* is also permitted to grant/revoke all existing access rights for this subject domain to/from the new element.

4 Access control policies that can be enforced by Argos

Requests are evaluated by instances of the predefined type "ReferenceMonitor". Each reference monitor (it is possible to have several reference monitors at the same time, but only a single monitor can be active for a system) encapsulates a security policy. Its state (19 attributes) determines the policy to be enforced. The settings of the attributes reflect the following policy aspects:

- the access rights to be supported (permissions, prohibitions, or both)
- the kind of closure assumption (open or closed world assumption)
- the authorisation paradigm to be applied (ownership or administration paradigm)
- the authorisation mechanisms (grant options, authorisation rights, both, or even none)
- should decentralised authorisation be restricted (using authorisation privileges)
- should roles be used, and if so should they be arranged in a role hierarchy (poset)
- should subject (protection object) domains be used, and if so should it be possible to nest them
- should implicit access rights be inferred along the type hierarchy
- should method classes be used
- should a context authentication service for access rights be used /HoTe 95/

Meaningless combinations like a positive system (only permissions can be granted) with the open world assumption, or grant options in a negative system are automatically detected and rejected.

These attributes determine the behaviour of the corresponding reference monitor. All methods check the current policy and behave accordingly (e.g., if roles but not role hierarchies are used, Argos does even not know the command to define a subordination relationship; accordingly, the rule to infer implicit access rights along the role hierarchy is disabled).

This flexibility permits to choose only a subset of the mechanisms offered by the "tool-kit" Argos that is required for a concrete application. Using all Argos features at the same time would probably result in a system which can hardly be managed.

The security policy can even be changed on the fly. In this case, however, the authorisation base (set of explicit access rights) is not transformed. Information about features that are no longer supported is not deleted; it is simply ignored. It is possible, for instance, to define a role hierarchy, and to change the policy afterwards so that subordination relationships cannot be established anymore. Although the existing relationships are not deleted, they are ignored during the evaluation phase of a request.

5 The propagation of authorisations in interoperable environments

Data repositories that are used today are often burdened with several million lines of code (so-called legacy applications; cf. Figure 1a). New applications which benefit from today's modern network technologies require the integration of existing data. Since replication is not only a waste of storage space but also causes considerable consistency problems if replicated data are updated, interoperable environments are mushrooming. In such environments, new systems can be built on top of existing ones. They provide for new services to global users (or applications), via a uniform interface (cf. Figure 1b). Persistent data that are needed for this service (*global system*) are not necessarily directly managed by the global system (although it may have its own data store as depicted in the figure); instead it gets the data from other data repositories (*local systems* like DBS, file systems, etc.).

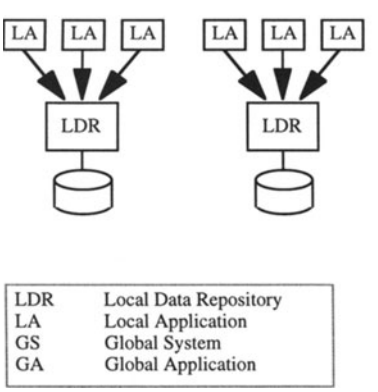

Figure 1a Isolated local systems.

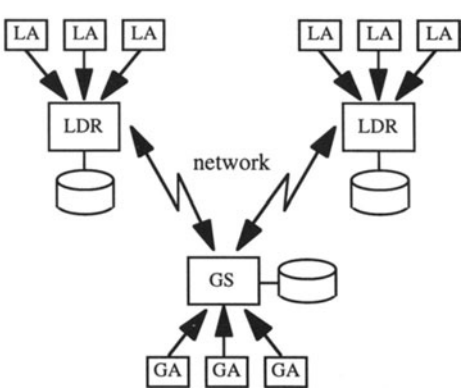

Figure 1b Interoperable environment.

Local data repositories usually have their own access control system. In many cases, however, the global system also needs an access control system. In /JoDi 94/ we have motivated the need for a global access control layer for a special case of interoperable systems, namely for database federations /ShLa 90/. New protection objects emerge at the global layer, where local systems only contribute parts to the construction of these virtual objects. Hence, only the global system can care for their proper protection. Another obvious reason for a global access control layer is given in environments where personal information is managed. Integrated systems aggravate privacy problems, and the global system has to offer very flexible access control mechanisms to enforce an appropriate security policy.

In interoperable systems, we are also faced with a specific authorisation problem. Local systems usually are autonomous, i.e. they enforce an access control policy independent from the global system. If the global system mediates a request to a local system, the local system checks whether this request is permitted from its own point of view, even if the request has already been evaluated against the security policy of the global system. In this case, a cooperation protocol between both access control layers is required in order to achieve consistency between the global and the local authorisation states. It is not satisfactory for global users if the global system permits their request, but the execution of the requested global operation fails due to insufficient local authorities. This problem is solved by the coupling part of Argos which has not been mentioned yet.

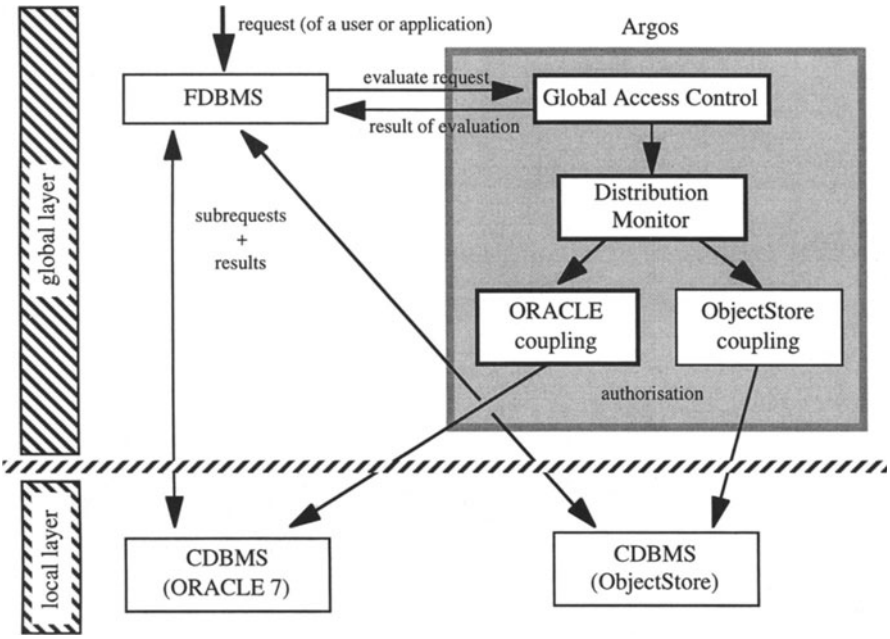

Figure 2 The Argos architecture.

Note that such a cooperation is not restricted to the area of database federations. It is required in any client/server system where both, the client and the server enforce their own access control policy.

In the following, we describe the coupling protocols and algorithms of Argos, exemplified in an environment where Argos is used as a global access control system of a tightly coupled database federation (cf. Figure 2; see /JoDi 93a, JoDi 94/ for a detailed discussion of tightly coupled federations and the security problems to be solved).

The FDBMS applies an operational integration, i.e. it offers global types with appropriately defined methods to global users. Users need not care which component systems are involved

(location transparency). The integration is actually carried out in the implementation of the global methods where the required calls of the involved component systems are embedded.[6] Further, our prototype is based on two component systems, a relational DBMS (Oracle, release 7.1) and an object-oriented DBMS (ObjectStore; release 3.1). Since ObjectStore does not yet provide for an access control concept beyond the operating system mechanisms (file protection), we have so far only implemented the Oracle coupling. Figure 2 shows the corresponding architecture (bold boxes represent parts of Argos that have been implemented).

If a request is submitted to the FDBMS, the FDBMS invokes the responsible reference monitor of Argos. The monitor checks whether the request is permitted and returns the result of this evaluation to the FDBMS, which has to enforce the decision.

Argos supports two levels of local autonomy (of CDBMS; cf. /JoDi 94/). *Low authorisation autonomy* means that the CDBMS does not check which global user is accessing its data. It only verifies that the request has been mediated by the FDBMS, and uses the identifier of the FDBMS for local access control. This scheme requires some trust of the CDBMS in the access control policy of the FDBMS. If a CDBMS does not want to trust the global system, it can insist on *full authorisation autonomy*. In this case, a CDBMS uses the (local) identifier of the corresponding global user (which has locally been authenticated) on behalf of whom the FDBMS has mediated the request to the CDBMS.

Full authorisation autonomy is the interesting case from the scientific point of view. If a global user grants a permission to another global subject, it is not sufficient to check whether the authorisation is permitted from the global point of view. When the global grantee (more precisely, a user to whom this global access right applies) issues a request that has been permitted by the global authorisation, she also needs the corresponding local permissions. Otherwise, the global permission would be useless. Our main objective is to ensure a *consistent global authorisation state*. Consistency means that a request which is permitted from the global point of view will not fail due to an access rejection by a CDBMS. Note that we do not want to ensure equivalence between both layers. Although an equivalent authorisation state seems to be desirable, it is almost impossible to achieve if we have to cope with heterogeneous CDBMS. Each CDBMS supports different access control mechanisms which are usually much simpler than those of Argos. Argos can support almost every identity-based mechanism of today's commercial DBMS, but not vice versa. If, e.g., a CDBMS only supports a coarse granule of protection objects, it may happen that a global permission requires a more general local permission. This amplification of rights (which has already been mentioned in /WaSp 87/) may be a problem if global users can also directly access the local system. Obviously, this amplification cannot occur in case of low authorisation autonomy /JoDi 94/.

Consistency can be achieved by involving all CDBMS which require full authorisation autonomy in the evaluation of a global authorisation. The principle is quite simple. If the global authorisation is permitted from the global point of view, Argos determines the CDBMS that have to be involved. Afterwards, it checks whether the corresponding local access rights are already given to the grantee(s). If not, Argos tries to grant the required local rights on behalf of the global grantor. This process is called *propagation of authorisations* (from the global system to the local systems). If at least one of the required local authorisations fails, the whole propagation process is undone, and the global authorisation also fails. However, it depends on the semantics of a global method whether all local rights must be available or whether it also makes sense to execute a global method when some of the corresponding local requests are rejected. An example of the latter is a retrieval system for a set of libraries. Even if some of the libraries are not accessible, a retrieval can be useful for a global user. Since such semantic issues are beyond the capabilities of computers, it is up to the global grantor to choose the appropriate protocol, i.e. Argos supports both:

---

6 Although operational integration seems to be inferior to schema integration approaches /ShLa 90/, it represents the state of the art if heterogeneous component systems have to be integrated.

a) consistency protocol: a global authorisation only succeeds if it is permitted from the global point of view *and* if all required local access rights are available or can be granted

b) best effort protocol: a global authorisation succeeds if it is permitted from the global point of view, and Argos attempts to get as many local rights as possible

In both cases, the global grantor gets a report *which* local rights are neither available nor can be granted, because in general the global grantor cannot know which local accesses are required for the successful execution of a global method. Afterwards, the global grantor can try to persuade the local authorities to get the missing local rights.

5.1 Required mappings

Suppose a global user *u* wants to grant a permission *ar* to a global subject *s*.[7] If the authorisation is permitted, Argos has to find out which local permissions correspond to the granted global permission. This requires the following information:

- Global and local security administrators have to agree upon defining the mappings of global onto local subjects. For each CDBMS a mapping has to be defined which associates each global user (role) with exactly one local user (role) or with "0". A subject that is mapped onto "0" has no local equivalent, i.e. it is not locally known.
- The designers of global methods must specify which local objects have to be accessed using which local action, when their methods are executed. (*)

In general, a global permission corresponds to several local permissions. Generalisation concepts of Argos that are not supported by the corresponding CDBMS (like domains) are handled according to the following principles:

- Global permissions which are based on a domain *d* are automatically mapped onto local permissions for the elements of *d*. Method classes are treated analogously.
- Authorisation rights are mapped onto access rights with grant option (for systems supporting grant options).

5.2 The distribution monitor

If the authorisation is permitted from the global point of view, *ar* is passed to the distribution monitor *dm* (usually only one exists). This monitor is the coordinator of the propagation process. It determines which CDBMS are affected, passes the information which local access rights are required to the involved coupling modules, and collects the information which local rights are neither available nor can be granted. If a coupling module reports a failure and the global grantor requested the consistency protocol, the distribution monitor is also responsible to enforce the failure protocol. The following steps are executed:

1. *dm* checks the protection object and the action of *ar*, and determines all pairs of global types and methods that are covered by this access right
2. *dm* determines which CDBMS are involved (the covered methods are known from step 1, and the required local capabilities can be retrieved according to (*))
3. the involved coupling modules receive the set of local capabilities they are responsible for, as well as the authorisation protocol ("consistency" or "best-effort")
4. each coupling module checks its set of local capabilities, and reports the result to the distribution monitor (cf. Section 5.3).

---

[7] Currently, we only consider the propagation of permissions since almost no commercial CDBMS supports prohibitions. Although it is possible to propagate only non-overridden permissions, the overhead seems to be prohibitive. Therefore, we do not consider this case for the first version of Argos.

If the authorisation protocol requires consistency and at least one coupling module has encountered an error, the whole global authorisation has to be rolled back. For this purpose, all involved coupling modules remember which local permissions they have granted during the current propagation. The distribution monitor informs each coupling module that has *not* found an error (for its CDBMS) that all successful local authorisations have to be rolled back. All other coupling modules can initiate this rollback operation without a message from the distribution monitor, because they know the authorisation protocol to be applied. Note that this scheme is similar to a two-phase commit protocol.

The distribution monitor returns the result of the propagation to the reference monitor, and the reference monitor continues the global authorisation. If the distribution monitor has reported an error and the consistency protocol must be applied, the global authorisation is rejected, and the global grantor gets a report about the missing local permissions. The latter also happens if the best effort protocol is applied and a coupling module has found an error.

5.3 The coupling modules

Each CDBMS has its own coupling module. These coupling modules know which access control mechanisms are applied locally and which authorisation interface is provided there.

Concrete coupling modules are instances of coupling module types, because several component database systems may exist which are based on the same version of a DBMS. In this case, the coupling modules are identical.

Argos applies a framework approach for coupling modules, i.e. each coupling module type is a subtype of a predefined, abstract coupling module type. Subtypes (i.e. coupling modules for particular kinds of component systems) only have to override some predefined methods. Argos requires the following functionality for coupling modules:

- check whether a local subject name refers to a valid local subject (user and role names)
- check whether a local protection object name refers to a valid local protection object
- check whether a local action name refers to a valid local action (according to the associated local protection object)
- check whether a concrete local permission exists
- grant a local permission (using the local authorisation interface)
- revoke a local permission (using the local authorisation interface)

This functionality can even be provided by a file system. In case of UNIX, subject names can be verified by accessing the *passwd* file. Protection objects are files, and the names can be verified by accessing the corresponding directory file. Action names are "r" (read), "w" (write) and "x" (execute). The existence of an access right can be checked by accessing the i-node of the corresponding file. An authorisation is only possible if the coupling module can act on behalf of the file's owner (using the *setuid* bit). A local right can be granted for the group of the owner (if the grantee belongs to this group) or to *others* (all registered users).

In the following, we describe the algorithm that is used by a coupling module *ocm* for an Oracle CDBMS:

1. The coupling module *ocm* gets the following information from the distribution monitor:
   - the global grantee
   - a set of local capabilities LC that have to be checked by *ocm*
   - the global grantor
   - the authorisation protocol to be applied ("consistency" or "best-effort")
2. If the Oracle CDBMS only requires low authorisation autonomy (local accesses are validated using the identifier of the FDBMS itself), nothing has to be done and *ocm* reports a success to the distribution monitor.

3. Otherwise, *ocm* checks the type of the global grantee (user, role or subject domain) and determines all affected global users and roles (subject domains are not supported by Oracle 7). Note that the coupling module *ocm* has to consider the relationships between the global and the local role hierarchy. We can only require that for each global role a local role exists. However, local autonomy means that the subordination relationships need not be identical. Argos checks which subordination relationships are missing at the local level. For each local "gap", the corresponding global superior role is added to the set of affected global roles, i.e. implicit global rights that cannot be inferred at the local level are granted explicitly.
4. Subsequently, *ocm* checks whether all affected global users and roles have a local equivalent for this CDBMS. For each global subject not having a local equivalent, and each local capability, *ocm* adds an element to the set of missing local permissions MLP. If a local equivalent exists, the corresponding local subject is added to the set of affected local users ALU or roles ALR:
5. The following steps are executed for each local subject $x \in (ALU \cup ALR)$:
   5.1 An empty set of local capabilities Y is created, and *ocm* logs on to Oracle with its own local identity. For each element of LC (the set of local capabilities), *ocm* checks whether the corresponding local permission is already available to x (the permission to read the local authorisation relations must be granted to Argos when the component system joins the federation). All missing local capabilities are added to Y.
   5.2 If Y is not empty, it is checked whether the global grantor has a local mapping for the CDBMS. Such a mapping is required to generate a local authorisation request automatically.
   If the mapping is missing, then each element of Y is combined with x and added to MLP (the set of missing local permissions). Afterwards, *ocm* proceeds with step 6.
   Otherwise, *ocm* logs on to Oracle with the local identity of the global grantor (the password is either stored in the Argos schema or has to be typed in on the fly), and generates a local authorisation request for each element of Y.
   According to the reaction of Oracle, the corresponding right is either added to MLP or to the set of successfully granted local permissions GLP. The latter is only necessary if the authorisation protocol requires consistency. In this case, it may be required to undo all local authorisations.
6. If the set of missing local permissions MLP is not empty, an error is reported to the distribution monitor; otherwise, success is reported.
7. If MLP is not empty and the authorisation protocol requires consistency, then all elements of the granted local permissions GLP are revoked and removed from GLP. Otherwise, *ocm* waits for a message from the distribution monitor, which is either "commit" (clear GLP) or "rollback" (revoke all elements of GLP and remove them from GLP).[8]
   MLP is cleared by the reference monitor when the failure report has been passed to the global grantor.

6 Summary and outlook

In this paper we have presented Argos, a configurable access control system which provides for very flexible mechanisms in the area of identity-based access control. Various security

[8] For each granted global permission, Argos keeps track which local permissions have been granted during the propagation of this authorisation. This information is required if the global permission is later revoked. In this case the revocation must also be propagated to the involved CDBMS, i.e. GLP and MLP are still kept by Argos.

policies can be enforced, from discretionary access control (unrestricted ownership paradigm) to more restrictive policies (ownership paradigm with restrictions who is permitted to participate in decentralised authorisation (authorisation privileges)), even to mandatory access control[9].

Furthermore, we have implemented an algorithm to revoke access rights (cascading as well as non-cascading revocation) which is not based on time-stamps, but is semantically equivalent to the approach of System R /GrWa 76, Fagi 78/ with the extensions given in /BeSJ 93, BeSJ 95/.

We have described a framework for integrating heterogeneous component systems (which support identity-based access control) into a federated access control system, where component systems can choose among two levels of local autonomy (low or full authorisation autonomy).

The grantor of a global access right can specify whether he wants to ensure a consistent local authorisation state (otherwise, the global authorisation is rolled back), or whether he is satisfied with any local access right that can be acquired. Argos automatically propagates a global authorisation to the involved component systems. The success of the propagation depends on the local access rights of the global grantor.

The current state of the implementation is as follows: The Argos kernel (21 types with an average of 7 methods per type; about 350 KB C++ source code), the propagation part (10 types with an average of 7 methods per type; about 150 KB C++ source code) and the programmatic interface (8 types with a total of 126 methods; about 400 KB C++ source code) have been implemented and are currently being debugged. The user interface, which will be a menu-oriented character interface in the first place, has not yet been started.

When the user interface is finished, we will integrate Argos into an OMG CORBA /OMG 91/ environment by defining the IDL (interface definition language) mappings for each method of the programmatic interface (which provides for the full functionality of Argos). This way, Argos can be used as an access control service for any application that is connected to the corresponding object request broker.

Argos can also be used as a simulation tool to check the behaviour of users according to a concrete access control design. The latter is useful because Argos is able to explain its evaluation decisions. Furthermore, the security policy can be modified until the most appropriate solution has been found. We do not expect that Argos – as it is currently implemented – can be used for productive systems. Flexibility inherently degrades performance, i.e. Argos mainly is a research vehicle. However, the implemented mechanisms (design and code) can be optimised and reused for "real" applications, taking only those mechanisms that are required to implement the chosen access control policy.

In a later step of the project, we intend to develop a configurable coupling module. Such a module would simplify the integration of new component systems into the federation. Instead of overriding the required methods of the framework type, an integrator simply needs to register the local access control mechanisms that are available (e.g.: Users = yes; Roles = yes; RoleHierarchies = yes; SubjectDomains = no; AuthorisationParadigm = ownership; GrantOptions = yes; AuthorisationRights = no; ...) and an instance of this generic module picks the appropriate algorithm to propagate a global access right (perhaps even a prohibition) to the component system. The basic principle behind this idea has already been implemented by the reference monitor of Argos.

Acknowledgement

We are grateful for excellent comments from one of the anonymous reviewers. Furthermore, we would like to thank Jonathan Moffett from the University of York for many beneficial discussions and criticisms.

---

9 With "mandatory access control" we do *not* refer to multilevel security, but to an administration paradigm without decentralised authorisation.

References

/Ahad 92/ Ahad, R. et al.; *Supporting Access Control in an Object-Oriented Database Language*; Proc. EDBT '92, Vienna; Lecture Notes in Computer Science, 580, Springer-Verlag, 1992, 184-200

/Atki 89/ Atkinson, M.; Bancilhon, F.; DeWitt, D.; Dittrich, K.; Maier, D.; Zdonik, S.; *The Object-Oriented Database System Manifesto*; 1st International Conference on Deductive and Object-Oriented Databases, Kyoto, Dec. 1989

/BeJS 93/ Bertino, E.; Jajodia, S.; Samarati, P.; *Access Control in Object-Oriented Database Systems – Some Approaches and Issues*; Adams, N.R.; Bhargava, B.K. (eds.); Advances in Database Systems; Lecture Notes in Computer Science, 759, Chapter 2, Springer-Verlag, Berlin, 1993

/BeOS 94/ Bertino, E.; Origgi, F.; Samarati, P.; *A New Authorization Model for Object-Oriented Databases*; IFIP WG 11.3 8th Int. Conference on Database Security, Bad Salzdetfurth, Aug. 1994

/Bert 92/ Bertino, E.; *Data Hiding and Security in Object-Oriented Databases*; Proc. of the International Conference on Data Engineering, IEEE Computer Society Press, Phoenix, Feb. 1992, 338-347

/BeSJ 93/ Bertino, E.; Samarati, P.; Jajodia, S.; *Authorization in Relational Database Management Systems*; 1st ACM Computer and Communications Security Conference, Fairfax, VA, Nov. 1993

/BeSJ 95/ Bertino, E.; Samarati, P.; Jajodia, S.; *An Extended Authorization Model for Relational Databases*; to appear: IEEE Transactions on Data and Knowledge Engineering, 1995

/Brüg 92/ Brüggemann, H.H.; *Rights in an Object-Oriented Environment*; Jajodia, S.; Landwehr, C. (eds.); Database Security, V: Status and Prospects, Elsevier, IFIP, 1992

/ClWi 87/ Clark, D.D.; Wilson, D.R.; *A Comparison of Commercial and Military Computer Security Policies*; Proc. IEEE Symp. on Security and Privacy, Oakland, Apr. 1987, 184-194

/Fagi 78/ Fagin, R.; *On an Authorisation Mechanism*; ACM Transactions on Database Systems, Vol. 3, No. 3, Sep. 1978, 310-319

/FaSp 91/ Faatz, D.B.; Spooner, D.L.; *Discretionary Access Control in Object-Oriented Engineering Database Systems*; Jajodia, S.; Landwehr, C. (eds.); Database Security, IV: Status and Prospects; Elsevier, IFIP, 1991

/FeGS 89/ Fernandez, E.B.; Gudes, E.; Song, H.; *A Security Model for Object-Oriented Databases*; 1989 IEEE Symp. on Security and Privacy, Oakland, CA, May 1989

/FeSW 81/ Fernandez, E.B.; Summer, R.C.; Wood, Ch.; *Database Security and Integrity*; Addison-Wesley, MA, 1981

/FeWF 94/ Fernandez, E.B.; Wu, J.; Fernandez, M.H.; *User Group Structures in Object-Oriented Database Authorization*; Proc. IFIP WG 11.3 8th Int. Conference on Database Security, Bad Salzdetfurth, Aug. 1994

/GaGF 93/ Gal-Oz, N.; Gudes, E.; Fernandez, E.B.; *A Model of Methods Access Authorization in Object-Oriented Databases*; Proc. of the 17th VLDB, Dublin, Aug. 1993, 52-61

/GrDe 72/ Graham, G.S.; Denning, P.J.; *Protection – Principles and Practice*; AFIPS Spring Joint Computer Conference, AFIPS Press, Montvale, 1972, 417-429

/GrWa 76/ Griffith, P.P.; Wade, B.W.; *An Authorization Mechanism for a Relational Database System*; ACM Transactions on Database Systems, Vol. 1, No. 3, Sep. 1976, 242-255

/GuSF 91/ Gudes, E.; Song, H.; Fernandez, E.B.; *Evaluation of Negative, Predicate, and Instance-based Authorization in Object-Oriented Databases*; Jajodia, S.; Landwehr, C. (eds.); Database Security, IV: Status and Prospects; Elsevier, IFIP, 1991

/HäDi 92/ Härtig, M.; Dittrich, K.R.; *An Object-Oriented Integration Framework for Building Heterogeneous Database Systems*; Proc. of the IFIP DS-5 Conference on Semantics of Interoperable Database Systems, Lorne, Australia, Nov. 1992

/HaRU 76/ Harrison, M.A.; Ruzzo, W.L.; Ullman, J.D.; *Protection in Operating Systems*, Comm. of the ACM, Vol. 19, No. 8, Aug. 1976, 461-471

/HoTe 95/ Holbein, R.; Teufel, S.; *A Security Service for Role Based Access Controls in Distributed Systems*; 11th IFIP TC 11 International Conference on Computer Security SEC'95, South Africa, 1995, 270-285

/HuDT 93/ Hu, M.-Y.; Demurjian, S.A.; Ting, T.C.; *User-Role Based Security Profiles for an Object-Oriented Design Model*; Thuraisingham, B.M.; Landwehr, C.E. (eds.); Database Security, VI: Status and Prospects, Elsevier, 1993, IFIP, 333-348

/HuDT 94/ Hu, M.-Y.; Demurjian, S.A.; Ting, T.C.; *User-Role Based Security in the ADAM Object-Oriented Design and Analysis Environment*; Proc. IFIP WG 11.3 8th Int. Conference on Database Security, Bad Salzdetfurth, Aug. 1994

/JoDi 93a/ Jonscher, D.; Dittrich, K.R.; *Access Control for Database Federations; a discussion of the state-of-the-art*; Proc. DBTA Workshop on Interoperability of Database Systems and Database Applications, Fribourg, Switzerland, Oct. 1993, 156-178

/JoDi 93b/ Jonscher, D.; Dittrich, K.R.; *A Formal Security Model based on an Object-Oriented Data Model*; Technical Report No. 93.41, Institut für Informatik der Universität Zürich, Nov. 1993

/JoDi 94/ Jonscher, D.; Dittrich, K.R.; *An Approach for Building Secure Database Federations*; Proc. 20th VLDB Conference, Santiago, Chile, Sep. 1994, 24-35

/Kent 93/ Kent, W.; *Object Orientation and Interoperability*; NATO Advanced Study Institute, Kusadasi, Turkey, Aug. 1993

/Lamp 71/ Lampson, B.W.; *Protection*; 5th Princeton Symp. on Information Science and Systems, Mar. 1971, 437-443

/LGSF 90/ Larrondo-Petrie, M.M.; Gudes, E.; Song, H.; Fernandez, E.; *Security Policies in Object-Oriented Databases*; Spooner, D.L.; Landwehr, C. (eds.); Database Security, III: Status and Prospects, Elsevier, IFIP, 1990

/Lunt 88/ Lunt, T.; *Access Control Policies: Some Unanswered Questions*; Computer Security Foundations Workshop, Franconia, Jun. 1988

/NiWM 93/ Nicol, J.R.; Wilkes, C.Th.; Manola, F.A.; *Object Orientation in Heterogeneous Distributed Computing Systems*; IEEE Computer, Jun. 1993, 57-67

/NyOs 92/ Nyanchama, G.M.; Osborn, S.L.; *Database Security Issues in Distributed Object Oriented Databases*; Proc. of the Int. Workshop on Distributed Object Management, Edmonton, Canada, Aug. 1992

/NyOs 93/ Nyanchama, M.; Osborn, S.; *Role-Based Security, Object Oriented Databases & Separation of Duty*; SIGMOD RECORD, Vol. 22, No. 4, Dec. 1993, 45-51

/OMG 91/ *The Common Object Request Broker: Architecture and Specification*, Document Number 91.12.1 Revision 1.1, Object Management Group and X Open

/PfHD 88/ Pfefferle, H.; Härtig, M.; Dittrich, K.; *Discretionary Access Control in Structurally Object-Oriented Database Systems*; Proc. IFIP 11.3 Workshop on Database Security, Kingston, Ontario, Canada, Oct. 1988

/RaWK 88/ Rabitti, F.; Woelk, D.; Kim, W.; *A Model of Authorization for Object-Oriented and Semantic Databases*; Proc. of the International Conference on Extending Database Technology, Venice, Italy, Mar. 1988

/RBKW 91/ Rabitti, F.; Bertino, E.; Kim, W.; Woelk, D.; *A Model of Authorization for Next-Generation Database Systems*; ACM Transactions on Database Systems, Vol. 19, No. 1, Mar. 1991, 88-131

/SaCG 91/ Saltor, F.; Castellanos, M.; Garcia-Solaco, M.; *Suitability of data models as canonical models for federated databases*; SIGMOD Record, Vol. 20, No. 4, Dec. 1991

/ShLa 90/ Sheth, A.P.; Larson, J.A.; *Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases*; ACM Computing Surveys, Vol. 22, No. 3, Sep. 1990, 180-236

/Spoo 88/ Spooner, D.L.; *The Impact of Inheritance on Security in Object-Oriented Database Systems*; Proc. IFIP WG 11.3 Workshop on Database Security, Kingston, Ontario, Canada, Oct. 1988

/TiDH 92a/ Ting, T.C.; Demurjian, A.; Hu, M.-Y.; *Requirements, Capabilities, and Functionalities of User-Role Based Security for an Object-Oriented Design Model*; Jajodia, S.; Landwehr, C.E. (eds.); Database Security, V: Status and Prospects, Elsevier, IFIP, 1992

/TiDH 92b/ Ting, T.C.; Demurjian, S.A.; Hu, M.-Y.; *On Information Hiding for Supporting User-Role Based Database Security in the Object-Oriented Paradigm*; Jajodia, S.; Landwehr, C. (eds.); Database Security, V: Status and Prospects, Elsevier, IFIP, 1992

/WaSp 87/ Wang, C.-Y.; Spooner, D.L.; *Access Control in a Heterogeneous Distributed Database Management System*; IEEE 6th Symp. on Reliability in Distributed Software and Database Systems, Williamsburg, VA, Mar. 1987, 84-92