

A Content-Based Authorization Model for Digital Libraries

Nabil R. Adam, Vijay Atluri, *Member, IEEE Computer Society*,
Elisa Bertino, *Senior Member, IEEE*, and Elena Ferrari

Abstract—Digital Libraries (DLs) introduce several challenging requirements with respect to the formulation, specification, and enforcement of adequate data protection policies. Unlike conventional database environments, a DL environment typically is characterized by dynamic user population, often making accesses from remote locations, and by an extraordinarily large amount of multimedia information, stored in a variety of formats. Moreover, in a DL environment, access policies are often specified based on user qualifications and characteristics, rather than user identity (for example, a user can be given access to an R-rated video only if he/she is older than 18 years). Another crucial requirement is the support for content-dependent authorizations on digital library objects (for example, all documents containing discussions on how to operate guns must be made available only to users who are 18 or older). Since traditional authorization models do not adequately meet access control requirements typical to DLs, in this paper, we propose a content-based authorization model suitable for a DL environment. Specifically, the most innovative features of our authorization model are: 1) flexible specification of authorizations based on the qualifications and characteristics of users (including positive and negative), 2) both content-dependent and content-independent access control to digital library objects, and 3) varying granularity of authorization objects ranging from sets of library objects to specific portions of objects.

Index Terms—Digital libraries, access control, authorization.



1 INTRODUCTION

DIGITAL libraries (DLs) are systems that carry out interactions among information and knowledge producers, librarians, and information and knowledge seekers [3]. A DL's goal is to provide the ability on a global scale to acquire, store, and retrieve information electronically. DL systems are typically a collection of distributed, autonomous, and heterogeneous sites. They deal with large amounts of multimedia information where objects may be stored on a variety of formats and media such as disks, tapes, or CD-ROMs and typically come from a variety of sources which may wish to control the DL use (retrieval or modification) or to add value to the content. Users of these systems have a wide variety of diverse background and interests and usually access the DL from remote sites.

One of the problem areas in DLs is securing information. On one hand, the system must be secure against malicious use and data-corruption and must ensure the privacy of its users as well as protect the intellectual property of the vendors and information producers. On the other hand, however, the system should provide open access so that vendors and information producers can add/update information and services any time. Thus, a full-fledged security model for DLs should address authentication and security services to guarantee privacy, integrity, confidentiality, access control, and copyright issues. In this paper, we

focus only on issues related to access control. We plan, however, to gradually extend our security model to cover the above-mentioned aspects.

In conventional database environments, access control is usually performed against a set of authorizations stated by security administrators or users according to some security policies [29]. An authorization, in general, is specified on the basis of three parameters, $\langle s, o, p \rangle$. This triple specifies that user s is authorized to exercise privilege p on object o . Such an approach to authorization specification is not suitable for a DL environment due to its highly dynamic user population and its extraordinarily large collection of objects. In this paper, we propose an authorization model in which access policies can be specified based on user qualifications and characteristics (for example, a user can be given access to an R rated video only if he/she is older than 18 years) and the content of the object or part of an object (for example, all documents containing discussions on how to operate guns must be made available only to users who are 18 or older). Our model enables the enforcement of security policies in conventional libraries, thus making the transition from paper based libraries to their digital counterparts easier. Moreover, our model is able to support articulated access control measures. For example, in traditional libraries it is not always possible to give access only to a part of a book, whereas our access control model will support selective accesses to components of DL objects.

We have implemented our access control model for the Global Legal Information Network (GLIN), a project originally undertaken by the Law Library of Congress (LLoC). In the following, we present an example from GLIN that demonstrates the necessity of new security features for a DL environment:

- N. Adam and V. Atluri are with the CIMIC and MS/IS Department, Rutgers University, 180 University Ave., Newark, NJ 07102. E-mail: adam@adam.rutgers.edu., atluri@andromeda.rutgers.edu.
- E. Bertino and E. Ferrari are with the Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano Via Comelico, 39/41 20135 Milano, Italy. E-mail: {bertino, ferrarie}@dsi.unimi.it.

Manuscript received 1 June 1999; revised 17 July 2000; accepted 27 July 2000.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 109965.

Example 1.1. The GLIN project began in 1993 in response to its responsibility to support the information needs of the United States Congress, the United States Supreme Court, and the executive agencies in the areas of foreign, international, and comparative law. The broad goal of GLIN is to create a knowledge base of international law and to make this knowledge base available to member countries from around the world (about 35 and growing). GLIN emerged from a long-standing experience in manually sorting, arranging, and indexing primary sources of law to gain controlled access to these sources. These are needed for analysis to prepare studies, reports, and memoranda in response to questions submitted to the LLoC. GLIN represents a unique attempt to create an environment where international transactions and interactions are based on trustworthy information and to expanding bonds among member countries that will help fuel growth of common markets and greater cultural ties.

In GLIN, only a certain group of users such as members of official government agencies or the United States Congress can gain access to certain documents. For example, the Law Library of Congress publishes monthly a *World Law Bulletin*, which includes a *blue page report* that provides a report about pending legislation which is a particular topic of interest to Congress. This report should not be public until it is open for public discussion because this could be a gold mine to lobbyists.

Moreover, based on the content of the report, only certain individuals are allowed to view or generate this report or parts of it. For example, the report, "Attorney's fees and Litigation Expenses in Selected Foreign Nations," publishes an overview of attorneys' fees legislation in fifteen countries, which consists of fifteen parts, one for each country. In order to produce a part pertaining to a country, it is required that an individual, such as a legal research analyst, has the national origin of that country.

Although all our examples in this paper are from a GLIN environment, similar access policies can be found in other domains as well (for example, only subscribers to a magazine are allowed to access that magazine).

Since traditional authorization models are not capable of specifying or enforcing such policies, in this paper, we propose a content-based authorization model suitable for a DL environment. Our model allows for the granting of privileges based on user qualifications and characteristics called *credentials* rather than users themselves and based on the concepts associated with objects rather than the objects themselves. Specifically, our authorization model provides

1. flexible specification of authorizations based on the qualifications and characteristics of users (including positive and negative authorizations) rather than just on specific users,
2. access control specification to digital library objects that is based not only on object identifiers but also on objects content, and
3. varying granularity of authorization objects ranging from sets of library objects to specific portions of objects.

The resulting model is, therefore, very flexible and powerful in terms of the kind of protection requirements it can represent. However, the flexibility provided to the user makes access control more complex. The main reason is that users accessing a DL are usually global users (that is, they are not users belonging to the organization owning the DL). Global accesses pose a number of problems, especially when global users are required to provide their credentials for access control purposes. Access control information that is recorded for users at the users' organization may differ with respect to the credentials required by the access control system of the DL. Consider, for instance, the case of a DL requiring the age of the user for giving access to certain objects whereas the user's organization keeps the birth date of the user. In such a case, it is easy to compute the required; however, more complicated situations may arise that require the establishment of some mappings among different sites. Another related question is how does a user (or an organization) wishing to access a DL determine the information to supply for access control purposes. Finally, other important issues include the use of certification and other authentication mechanisms to ensure the authenticity of the information provided for access control purposes.

In this paper, we address all the above issues by first discussing different techniques that can be used for credential mappings. Then, we provide a methodology that allows the user to determine the credential he/she must supply to gain access to a DL document. Finally, we provide a protocol for secure communication of access control information.

The remainder of the paper is organized as follows: Section 2 surveys related work. Section 3 presents an overview of our approach. Section 4 introduces preliminary concepts on which our model relies. Section 5 introduces our authorization model, whereas Section 6 deals with access control. Section 7 describes DLAM (Digital Library Authorization Module), the prototype system implementing our access control model. Section 8 concludes the paper and outlines future work. Due to lack of space, formal proofs are not included in the paper. We refer the interested reader to [1].

2 RELATED WORK

Recently, there have been several extensions made to authorization models adopted in both traditional centralized data management systems [4], [5], [6], [7], [8], [9], [24] and distributed and federated DBMSs [17], [19]. Such models, however, are inadequate for the protection of information in a DL. The main reason is that authorizations are specified in terms of user and object identifiers rather than in terms of user characteristics and object contents. By contrast, since a DL is a new and emerging area of research, there has been very little prior work that addresses security for DLs. Content-based specification of authorizations for DLs is novel and has never been addressed before. However, there has been some preliminary work on specifying authorizations with different levels of granularity and using credentials, which we review below.

Much of the work on security for digital libraries has focused on authentication, copyright, and secure network channel issues rather than authorization. The research and implementation efforts on authorization provide access control based on users and object identifiers rather than user credentials and object concepts, as in our model. Therefore, these approaches do not scale well in large-scale digital library environments. In the following, we briefly present some of the recent DL security efforts. A broker method was implemented in [14], where the broker sends the request to the appropriate authorization system based on the credentials of the users and the services desired by the user. Here, the credentials are simply the user name and password. Later, this work has been extended in [22] by incorporating certificates in user credentials. A hierarchical access control model proposed in [12] organizes digital libraries into hierarchies, such as collections, subcollections, and items, and defines various levels of authorizations such as superuser, collection administrator, and curator. A document protection approach, called Cryptolope, has been presented in [20]. It basically stores the documents in an encrypted form, and the authorized users are given the keys that enable them to decrypt the documents.

One of the first proposals of an access control model for WWW documents has been developed by Samarati et al. [25]; in such a model, documents are organized as unstructured pages connected by links. Authorizations can be given either to the whole document or to selected portions within a document. Although we borrow from [25] the idea of selectively granting access to a document (by authorizing a user to see only some portions and/or links in the document), our work substantially differs from the work in [25]. The major difference is that we support a flexible user specification by which users are qualified by predicates on credentials. By contrast, the model by Samarati et al. does not support such a flexible user specification, but relies only on user-ids, which we believe is not suitable for an open access environment like DLs. Another significant difference is that, in our access control model, authorizations are based on the document contents, whereas the model by Samarati et al. provides access control based on the identifiers of documents only, which again is not suitable for environments like DLs with a very large number of objects. We believe that content-based access control is crucial for DLs.

Winslett et al. [32] have first identified the need to use user credentials for providing access control in a DL environment. The idea is that, in distributed systems, access control cannot be always based on user-ids. Rather, users must provide information, typically about themselves, allowing the access control mechanism to decide whether or not to give access. Building on this idea, we formalize the specification of user credentials by providing a credential specification language. The advantage of using a formal language is that, it not only allows flexible specification of credentials, but also enables easy evaluation of authorization and verification of the consistency of specification. Moreover, our use of credential hierarchy makes the security administration simpler. Another major difference is that we provide an articulated set of features

for enforcing access control ranging from content-based authorizations for nonstructured data, to the support for positive and negative authorizations and varying granularity of authorization objects. Furthermore, we provide a conflict resolution methodology to automatically resolve authorization conflicts and exceptions. In addition, we provide a methodology to compute the information the user should provide to a DL when accessing it from global sites.

Due to the fact that, we organize both concepts and credentials into hierarchies, our model may appear similar to the authorization model proposed for the Orion object-oriented database [24]. The model proposed in [24] is specifically tailored for an object-oriented DBMS storing conventional, structured data, as such, great attention has been devoted to concepts such as versions and composite objects, which are typical of an object-oriented context. Authorizations in Orion are specified based on the $\langle s, o, p \rangle$ paradigm, which is not suitable for a DL environment as has been alluded to earlier. The novelty of our model is its support for a set of features, such as content-based authorizations on unstructured/semistructured textual information and the specification of authorizations based on credentials, which are crucial for a DL environment. To provide such functions, we use some features of the object-oriented paradigm, such as inheritance hierarchies for structuring both concepts and credentials, but this is no more than a modeling tool we use to make authorization specification simpler. Thus, if one were to use Orion for a DL environment, they must incorporate authorization of subjects to objects based on their credential specification and the concept associated with objects with the ability to specify privileges with finer granularity and based on links to other objects. We believe these extensions to Orion would result in a model similar to the one proposed in this paper. In addition, one must address issues arising due to the distributed and heterogeneous nature of DL, including mapping between different credential hierarchies and secure exchange of information among different DL sites.

Content-dependent access control in the context of object-oriented databases have been proposed by Gudes et al. in [15] and by Bertino and Weigand in [10]. However, both proposals deal with conventional data (that is, data whose structure completely fits into a database schema). In such context, content-based access control can be enforced by imposing some conditions on the attribute values. In this context, authorizations rules can be augmented with special *predicates* allowing one to specify conditions on the attributes' values. (For instance, the security policy stating that a user u can only access the information on employees whose salary is less than 30K can be specified using a rule of the form: $\langle u, employees, salary < 30k \rangle$, provided that the database contains a class *employee* with an attribute *salary*). By contrast, due to the nature of DL documents, content-dependent access control in a DL environment deals with the *concepts* a document is related to, rather than with attribute values. Access control enforcement, thus, becomes more difficult. First, it is necessary to have some mechanism in place that extracts concepts from documents. Then, it must be taken into account the relations that may exist among concepts in a given domain (for instance, a concept

may subsume other concepts and so on). These aspects must be considered when enforcing access control. In this paper, we provide a complete architecture for dealing with content-based access control to DL documents, which addresses all the above-mentioned aspects.

The work reported in this paper is also related to work on flexible authorization models [5], [6], [7], [8], [9]. The main goal of such models is to develop authorization models able to accommodate both explicit authorizations and denials, and to support different conflict resolution policies. Even though our model also supports such features, it has a number of relevant differences. First of all, our model supports the notion of credentials as the basis for specifying authorization subjects; by contrast, those other models use user-ids or groups as authorization subjects. Another relevant difference is that our access control model, as any access control model for DLs, must be able to work in open, distributed environments, whereas those access control models have been developed for closed, controlled environments. Also, those models have been developed for traditional, formatted data, under which the various portions of data are well-identified by data model elements, such as relations and attributes in the relational model. By contrast, in our model, we deal with unstructured data, whose semantic contents may not be well-identified. Therefore, we have to base our access control model also on concepts associated with authorization objects, rather than on object names or object component names only. Finally, issues related to credential exchange have not been investigated by such prior work.

Finally, note that the concept of credential has the same similarity with that of role which is the basis for a class of models known as Role-Based Access Control (RBAC) models [23], [28]. Roles can be seen as a set of actions or responsibilities associated with a particular working activity. Under role-based models, all authorizations needed to perform a certain activity are granted to the role associated with that activity, rather than being granted directly to users. Users are then made members of roles, thereby acquiring the roles' authorizations. User access to data is mediated by roles; each user is authorized to play certain roles and, on the basis of the role, he/she can perform accesses on the data. Whenever a user needs to perform a certain activity, the user only needs to be granted the authorization of playing the proper role, rather than being directly assigned the required authorizations. A basic distinction between roles and credentials is that credentials are characterized by a set of attributes, and this allows us to grant access authorizations only to users whose credentials satisfy certain conditions (for instance, access to a document can be granted to all the users with a given age or with a given nationality). This can, of course, be done also through roles but it requires the creation of a distinct role for each condition we would like to enforce (for instance, enforcing the access control policy of the previous example requires the creation of two distinct roles, one corresponding to the users with the specified age, and the other corresponding to the users with the specified nationality). This makes the specification and management of authorizations very difficult, given also the large variety of users that typically access a DL.

3 OVERVIEW OF THE PROPOSED AUTHORIZATION MODEL

Unlike traditional access control models where authorizations are specified as $\langle \text{user}, \text{object}, \text{privilege} \rangle$, in our model, authorizations are specified as $\langle \text{credentials}, \text{concepts}, \text{privilege} \rangle$. Credentials represent the characteristics and qualifications of users. We associate *credentials* with each user. Such credentials are then used to determine the valid authorizations of that user based on *credential expressions*.

A concept in a DL object means the abstractions or notions one expects to find in that object. In other words, a concept is a means for concisely describing the content of a DL object. However, concepts are not just keywords. For example, a combination of words in a document may elicit a sorrowful reaction from a reader although they do not contain any keywords related to "sadness." Often, concepts in a given domain (denoted as \mathcal{CP}) are interrelated, and therefore can be organized into a *conceptual hierarchy*, which is a partial order $\prec_{\mathcal{CP}}$.¹ Given two concepts $cp_1, cp_2 \in \mathcal{CP}$, we say that cp_1 is a more specific concept than cp_2 if and only if $cp_1 \prec_{\mathcal{CP}} cp_2$. Fig. 1 depicts the hierarchy of concepts typically encountered in GLIN. In the figure, there is an arc from concept cp_1 to concept cp_2 if $cp_2 \prec_{\mathcal{CP}} cp_1$. Thus, concepts toward the top of the hierarchy are more general and common to most DL objects, whereas concepts toward the bottom of the hierarchy are more specific. For instance, the concept `Imports Tax` can be regarded as a specialization of the concept `Import-Export`.

The ability to extract concepts and construct a conceptual hierarchy enables us to authorize a user to access a certain DL object only if the object deals with a particular concept or a particular combination of concepts for which the user possesses the appropriate credentials. In this paper, we limit our focus to content-based authorization for textual DL objects. In our work, we use a document management mechanism [16], currently in use by GLIN, which is capable of extracting the concepts and building a conceptual hierarchy. Our goal is to augment this system with content-based access control.

Our model supports *browsing* and *authoring* privileges with various subtypes within each privilege type. These privilege types subsume the conventional privileges such as read and write. Additionally, we provide support for the specification of negative as well as positive authorizations.

Thus, our authorization model provides both content-dependent and content-independent access control to DL objects. Moreover, in our model, authorizations can be specified on the basis of either the user identity or user characteristics.

4 PRELIMINARIES

In this section, we introduce the basic components on which our access control model relies. We first characterize how DL objects are represented in our model. Then, we introduce the concept of *credential*. Finally, we illustrate the access privileges supported by our model.

1. By hierarchy, we mean a poset (S, \prec) , where S is a set and \prec is a partial order over S .

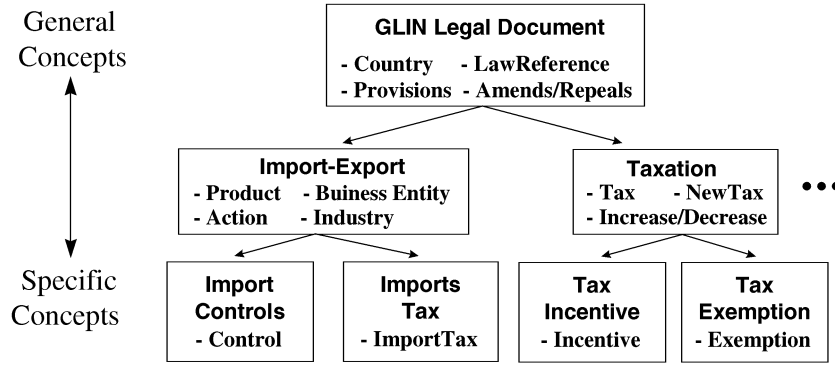


Fig. 1. An example of conceptual hierarchy.

4.1 DL Objects

Each DL document, referred to as *digital library object* (dlo), typically contains unstructured information of different media types (e.g., text, images, videos). We assume that each dlo is associated with a unique object identifier (ID), which is assigned by the system upon the object creation and remains unchanged for the life of the object.²

Often different portions of the same dlo have varying protection requirements. As an example, consider the case of an article from a journal, where its abstract could be made available to everyone, whereas the rest of the article should be made available only upon subscription to the journal. For this reason, we allow the dlo to be divided into *slots*. A slot is a piece of information contained in a document that can be identified by a name. Like DL objects, slots are untyped in that they may contain different types of data. For example, abstract, title, authors, and bibliography are slots that can be associated with a scientific article. Not all slots must be identified by a name. We consider a default unnamed slot as the slot containing the part of the dlo that is not included in any defined slot. If no slot of an object is identified by a name, the unnamed slot will be the entire object.

A dlo can also contain *links* to other related dlo s. The semantics of a link connecting two dlo s is that the contents of the two objects are related. For instance, a dlo representing a scientific article may contain links to publications on the same topic. Each link is represented as a triple $(link_id, source_id, destination_id)$, where $link_id$ is the link identifier and, $source_id$ and $destination_id$ are the IDs of the two objects connected by the link.

In the following, we use \mathcal{OI} , \mathcal{LI} , \mathcal{SN} , and \mathcal{CP} to denote a set of DL object identifiers, a set of link identifiers, a set of slot names, and a set of concepts, respectively. Moreover, given a tuple t with component names c_1, \dots, c_n , we use notation $c_i(t)$ to denote the value of component c_i , $i = 1, \dots, n$.

A dlo can therefore be represented as a 4-tuple $(i, slots, links, concepts)$, where $i \in \mathcal{OI}$ is the ID of the dlo , $slots \in 2^{\mathcal{SN}}$ is a set of slot names identifying relevant portions within the dlo , $links \in 2^{\mathcal{LI}}$ is a set of link identifiers, and $concepts \in 2^{\mathcal{CP}}$ is the set of relevant concepts in the dlo . The *Object Base*, denoted by \mathcal{OB} , is the set of all

dlo s. Given a $dlo = (i, slots, links, concepts)$, we use $C(dlo)$ to denote the set of concepts characterizing dlo . This set of concepts is a union of those in $concepts$ and all the concepts more general than those in $concepts$. Formally, $C(dlo) = concepts \cup \{cp \mid cp \in \mathcal{CP} \text{ such that } \exists cp' \in concepts, \text{ with } cp' \prec_{\mathcal{CP}} cp\}$. Note that the definition of $C(dlo)$ is directly implied by the semantics of the $\prec_{\mathcal{CP}}$ relationship in that, if the content of an object is described by a given concept, then it is also described by all the (more general) concepts which precede it in the conceptual hierarchy.

Access authorizations can be given either on specific objects (by listing their IDs) or on all the objects containing a particular concept (or a particular combination of concepts). Moreover, a finer-grained access control can be enforced by authorizing users to access only specific slots and/or links within an object.

All these possibilities are summarized by the notion of *entity specification*. Before formally introducing the notion of entity specification, we need to introduce the following definitions:

Definition 4.1 (Conceptual Expression). *The set $COEX$ of conceptual expressions is defined as follows:*

- each element of \mathcal{CP} is a conceptual expression,
- if ce_1 and ce_2 are conceptual expressions, then $(ce_1 \wedge ce_2)$ and $(ce_1 \vee ce_2)$ are conceptual expressions.

Definition 4.2 (Entity Specification). *An entity specification has one of the following two forms:*

1. $co_spec.slot_spec$, where co_spec is either a conceptual expression in $COEX$, or a set, possibly empty, of object identifiers in $2^{\mathcal{OI}}$, and $slot_spec \in 2^{\mathcal{SN}}$ is a set, possibly empty, of slot names, or
2. $link_spec$, where $link_spec \in 2^{\mathcal{LI}}$ is a set, possibly empty, of link identifiers.

The entity specification $co_spec.slot_spec$ denotes the slots whose names are in $slot_spec$ of the objects denoted by co_spec . If $slot_spec = \emptyset$, the specification denotes all objects denoted by co_spec , whereas if $co_spec = \emptyset$, it denotes all slots whose names are in $slot_spec$, regardless of the objects in which they appear. Similarly, $link_spec$ denotes the links whose identifiers are in $link_spec$.

2. We do not make any assumption on how IDs are generated.

In practice, *co-spec*, *slot-spec*, *link-spec* are omitted when they are empty, whereas they are represented by their unique element when they are a singleton. Moreover, given an entity specification *ent-spec*, we use *co-spec(ent-spec)* to denote the set of object identifiers or the conceptual expression in *ent-spec*, whereas *slot-spec(ent-spec)* (respectively, *link-spec(ent-spec)*) denotes the set of slot names (respectively, link identifiers) in *ent-spec*.

Example 4.1. World Law Bulletin.Blue page report, Imports Tax \vee Tax Incentive, Attorney's fees and Litigation Expenses in Selected Foreign Nations.Italian part are examples of entity specifications.

Given a conceptual expression *cx*, *Concepts(cx)* indicates the set of concepts appearing in *cx*, whereas *Obj(cx)* denotes the set of IDs of all the *dlos* whose associated concepts (that is, those in *C(dlo)*) satisfy *cx*.

4.2 Credentials

To allow for the specification of authorizations based not only on the user identity but also on the user characteristics, each user is associated with one or more *credentials*. A credential is a set of user attributes that are needed for security purposes. Credentials are assigned when a user is created and are updated automatically according to the user's profile. To make the task of credential specifications easier, credentials with similar structures are grouped into *credential-types*.

To formalize the concepts of credentials and credential-types, we use \mathcal{AN} to denote a set of attribute names, \mathcal{T} to denote the set of possible types of attributes in \mathcal{AN} , and \mathcal{V} to denote the set of legal values for types in \mathcal{T} . \mathcal{T} includes both basic types (such as integer, real, boolean, character, and string) and structured types, built by applying the constructors, sets, lists, and records to basic as well as to structured types. Moreover, we denote the set of credential-type identifiers with \mathcal{CT} and the set of credential identifiers with \mathcal{CI} . We use \mathcal{U} to denote a set of account identifiers, associated with the users authorized to access the system. Credential-types are formally defined as follows:

Definition 4.3 (Credential-Type). A credential-type is a pair $(ct_id, attr)$, where $ct_id \in \mathcal{CT}$ is the credential-type identifier and $attr$ is a set containing an item for each attribute of the credential type. $attr$ in turn is a triple (a_name, a_dom, a_type) , where $a_name \in \mathcal{AN}$ is the attribute name (the attribute names must be distinct), $a_dom \in \mathcal{T}$ is the attribute domain, and $a_type \in \{\text{opt}, \text{mand}\}$ indicates whether the attribute can assume a null value ($a_type = \text{"opt"}$) or not ($a_type = \text{"mand"}$).

Example 4.2. The following is an example of credential-type:

```
(employee, {-(age, string, opt), (address,
string, mand), (salary, integer, opt),
(nationality, string, mand), (national
origin, string, mand)}).
```

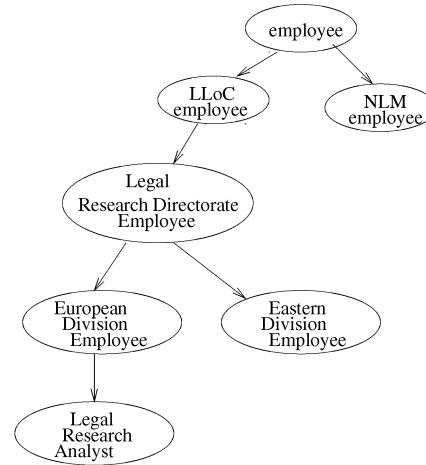


Fig. 2 An example of credential-type hierarchy.

Credential types in \mathcal{CT} are organized into a hierarchy, referred to as *credential-type hierarchy*, according to a partial order $\prec_{\mathcal{CT}}$. Given two credential types ct_1 and $ct_2 \in \mathcal{CT}$, we say that ct_2 is a *subcredential type* (or shortly subtype) of ct_1 if and only if $ct_2 \prec_{\mathcal{CT}} ct_1$. Multiple inheritance is not supported, that is, we do not allow a credential-type to be a subtype of more than one credential-type. Each credential-type contains all the attributes of the credential-types preceding it in the hierarchy. Moreover, it can contain additional attributes, apart from the inherited ones. Given a credential-type $ct \in \mathcal{CT}$, $A(ct)$ denotes the set of attributes of the credential-type instances.

Example 4.3. Consider the credential-type *employee* of Example 4.2 and suppose that a new credential-type *legal research analyst*, subtype of *employee*, is created with an additional attribute *project*. Thus, $A(\text{legal research analyst}) = A(\text{employee}) \cup \{\text{project}\} = \{\text{age, address, salary, project, nationality, national origin}\}$, since *legal research analyst* inherits all the attributes of the credential-type *employee*.

An example of credential-type hierarchy is reported in Fig. 2. In the figure, there is an arc from a credential-type ct_1 to a credential-type ct_2 if $ct_2 \prec_{\mathcal{CT}} ct_1$.

Note that the existence of a common root of the entire credential-type hierarchy is not assumed. This means that the Security Administrator can structure the credential-type hierarchy into multiple direct trees with a distinguished root. Roots are classes without superclasses. However, we assume the existence of a built-in credential-type, denoted $\top_{\mathcal{CT}}$, such that $ct \prec_{\mathcal{CT}} \top_{\mathcal{CT}}$, for all $ct \in \mathcal{CT} \setminus \{\top_{\mathcal{CT}}\}$. This predicate, which is automatically inserted by the system upon the creation of the credential type hierarchy, is used in the evaluation of authorizations not tagged to a specific credential-type, like, for instance, an authorization for all the users older than 18.

A credential is an instance of a credential-type ct and is, in turn, a member of all credential-types ct' , such that $ct \prec_{\mathcal{CT}} ct'$. (Here, we use the terminology adopted by object-oriented data models). A credential can be formally defined as follows:

Definition 4.4 (Credential). A credential c is a 4-tuple $(c.id, user.id, state, ct.id)$, where $c.id \in CT$ is the credential identifier, $user.id \in U$ is the identifier of the user with whom the credential is associated, $state = (a_1 : v_1, \dots, a_n : v_n)$, where $a_1, \dots, a_n \in A(ct.id)$ are the names of the attributes of c , $v_1, \dots, v_n \in \mathcal{V}$ are their values, and $ct.id \in CT$ is the identifier of the credential-type of which c is an instance.

The *Credential Base*, denoted CB , is the set of credentials associated with users in the system. Given a user u , $CT(u)$ denotes the set of credential types characterizing user u . This set is composed by the credential types of which the credentials associated with u are either members or instances. Formally, let $CT' = \{ct \mid \exists(c.id, u', state, ct') \in CB \text{ such that } u = u' \text{ and } ct = ct'\}$. Then, $CT(u) = CT' \cup \{ct \mid ct \in CT \text{ and } \exists ct' \in CT' \text{ with } ct' \prec_{CT} ct\}$.

Example 4.4. The following are examples of credentials:

- $(c_1, \text{Bob}, (\text{age}:\text{null}, \text{address}:\text{Queen Street}, \text{salary}:2,000, \text{nationality}:\text{US}, \text{national origin}:\text{Italy}), \text{employee})$;
- $(c_2, \text{Ann}, (\text{age}:29, \text{address}:\text{Broad Street}, \text{salary}:\text{null}, \text{project}:\text{P125}, \text{nationality}:\text{US}, \text{national origin}:\text{US}), \text{legal research analyst})$.

In our model, authorizations can be given explicitly to users, by specifying their identifiers, or implicitly by imposing a set of conditions that the credential must satisfy. For instance, it is possible to specify that a particular object is accessible only to users whose age is greater than 18, or only to the legal research analysts working on a particular project. Conditions on credentials are expressed by means of the specification language we present in the next section.

4.2.1 The Credential Constraint Specification Language

The language we provide to express constraints on credentials consists of the following components:

- a set of variables Var_U , ranging over the set U of user identifiers;
- a set of predicate symbols $Pred$ of arity one, with type Var_U . For each credential-type $ct \in CT$, a corresponding predicate symbol $ct()$ is defined in $Pred$. Intuitively, predicates in $Pred$ are used to capture the associations of users with credential types.

Expressions that can be specified in our language, referred in the following as *credential expressions*, are formally defined as follows:

Definition 4.5 (Credential Expression). The set \mathcal{CE} of credential expressions is defined as follows:

- if $X \in Var_U$, $p \in Pred$, then $p(X)$ is a credential expression;
- if $X \in Var_U$, $a \in AN$, $v \in \mathcal{V}$, and $OP \in \{=, \neq, >, <, \geq, \leq, \in, \notin, \subseteq, \supseteq, \subset, \supset, \ni, \not\ni\}$, then $X.a OP v$ is a credential expression;³

3. Note that, the operators that can be used in an expression of the form $X.a OP v$ are actually a subset of the ones listed above since they depend on the type of the attribute $X.a$. We assume that there is some mechanism in place that verifies that only the meaningful operators are used.

- if ce_1 and ce_2 are credential expressions, then $(ce_1 \wedge ce_2)$ and $(ce_1 \vee ce_2)$ are credential expressions.

If a credential expression does not contain any credential-type, then its variables are implicitly bound to \top_{CT} , that is, to the most general credential-type. Given a credential expression ce , $Ctypes(ce)$ denotes the set of credential types in ce . If ce does not contain any credential type, then $Ctypes(ce) = \{\top_{CT}\}$.

Example 4.5. The following are examples of credential expressions in \mathcal{CE} :

- $\text{legal research analyst}(X)$. Intuitively, this expression denotes the users that have a credential of type $\text{legal research analyst}$ associated with them.
- $(X.\text{project} \neq p_1)$. This expression denotes the users not involved in project p_1 .
- $(\text{employee}(X) \wedge X.\text{age} > 18)$. This expression denotes the employees whose age is greater than 18;
- $(\text{Eastern division employee}(X) \vee \text{NLM employee}(X))$. This expression denotes the users with a credential of type $\text{Eastern division employee}$ or NLM employee .

4.2.2 Credential Specification

The users to which an authorization applies can thus be identified by means of a *credential specification*, defined as follows:

Definition 4.6 (Credential Specification). A credential specification is either a set of user identifiers in 2^U , or a credential expression in \mathcal{CE} .

Each credential specification denotes a set, possibly empty, of users. For access control purposes, it is also important to identify the set of users that do not satisfy a given credential specification because of null values in their credentials.⁴ For instance, let $ce = X.\text{salary} > 2,000$. Suppose there exist two users u_1 and u_2 such that the value of attribute salary in u_1 's credential is 1,000, whereas the value of attribute salary in u_2 's credential is null. Thus, both u_1 and u_2 do not belong to the set of users identified by ce even if the value of u_2 's salary could be greater than 2,000. To keep track of these users, we introduce two functions, *Denotes* and *Undef* that, given a credential expression ce , respectively return the set of users identified by ce and the set of users that do not satisfy ce because of null values in their credentials. Table 1 lists, for each different type of credential expression that can be formulated in our language, the value that functions *Denotes* and *Undef* assume. With reference to the table, ce_1 and ce_2 are elements in \mathcal{CE} , $p \in Pred$, $X \in Var_U$, $a \in AN$, $OP \in \{=, \neq, >, <, \geq, \leq, \in, \notin, \subseteq, \supseteq, \subset, \supset, \ni, \not\ni\}$, $v' \in \mathcal{V}$, $a(v)$ denotes the value of an attribute a .

Note that, according to the conditions listed in Table 1, the expression $p(X)$, where $p \in Pred$, denotes the users that have an associated credential whose type is either p or a subtype of p . For instance, with reference to Fig. 2, the expression

4. We will elaborate on this in Section 5.

TABLE 1
Functions *Denotes()* and *Undef()*

Credential Expression	Denotes()	Undef()
$p(X)$	$\{u \mid (c_id, user_id, state, ct_id) \in \mathcal{CB}$ $user_id = u \text{ and } (ct_id = p \text{ or } ct_id \prec_{cT} p)\}$	\emptyset
$X.a \text{ OP } v'$	$\{u \mid (c_id, user_id, state, ct_id) \in \mathcal{CB}$ $user_id = u, a \in A(ct_id), a(v) \text{ OP } v' \text{ holds}\}$	$\{u \mid (c_id, user_id, state, ct_id) \in \mathcal{CB}$ $user_id = u, a \in A(ct_id), a(v) = null\}$
$ce_1 \wedge ce_2$	$Denotes(ce_1) \cap Denotes(ce_2)$	$Undef(ce_1) \cup Undef(ce_2)$
$ce_1 \vee ce_2$	$Denotes(ce_1) \cup Denotes(ce_2)$	$Undef(ce_1) \cap Undef(ce_2)$
$\neg ce_1$	$U \setminus Denotes(ce_1)$	$Undef(ce_1)$

(European division employee(X) \wedge $X.age > 18$), denotes all the users whose security type is either European division employee or legal research analyst and whose age is greater than 18.

Example 4.6. Let $\mathcal{CB} = \{c_1, c_2\}$, where c_1, c_2 are the credentials in Example 4.4. Let

$$U = \{\text{Ann}, \text{Bob}\}. Denotes(\text{employee}(X)) = \{\text{Bob}, \text{Ann}\};$$

$$Denotes(X.age > 18) = \{\text{Ann}\};$$

$$Denotes((\text{employee}(X) \wedge \text{salary} \geq 2,000)) = \{\text{Bob}\};$$

$$Undef(\text{employee}(X)) = \emptyset;$$

$$Undef(X.age > 18) = \{\text{Bob}\};$$

$$Undef((\text{employee}(X) \wedge \text{salary} \geq 2,000)) = \{\text{Ann}\}.$$

4.3 Privileges

The set P of privileges supported by our model is divided into two groups: *browsing privileges* and *authoring privileges*.

Browsing privileges allow users to see the information in an object (or in some of its slots) and/or to see the existence of a link in an object. Therefore, three different types of browsing privileges are supported: *view*, *link*, and *view-all*. The *view* privilege allows a user to see the information in an object or in some of its slots. The *link* privilege authorizes a user to see the existence of a specific link or of all the links in a given object. Clearly, for a link authorization to be applicable, the user needs to be authorized to access the object source of the link, that is, the user must hold a *view* authorization on the object. Finally, the *view-all* privilege subsumes both the *link* and the *view* privilege since it allows users to see the information in an object and all the links it contains. Thus, the *view-all* privilege on an object is equivalent to the *view* privilege on the object and the *link* privilege on all the links contained in the object.

Note that we have chosen to distinguish between *link* and *view* privileges because such a difference makes it possible to grant users access to the information in an object (or in some of its slots) without disclosing the relationships of this object to other objects. For example, to facilitate the blind reviewing of research papers, the reviewers should be given access to the papers without revealing possible links to publications by the same author(s) of the paper being reviewed.

Authoring privileges allow users to modify the content of an object or create a link. We distinguish among the following authoring privileges: *refer*, to include a link in

an object or in a slot; *append*, to write information in an object (or in some of its slots) without deleting any preexisting information,⁵ *update*, to modify the content of an object (or of some of its slots) and to include links in the object. Thus, the *update* privilege subsumes both the *refer* and the *append* privilege. Moreover, we assume that if a user has the *update* privilege on an object, then he/she can also delete the object. Note that a user holding the *refer* or the *update* privilege on an object can specify links from the object to any other object. No authorization is needed on an object to specify links with that object as destination of the link. Elements of P are, therefore, related in that some privileges are subsumed by others (for instance, *view* and *link* are subsumed by *view-all*). For this reason, also the set of privileges P is organized into a hierarchy, called *privilege hierarchy*, by means of a partial order \prec_P . We say that a privilege p_j is a *subprivilege* of privilege p_i if and only if the relation $p_i \prec_P p_j$ holds. This means that privilege p_j is subsumed by privilege p_i .

Table 2 summarizes the privileges supported by our model and their interactions.

5 THE AUTHORIZATION MODEL

In this section, we illustrate the basic components of our authorization model. We first illustrate how authorizations can be specified in our model, then we discuss how to deal with authorization conflicts.

5.1 Access Authorizations

In our model, both positive and negative authorizations can be specified. Positive authorizations indicate access permissions, whereas negative authorizations indicate access denials. Authorizations apply to concepts, objects, slots within objects, as well as to links. Access authorizations are formally defined as follows:

Definition 5.1 (Authorization). *An authorization is a 4-tuple $(crd\text{-}spec, ent\text{-}spec, priv, sign)$, where $crd\text{-}spec$ is a credential specification denoting the users to whom the authorization is granted, $ent\text{-}spec$ is an entity specification denoting the contents, objects, and/or the slots or links to which the authorization refers, $priv \in P$ is the privilege for which the authorization is granted, and $sign \in \{+, -\}$ indicates whether the authorization is positive (+) or negative (-).*

5. The *append* privilege can be used to authorize users to annotate a given object without modifying its content.

TABLE 2
Authorization Privileges and Their Semantics

Type	Privilege	Meaning
Browsing	view	to see the information in an object (or in some of its slots). Links are not displayed
	link	to see the existence of links
	view-all	to see the information in an object and all the links it contains. It is equivalent to the view privilege on the object and the link privilege on all the object links
Authoring	refer	to include a link in an object (or in some of its slots)
	append	to write information in an object (or in some of its slots) without deleting any pre-existing information
	update	to modify (or delete) the content of an object. It subsumes both the refer and the append privilege.

Tuple $(\text{crd-spec}, \text{ent-spec}, \text{priv}, +)$ states that the users denoted by crd-spec have been granted the privilege priv on the objects and/or slots or links denoted by ent-spec . Tuple $(\text{crd-spec}, \text{ent-spec}, \text{priv}, -)$ states that the users denoted by crd-spec have been forbidden privilege priv on the objects (slots or links) denoted by ent-spec . Since the **view-all** privilege allows one to see all information in an object, authorizations with $\text{priv} = \text{view-all}$ have necessarily ent-spec of the form co-spec.slot-spec , with $\text{slot-spec} = \emptyset$ (cfr. Definition 4.2). Moreover, authorizations with $\text{priv} = \text{view}$ have necessarily ent-spec of the form co-spec.slot-spec , with slot-spec possibly empty, whereas authorizations with $\text{priv} = \text{link}$ have ent-spec of the form co-spec.slot-spec with $\text{slot-spec} = \emptyset$, or of the form link-spec . Finally, authorizations for an authoring privilege have ent-spec of the form co-spec.slot-spec , with slot-spec possibly empty. All the above conditions are summarized by Table 3.

Example 5.1. The following are examples of authorizations:

- $A_1 = (\text{LLOC employee}(X), \text{World Law Bulletin.Blue page report}, \text{view-all}, -)$ prevents all the users with an associated credential of type **LLOC employee** from seeing the information in the **Blue page report** of the **World Law Bulletin**.
- $A_2 = ((\text{NML employee}(X) \vee \text{legal research directorate employee}(X)), (\text{Imports Tax} \vee \text{Tax Incentive}), \text{view}, +)$ authorizes users with an associated credential of type **NML employee** or **legal research directorate employee** to

see the information in all the objects containing the concept **Imports Tax** or the concept **Tax Incentive**, but not the links contained in such objects.

- $A_3 = (\text{NMLEmployee}(X), \text{Import Controls}, \text{link}, -)$ prevents **NML employees** from seeing links in objects containing the concept **Import Controls**.
- $A_4 = ((\text{legal research analyst}(X) \wedge X.\text{national origin} = \text{Italy}), \text{Attorney's fees and Litigation Expenses in Selected Foreign Nations.Italian part}, \text{update}, +)$ authorizes all the legal research analysts, whose country of origin is **Italy**, to modify the part pertaining to **Italy** of the report **Attorney's fees and Litigation Expenses in Selected Foreign Nations**.

In the following, the term *Authorization Base* (AB) is used to denote a set of authorizations. Moreover, given an authorization $A = (\text{crd-spec}, \text{ent-spec}, \text{priv}, \text{sign})$, we denote with $\text{crd-spec}(A)$, $\text{ent-spec}(A)$, $\text{priv}(A)$, and $\text{sign}(A)$ the credential specification, the entity specification, the privilege, and the sign of A , respectively.

According to Definition 5.1, a single authorization may be used to authorize a set of users to exercise the same privilege on the same objects, slots, or links. These users can be explicitly specified by listing their identifiers, or implicitly denoted by means of a credential expression. In order to enforce access control, it is necessary to identify, given an authorization, the set of users to which it applies. A critical issue is represented by the possible presence of null values in the credentials associated with users. Indeed, suppose that the credential expression $\text{legal research analyst}(X) \wedge X.\text{age} > 18$, identifying all the users with an associated credential of type **legal research analyst** and whose age is greater than 18, appears in an authorization A . Consider a user u , with an associated credential of type **legal research analyst**, such that no value is specified for attribute **age** in his/her credential. The problem arises in determining whether authorization A applies to user u since we have no information on his/her age. In our model, we take the most conservative approach: If A is positive, then A does not apply to user u ; by contrast if A is negative then it holds for user u also.

TABLE 3
Authorization Privileges and Entity Specifications

Privilege	Entity Specification
view-all	co-spec.slot-spec with $\text{slot-spec} = \emptyset$
view	co-spec.slot-spec
link	co-spec.slot-spec with $\text{slot-spec} = \emptyset$ or link-spec
authoring	co-spec.slot-spec

These concepts are formalized by the notion of *denoted users* which is the set of users to which an authorization applies. This set is defined by using functions $Denotes()$ and $Undef()$, introduced in Section 4.2.2.

Definition 5.2 (Denoted Users). *Let A be an authorization. The set of users to which A applies, denoted as U_A , is defined as follows:*

- if $crd-spec(A) \in 2^U$, then $U_A = crd-spec(A)$;
- if $crd-spec(A) \in \mathcal{CE}$, then:
 - if $sign(A) = "+"$, then $U_A = Denotes(crd-spec(A))$;
 - if $sign(A) = "-"$, then

$$U_A = Denotes(crd-spec(A)) \cup Undef(crd-spec(A)).$$

Similarly, we denote with *denoted objects*, the set of IDs of objects to which an authorization applies. This set is formally defined as follows:

Definition 5.3 (Denoted Objects). *Let A be an authorization. The set of IDs of objects to which A applies, denoted as O_A , is defined as follows:*

- if $co-spec(ent-spec(A)) \in 2^{OI}$, then $O_A = co-spec(ent-spec(A))$;
- if $co-spec(ent-spec(A))$ is a conceptual expression, then $O_A = Obj(co-spec(ent-spec(A)))$;
- if $co-spec(ent-spec(A)) = \emptyset$, then:
 1. if $slot-spec(ent-spec(A)) \neq \emptyset$, then $O_A = \{i' \mid (i, slots, links, concepts) \in \mathcal{OB}, i = i' \text{ and } slots \cap slot-spec(ent-spec(A)) \neq \emptyset\}$;
 2. if $link-spec(ent-spec(A)) \neq \emptyset$, then $O_A = \{i' \mid (i, slots, links, concepts) \in \mathcal{OB}, i = i' \text{ and } links \cap link-spec(ent-spec(A)) \neq \emptyset\}$.

Example 5.2. Consider the authorizations of Example 5.1.

Let $\mathcal{CB} = \{c_1, c_2\}$, where c_1, c_2 are the credentials of Example 4.4. $U_{A_1} = \{Ann\}$, $U_{A_2} = \{Ann\}$, $U_{A_3} = U_{A_1} = \emptyset$, $O_{A_1} = \{World\ Law\ Bulletin\}$, $O_{A_2} = \{Attorney's\ fees\ and\ Litigation\ Expenses\ in\ Selected\ Foreign\ Nations\}$.

5.2 Authorization Propagation and Conflict Resolution

The fact that concepts, credential types, and privileges are hierarchically structured impacts the authorizations a user possesses, in that authorizations (both positive and negative) propagate along these hierarchies according to a set of predefined rules. As an example, suppose that a negative authorization is given to all users with an associated credential of type *employee* and whose age is greater than 18. Then, it is reasonable to assume that this denial propagates to all the legal research analyst older than 18, since the credential type *legal research analyst* is a specialization of *employee*. Therefore, in our model, an authorization given to all the users with a credential of a given type propagates to all the users whose credential is of a subtype of the credential-type which appears in the authorization.

Similarly, authorizations propagate along the conceptual hierarchy, in that an authorization given on the objects containing a given concept c implies an analogous authorization on all objects containing a concept more specific than c . For instance, if a user has an authorization to access all objects related to the concept *Taxation*, then he/she can also access all the objects related to the concept *Tax Exemption*.

Finally, an authorization for a privilege p implies an analogous authorization for all the privileges subsumed by p , that is, for all the privileges p' such that $p' \prec_p p$.

Example 5.3. Consider the authorizations of Example 5.1 and the credential-type hierarchy of Fig. 2. Authorization A_1 implies an analogous authorization for all the users with a credential of type *legal research directorate employee*, *European division employee*, *eastern division employee* and *legal research analyst*.

Note that authorization propagation is very useful since it is a means of concisely expressing a set of related authorizations. However, our model allows one to specify exceptions with respect to how authorizations propagate along the hierarchies by issuing proper negative authorizations. Although negative authorizations greatly augment the expressive power of the model, they have the drawback of introducing potential conflicts since a user may simultaneously hold both a negative and a positive authorization for the same privilege on the same object, slot, or link. However, we do not consider the simultaneous presence of conflicting authorizations as an inconsistency, rather, we define a conflict resolution policy to deal with conflicting authorizations. Our conflict resolution policy keeps into account the conceptual, credential type, and privilege hierarchy, and is based on the concept of *most specific authorization* (first introduced in [21]). The idea is that authorizations specified on lower-level elements of the hierarchies prevail over authorizations specified on upper-level elements. Moreover, since our model supports more than one hierarchy, our conflict resolution policy establishes a priority order among the hierarchies, used when dealing with conflicting authorizations. More precisely, our conflict resolution policy is based on the following principles:

1. Authorizations explicitly given to users (that is, authorizations in which the user identifiers explicitly appear) have the highest priority.
2. In all the other cases, the most specific authorizations, with respect to the credential type hierarchy, are considered as prevailing.
3. Otherwise, the most specific authorizations, with respect to the conceptual hierarchy, are considered as prevailing.
4. When conflicts are not solved by the above rules, we consider the privilege hierarchy in that the authorizations specified for the most specific privileges are considered as prevailing.
5. Finally, when conflicts are not solved by the principles above, negative authorizations take precedence.

Example 5.4. Consider the credential-type hierarchy in Fig. 2, the conceptual hierarchy in Fig. 1 and the authorizations of Example 5.1:

- A_2 prevails over $(\text{employee}(X), (\text{Imports Tax} \vee \text{Tax Incentive}), \text{view}, -)$ since $\text{NML employee}(X) \prec_{CT} \text{employee}$, and $\text{legal research directorate employee} \prec_{CT} \text{employee}$.
- A_1 prevails over $(\text{LLOC employee}(X), \text{World Law Bulletin.Blue page report}, \text{view-all}, +)$, in that the two authorizations differ only for the sign. In such case, the most conservative approach is taken and, thus, we consider the negative authorization as prevailing (see point 5 above).
- A_2 prevails over $(\text{NML employee}, \text{Import-Export} \vee \text{Taxation}, \text{view}, -)$, since $\text{Tax Incentive} \prec_{CP} \text{Taxation}$ and $\text{Imports Tax} \prec_{CP} \text{Import-Export}$.

Our conflict resolution policy is formalized by the concept of *stronger authorizations*. Consider two authorizations, A_1 and A_2 , and a privilege p such that $\text{priv}(A_i) = p$ or $p \prec_P \text{priv}(A_i)$, $i = 1, 2$. Suppose there exists a user u and an object dlo such that $u \in U_{A_1} \cap U_{A_2}$ and $i(dlo) \in O_{A_1} \cap O_{A_2}$. Intuitively, A_1 is stronger than A_2 with regard to u and dlo , if whenever u requires the exercising of privilege p on object dlo , authorization A_1 prevails over authorization A_2 , according to our conflict resolution policy.

Before formally defining the notion of stronger authorization, we need to introduce some preliminary definitions.

Definition 5.4 (Stronger Credential Specification). Let u be a user. Let A_1 and A_2 be two authorizations such that $u \in U_{A_1} \cap U_{A_2}$. $\text{crd-spec}(A_1)$ is stronger than $\text{crd-spec}(A_2)$ with regard to user u , written $\text{crd-spec}(A_1) >_u \text{crd-spec}(A_2)$, iff one of the following conditions holds:

1. $\text{crd-spec}(A_1) \in 2^U$ and $\text{crd-spec}(A_2) \in \mathcal{CE}$.
2. $\text{crd-spec}(A_1), \text{crd-spec}(A_2) \in \mathcal{CE}$ and $\forall ct_2 \in C.\text{types}(\text{crd-spec}(A_2)) \cap CT(u) \exists ct_1 \in C.\text{types}(\text{crd-spec}(A_1)) \cap CT(u)$ such that $ct_1 \prec_{CT} ct_2$.

Example 5.5. Let u be a user with a credential of type LLOC employee and with age greater than 18. Thus, according to the credential-type hierarchy in Fig. 2, $CT(u) = \{\text{LLOC employee}, \text{employee}, \top_{CT}\}$. Hence,

- $\text{LLOC employee}(X) >_u (\text{employee}(X) \wedge X.\text{age} > 18)$.
- $\text{LLOC employee}(X) >_u X.\text{age} > 18$.
- $\{u_1, u\} >_u \text{LLOC employee}(X)$.

Definition 5.5 (Stronger Entity Specification). Let dlo be an object. Let A_1 and A_2 be two authorizations such that $i(dlo) \in O_{A_1} \cap O_{A_2}$. $\text{ent-spec}(A_1)$ is stronger than $\text{ent-spec}(A_2)$ with regard to object dlo , written $\text{ent-spec}(A_1) >_{dlo} \text{ent-spec}(A_2)$ iff one of the following conditions holds:

1. $\text{co-spec}(\text{ent-spec}(A_1)) \in 2^{CT}$, $\text{co-spec}(\text{ent-spec}(A_2)) \in 2^{CT}$, $\text{ent-spec}(A_1)$ and $\text{ent-spec}(A_2)$ are of the form $\text{ent-spec.slot-spec}$ and $\text{slot-spec}(\text{ent-spec}(A_1)) \neq \emptyset$, whereas $\text{slot-spec}(\text{ent-spec}(A_2)) = \emptyset$.
2. $\text{co-spec}(\text{ent-spec}(A_1)) \in 2^{CT}$ and $\text{co-spec}(\text{ent-spec}(A_2))$ is a conceptual expression.

co-spec	slot-spec	
object identifiers	slot names	↓
object identifiers	-	
boolean expression of concepts	slot names	
boolean expression of concepts	-	

Fig. 3. Stronger entity specification.

3. $\text{co-spec}(\text{ent-spec}(A_1))$ and $\text{co-spec}(\text{ent-spec}(A_2))$ are conceptual expressions and $\forall cp_2 \in \text{Concepts}(\text{co-spec}(\text{ent-spec}(A_2))) \cap C(dlo) \exists cp_1 \in \text{Concepts}(\text{co-spec}(\text{ent-spec}(A_1))) \cap C(dlo)$ such that $cp_1 \prec_{CP} cp_2$.
4. $\text{co-spec}(\text{ent-spec}(A_1))$ and $\text{co-spec}(\text{ent-spec}(A_2))$ are conceptual expressions such that the condition in point 3 does not hold, and $\text{slot-spec}(\text{ent-spec}(A_1)) \neq \emptyset$, whereas $\text{slot-spec}(\text{ent-spec}(A_2)) = \emptyset$.
5. $\text{ent-spec}(A_1)$ is of the form link-spec , whereas $\text{ent-spec}(A_2)$ is of the form co-spec.slot-spec , with $\text{slot-spec} = \emptyset$.

All the above conditions are graphically summarized by Fig. 3. For the sake of simplicity, in the figure, we consider only entity specifications of the form co-spec.slot-spec . With reference to the figure, the entity specifications toward the top of the figure are stronger than the entity specifications towards the bottom.

Example 5.6. Consider the conceptual hierarchy of Fig. 1. Let dlo be an object and let $es_1 = (\text{Tax Exemption} \vee \text{Import-Export})$ and $es_2 = (\text{Taxation} \vee \text{Import Controls})$ be two entity specifications.

- If $\text{concepts}(dlo) = \{\text{Tax Exemption}\}$, then $C(dlo) = \{\text{Tax Exemption}, \text{Taxation}, \text{GLIN Legal Document}\}$, $\text{Concepts}(es_1) = \{\text{Tax Exemption}, \text{Import-Export}\}$, $\text{Concepts}(es_2) = \{\text{Taxation}, \text{Import Controls}\}$. Thus, $es_1 >_{dlo} es_2$, since $\text{Tax Exemption} \prec_{CP} \text{Taxation}$.
- If $\text{concepts}(dlo) = \{\text{Taxation}, \text{Import Controls}\}$, then $C(dlo) = \{\text{Taxation}, \text{GLIN Legal Document}, \text{Import Controls}, \text{Import-Export}\}$ and, thus, $es_2 >_{dlo} es_1$.
- if $\text{concepts}(dlo) = \{\text{Tax Exemption}, \text{Import Control}\}$, then $C(dlo) = \{\text{Tax Exemption}, \text{GLIN Legal Document}, \text{Import Controls}, \text{Import-Export}\}$ and, thus, es_1 and es_2 are incomparable with regard to $>_{dlo}$.

We are now ready to introduce the notion of stronger authorization.

Definition 5.6 (Stronger Authorization). Let u be a user, and let dlo be an object. Let A_1, A_2 be two authorizations such that $u \in U_{A_1} \cap U_{A_2}$ and $i(dlo) \in O_{A_1} \cap O_{A_2}$. Authorization A_1 is stronger than authorization A_2 with regard to u and dlo , written $A_1 >_{u,dlo} A_2$, iff one of the following conditions holds:

1. $\text{crd-spec}(A_1) >_u \text{crd-spec}(A_2)$.
2. $\text{crd-spec}(A_1)$ and $\text{crd-spec}(A_2)$ are incomparable with regard to the $>_u$ relation and $\text{ent-spec}(A_1) >_{dlo} \text{ent-spec}(A_2)$.
3. $\text{crd-spec}(A_1)$ and $\text{crd-spec}(A_2)$ are incomparable with regard to the $>_u$ relation, $\text{ent-spec}(A_1)$ and

$\text{ent-spec}(A_2)$ are incomparable with regard to the $>_{dlo}$ relation, and $\text{priv}(A_1) <_P \text{priv}(A_2)$.

4. $\text{crd-spec}(A_1)$ and $\text{crd-spec}(A_2)$ are incomparable with regard to the $>_u$ relation, $\text{ent-spec}(A_1)$ and $\text{ent-spec}(A_2)$ are incomparable with regard to the $>_{dlo}$ relation, $\text{priv}(A_1)$ and $\text{priv}(A_2)$ are incomparable with regard to the $<_P$ relation and $\text{sign}(A_1) = "-"$, whereas $\text{sign}(A_2) = "+"$.

We can finally introduce the notion of valid authorizations, that is, the strongest authorizations in \mathcal{AB} . We distinguish between authorizations valid for a slot or a link.

Definition 5.7 (Valid Slot Authorization). Let u be user and let dlo be an object. Let s be a slot of dlo . Let \mathcal{AB} be an authorization base and let A be an authorization in \mathcal{AB} such that $\text{priv}(A) \neq \text{link}$ and either $\text{slot-spec}(\text{ent-spec}(A)) = \emptyset$ or $s \in \text{slot-spec}(\text{ent-spec}(A))$. Authorization A is valid for user u with regard to slot s of object dlo if and only if no authorization $A' \in \mathcal{AB}$ exists such that $\text{sign}(A') \neq \text{sign}(A)$, $\text{priv}(A') \neq \text{link}$, $A' >_{u,dlo} A$, and either $\text{slot-spec}(\text{ent-spec}(A')) = \emptyset$ or $s \in \text{slot-spec}(\text{ent-spec}(A'))$.

Definition 5.8 (Valid Link Authorization). Let u be a user and let dlo be an object. Let l be a link of dlo . Let \mathcal{AB} be an authorization base and let A be an authorization in \mathcal{AB} such that $\text{priv}(A) \neq \text{view}$ and either $\text{ent-spec}(A)$ is of the form co-spec.slot-spec with $\text{slot-spec} = \emptyset$, or $\text{ent-spec}(A)$ is of the form link-spec with $l \in \text{link-spec}$. Authorization A is valid for user u with regard to link l of object dlo if and only if no authorization $A' \in \mathcal{AB}$ exists such that $\text{priv}(A') \neq \text{view}$, $\text{sign}(A') \neq \text{sign}(A)$, $A' >_{u,dlo} A$, and either $\text{ent-spec}(A')$ is of the form co-spec.slot-spec , with $\text{slot-spec} = \emptyset$, or $\text{ent-spec}(A')$ is of the form link-spec , with $l \in \text{link-spec}$.

6 ACCESS CONTROL

Access control deals with verifying whether a user requiring access to a certain dlo can be authorized, according to the authorizations in \mathcal{AB} . With respect to administering access control, we distinguish two different types of sites: *client sites* that are sites from which the access request originates and *server sites* that are "information provider" sites storing one or more digital libraries and which are in charge of evaluating access requests. Note, however, that the same physical site may host both client and server sites. Each site has a security administration system (AS), which is responsible for granting or denying accesses to the $dlos$ located at that site and for submitting access requests to other sites. In the following, we denote with SSA the administration system at a server site and with CSA the administration system at a client site. We use AS when no distinction between client and server sites is needed.

Enforcing access control entails three main issues. First, each server site must have a proper access control mechanism to enforce the server site access control policy. The second task is the evaluation of credentials. Indeed, the credential system of the server site may differ from that of the DL site from which access is requested, or the client site may not have any credential mechanism in place. Addressing these issues may require the establishment of mappings among different sites. Finally, whenever a user submits an access request along with the credential

information, the access control mechanism of the server site must be sure that the credentials are authentic. In the following sections, we address these three issues.

6.1 Enforcing Access Control

Once a user submits an access request along with the appropriate information, that request must be evaluated against the authorizations stored in the server \mathcal{AB} . Suppose that a user u wishes to exercise privilege p on objects dlo . At the server site, his/her access request can be represented as a triple (u, dlo, p) , where u is the user requesting the access, $dlo \in \mathcal{OB}$ is the object to which u requires the access, and p is the privilege that user u wishes to exercise on dlo . In our model, one can authorize a user to exercise a privilege on a whole object or only selected portions (i.e., slots) and/or particular links within the object. This means that an access request can be *partially* authorized.

Given an authorization base \mathcal{AB} and an access request (u, dlo, p) , we first need to identify the subset of the authorizations in \mathcal{AB} that must be evaluated in order to decide whether the access request can be authorized. This subset is formally identified by means of the notion of *authorization base projection*, defined as follows:

Definition 6.1 (Authorization Base Projection). Let (u, dlo, p) be an access request and let \mathcal{AB} be an authorization base. The projection of \mathcal{AB} with regard to (u, dlo, p) , denoted $\Pi_{(u,dlo,p)}(\mathcal{AB})$, is the subset of \mathcal{AB} , such that, $\forall A \in \mathcal{AB}$, $A \in \Pi_{(u,dlo,p)}$ iff $u \in U_A$, $i(dlo) \in O_A$, and $\text{priv}(A) = p$ or $p <_P \text{priv}(A)$.

Thus, upon an access request, the user is given a *view* of the object that contains only those slots and links for which he/she has a valid positive authorization. In the case of totally authorized requests, the view is equal to the whole object. The notion of object view is formally defined as follows:

Definition 6.2 (Object View). Let (u, dlo, p) be an access request. The view of user u on object dlo with regard to privilege p (denoted as $V_u^p(dlo)$) is the union of a set $S \subseteq \text{slots}(dlo)$ of slots and a set $L \subseteq \text{links}(dlo)$ of links such that:

- $\forall s \in \text{slots}(dlo), s \in S$ iff there exists in $\Pi_{(u,dlo,p)}$ a positive authorization A valid for user u with regard to slot s of object dlo .
- $\forall l \in \text{links}(dlo), l \in L$ iff there exists in $\Pi_{(u,dlo,p)}$ a positive authorization A valid for user u with regard to link l of object dlo .

An algorithm enforcing access control is reported in Fig. 4. Algorithm 5.1 receives as input an access request (u, dlo, p) and an Authorization Base; it returns REJECT, if the access request cannot be authorized, it returns the view of user u on object dlo with regard to privilege p , otherwise. Suppose that user u requires to access object dlo under privilege p . The algorithm makes use of two arrays, namely, Curr_slots and Curr_links . Suppose that s_1, \dots, s_n are the slots in dlo , whereas l_1, \dots, l_m are the links in dlo . Curr_slots and Curr_links are incrementally updated and, at the end of the algorithm, the i th element of Curr_slots (Curr_links) is equal to "+" iff user u is authorized to access slot s_i (link l_i)

Algorithm 5.1 Access Control Algorithm

INPUT: 1) An access request (u, dlo, p) , 2) The authorization base \mathcal{AB}

OUTPUT: 1) $V_u^p(dlo)$, if $V_u^p(dlo) \neq \emptyset$, 2) REJECT, otherwise

METHOD:

1. Compute $\Pi_{(u,dlo,p)}(\mathcal{AB})$
2. $Curr_slots$, $Curr_links$ and $V_u^p(dlo)$ are initialized to be empty
3. **If** $\Pi_{(u,dlo,p)}(\mathcal{AB}) \neq \emptyset$
 - Repeat**
 - $Act_Auth := \{A \mid A \in \Pi_{(u,dlo,p)}(\mathcal{AB}), \text{ and } \nexists A' \in \Pi_{(u,dlo,p)}(\mathcal{AB}) \text{ such that } \text{crd-spec}(A') >_u \text{crd-spec}(A)\}$
 - If** $Act_Auth \neq \emptyset$: $Find_strong_auth(Act_Auth, dlo, Curr_slots, Curr_links)$
 - Remove Act_Auth from $\Pi_{(u,dlo,p)}(\mathcal{AB})$
 - Until** $\Pi_{(u,dlo,p)}(\mathcal{AB}) \neq \emptyset$
 - For** each i such that $Curr_slots[i] = '+'$: Add s_i to $V_u^p(dlo)$
 - For** each i such that $Curr_links[i] = '+'$: Add l_i to $V_u^p(dlo)$
 - If** $V_u^p(dlo) \neq \emptyset$: **return** $V_u^p(dlo)$
 - else**: **return** REJECT
- else**: **return** REJECT

Fig. 4. Access control algorithm.

of object dlo under privilege p . First, the algorithm computes (Step 1) the projection of the authorization base with regard to the access request. If the projection is empty, the access is denied. Otherwise, in Step 3 the authorizations in $\Pi_{(u,dlo,p)}(\mathcal{AB})$ with the strongest credential specifications are considered. From these authorizations, the strongest ones are selected (this selection is performed by procedure *Find-strong-auth* illustrated in Figs. 5 and 6). Procedure *Find-strong-auth* receives as input the set of authorizations in $\Pi_{(u,dlo,p)}(\mathcal{AB})$ with the strongest credential specifications (these authorizations are contained into set Act_Auth), the object on which the access request is performed and the arrays $Curr_slots$ and $Curr_links$ computed till that point in the computation. For each slot s_i (respectively, link l_i) of object dlo , procedure *Find-strong-auth* determines if there exists in Act_Auth a positive valid authorization for user u with regard to slot s_i (respectively, link l_i) of object dlo . If such authorizations exists, then $Curr_slots[i]$ (respectively, $Curr_links[i]$) is set equal to "+." Then, the authorization in Act_Auth are removed from $\Pi_{(u,dlo,p)}(\mathcal{AB})$ and the process is iteratively repeated until all the authorizations in $\Pi_{(u,dlo,p)}(\mathcal{AB})$ have been considered. At the end of the algorithm, $V_u^p(dlo)$ is constructed from $Curr_slots$ and $Curr_links$.

The correctness of Algorithm 5.1 is stated by the following theorem:

Theorem 6.1. *Algorithm 5.1 terminates. Moreover, given an access request (u, dlo, p) it returns $V_u^p(dlo)$ iff $V_u^p(dlo) \neq \emptyset$; it returns REJECT, otherwise.⁶*

Example 6.1. Suppose that $\mathcal{AB} = \{A_1, A_2, A_3, A_4\}$, where A_1, \dots, A_4 are the authorizations of Example 5.1. Let Tom be a NML Employee and let dlo_1 be an object containing the concepts `Import Controls` and `Imports Tax`.

Suppose that Tom issues the following access request: $(Tom, dlo_1, \text{view-all})$. By authorizations A_2 and A_3 , Tom is allowed to see all the information in dlo_1 , except for the links that dlo_1 eventually contains. Now, suppose that the access request $(Helen, \text{World Law Bulletin}, \text{view-all})$ is submitted to the system, where user Helen is a LLOC employee. By authorization A_1 , Helen is returned a view of the `World Law Bulletin` which does not contain the `Blue page report`.

6.2 Credential Evaluation

If the client site has a credential mechanism, the server can selectively grant access to users of the client site having specific credentials. For instance, the American Law Library of Congress can decide to authorize all the users of the Italian Law Library of the Parliament to submit their access requests or it can give access only to the legal research analysts of the Italian Law Library of the Parliament. In the latter case, there is a need for first mapping the client credentials onto the server credential hierarchies because the credential hierarchy of the Italian DL may be different from that of the American DL (for instance, the Italian DL can use a different set of attributes to describe the legal research analyst profile, or it can use different names to represent the same information). Such mapping is stored at the client site and used by the CSA when it submits a user request to the SSA. It is also possible that the mapping requires an extension of the client credentials. Indeed, suppose that the credential-type legal research analyst, stored at the server site, contains a mandatory attribute `SSN` which is not present in the Italian counterpart. This attribute must be added to the Italian credential-type during the mapping since it is mandatory to supply a value for `SSN` each time an Italian legal research analyst's access request is submitted to the American Law Library of Congress.

6. See [1] for the proof.

```

Procedure Find-strong-auth(Auth,dlo, slots, links)

1. Curr_Auth := {A | A ∈ Auth, i(dlo) ∈ co-spec(ent-spec(A)) and slot-spec(ent-spec(A)) ≠ ∅, or
   ent-spec(A) is of the form link-spec }

2. Repeat
   Curr_Auth' := {A | A ∈ Curr_Auth and ∄ A' ∈ Curr_Auth such that priv(A') <P priv(A)}
   For each A ∈ Curr_Auth' such that sign(A) = '+':
     If priv(A) ≠ link:
       For each si ∈ slot-spec(ent-spec(A)) such that slots[i] is empty:
         If ∄ A' ∈ Curr_Auth' such that sign(A') = '-' and si ∈ slot-spec(ent-spec(A')): slots[i] := '+'
         else: slots[i] := '-'
       else:
         For each li ∈ link-spec(ent-spec(A)) such that links[i] is empty:
           If ∄ A' ∈ Curr_Auth' such that sign(A') = '-' and li ∈ link-spec(ent-spec(A')): links[i] := '+'
           else: slots[i] := '-'
         endif
       Remove Curr_Auth' from Curr_Auth
     Until Curr_Auth ≠ ∅

3. Curr_Auth := {A | A ∈ Auth, i(dlo) ∈ co-spec(ent-spec(A)) and slot-spec(ent-spec(A)) = ∅}

4. Repeat
   Curr_Auth' := {A | A ∈ Curr_Auth and ∄ A' ∈ Curr_Auth such that priv(A') <P priv(A)}
   For each A ∈ Curr_Auth' such that sign(A) = '+':
     If priv(A) ≠ link:
       If ∄ A' ∈ Curr_Auth' such that sign(A') = '-' and priv(A') = priv(A):
         For each i such that slots[i] is empty: slots[i] = '+'
         else: For each i such that slots[i] is empty: slots[i] := '-'
       endif
     If priv(A) ≠ view or priv(A) = link:
       If ∄ A' ∈ Curr_Auth' such that sign(A') = '-' and priv(A') = priv(A):
         For each i such that link[i] is empty: links[i] = '+'
         else: For each i such that slots[i] is empty: slots[i] := '-'
       endif
     Remove Curr_Auth' from Curr_Auth
   Until Curr_Auth ≠ ∅

5. Curr_Auth := {A | A ∈ Auth, co-spec(ent-spec(A)) is a conceptual expression}

```

Fig. 5. Steps 1, ..., 5 of procedure *Find-strong-auth*.

Note, moreover, that a different approach would consist of extending the server credential hierarchy by a new credential-type foreign legal research analyst and then mapping the credential-type legal research analyst of the Italian credential hierarchy onto the new credential-type. Accesses to the Italian legal research analysts can then be given by adding proper authorizations for the foreign legal research analyst profile to the *AB* of the American Law Library of Congress.

As the above discussion shows, different mappings among credential type hierarchies can be devised and a fully automatic mapping is not always possible. The problem is similar to the problem of schema integration in heterogeneous database systems. Solutions proposed in this context [11] or in ontology [31] can be used for credential-type hierarchy mapping. Note that, it is not always possible to find a one-to-one mapping between the credential-type

hierarchies at the server and client site but, usually, you find credential types with a certain degree of similarity with the given one. In the following, we assume that the mapping is performed manually or semiautomatically by the ASs at the client and server sites. Once, the mapping has been defined, mapping information is entered into the CSA.

If the client site does not have a credential mechanism in place, or if the client and the server credential-type hierarchies belong to completely different domains, no mapping exists. In such a case, when the CSA submits an access request on behalf of one of its users, the SSA sends the CSA a list of information it must submit. Such information is necessary for the SSA to process the access request against the current authorizations stated for the requested object. It is possible that access may be denied because, based on the information provided by the CSA on behalf of its user, the user does not qualify for the access.

```

Procedure Find-strong-auth(Auth, dlo, slots, links)
.....

6. Repeat
    $Auth' := \{A \mid A \in Curr\_Auth \text{ and } \nexists A' \in Curr\_Auth \text{ such that } \text{ent-spec}(A') >_{dlo} \text{ent-spec}(A)\}$ 
   Remove  $Auth'$  from  $Curr\_Auth$ 
   Repeat
      $Curr\_Auth' := \{A \mid A \in Auth' \text{ and } \nexists A' \in Auth' \text{ such that } \text{priv}(A') <_P \text{priv}(A)\}$ 
     For each  $A \in Curr\_Auth'$  such that  $\text{sign}(A) = '+'$ :
       If  $\text{priv}(A) \neq \text{link}$ :
         If  $\nexists A' \in Curr\_Auth'$  such that  $\text{sign}(A') = '-'$  and  $\text{priv}(A') = \text{priv}(A)$ :
           For each  $i$  such that  $\text{slots}[i]$  is empty:  $\text{slots}[i] := '+'$ 
         else: For each  $i$  such that  $\text{slots}[i]$  is empty:  $\text{slots}[i] := '-'$ 
       If  $\text{priv}(A) \neq \text{view}$  or  $\text{priv}(A) = \text{link}$ :
         If  $\nexists A' \in Curr\_Auth'$  such that  $\text{sign}(A') = '-'$  and  $\text{priv}(A') = \text{priv}(A)$ :
           For each  $i$  such that  $\text{links}[i]$  is empty:  $\text{links}[i] := '+'$ 
         else: For each  $i$  such that  $\text{slots}[i]$  is empty:  $\text{slots}[i] := '-'$ 
       endfor
     Remove  $Curr\_Auth'$  from  $Auth'$ 
     Until  $Auth' \neq \emptyset$ 
   Until  $Curr\_Auth \neq \emptyset$ 

7.  $Curr\_Auth := \{A \mid A \in Auth \text{ such that } \text{ent-spec}(A) \text{ is of the form } \text{slot-spec}\}$ 

8. Repeat
    $Curr\_Auth' := \{A \mid A \in Curr\_Auth \text{ and } \nexists A' \in Curr\_Auth \text{ such that } \text{priv}(A') <_P \text{priv}(A)\}$ 
   For each  $A \in Curr\_Auth'$  such that  $\text{sign}(A) = '+'$ :
     For each  $s_i \in \text{slot-spec}(\text{ent-spec}(A))$  such that  $\text{slots}[i]$  is empty:
       If  $\nexists A' \in Curr\_Auth'$  such that  $\text{sign}(A') = '-'$  and  $s_i \in \text{slot-spec}(\text{ent-spec}(A'))$ :  $\text{slots}[i] := '+'$ 
       else: slots}[i] := '-'
     endfor
   Remove  $Curr\_Auth'$  from  $Curr\_Auth$ 
   Until  $Curr\_Auth \neq \emptyset$ 

```

Fig. 6. Steps 6, 7, and 8 of procedure *Find-strong-auth*.

The following example clarifies the discussion:

Example 6.2. Suppose that a user u connected to a client C submits a request to access an object dlo stored at a server S . Moreover, suppose that C does not have a credential hierarchy but the server does. When the SSA receives the access request, it verifies whether user u can access dlo , according to the security policies of site S . For instance, access can be granted if there is an explicit rule authorizing each user of C to access dlo . If this is not the case, the SSA analyzes the authorizations associated with dlo , to determine which are the attributes that appear in their credential specification. For instance, if the AB at the server site contains the authorization $(X.\text{age} > 18, dlo, \text{view-all}, +)$, then the CSA is requested to send the age of user u since without such information the above authorization cannot be applied to user u .

Computing this information each time a CSA submits an access request is inefficient since it may require the evaluation of a large number of authorizations. A possible solution could be to permanently maintain, for each object in the DL, the list of information the CSA should submit.

However, such precomputation could be very expensive and space consuming, due to the large number of objects a DL may contain. For this reason, we adopt a strategy based on a *partial* precomputation. We recall that, in our system, authorizations can be granted both on specific *dlos* (by specifying their IDs) or on the basis of the concepts an object contains. Since, in general, the number of concepts are significantly lower than that of *dlos*, the SSA permanently maintains a table which contains, for each concept, the information the CSA should provide to gain access to the objects containing that particular concept. Such information is computed using the algorithm in Fig. 7. The algorithm receives as input an authorization base and returns a table, called *Client_Info*, containing an entry for each concept in CP . This entry contains a set of values the CSA must provide when it requires access to an object containing that particular concept. In the algorithm, we use function “Add()” to add or modify *Client_Info*. The result of the statement “Add($cp, attr$) to *Client_Info*” is the addition of ($cp, attr$) to *Client_Info* if there is no entry for cp in the table. Otherwise, it is the addition of $attr$ to such entry.

Algorithm 6.1 Client_Info Generator Algorithm

INPUT: An authorization base \mathcal{AB}

OUTPUT: The table *Client_Info*. Each tuple in *Client_Info* has the form $(cp, attr_set)$, with $cp \in \mathcal{CP}$ and $attr_set \in 2^{\mathcal{AN}}$

METHOD:

1. $\mathcal{AB}' := \{A \mid A \in \mathcal{AB}, \text{co-spec}(\text{ent-spec}(A)) \text{ is a conceptual expression such that } C_types(\text{crd-spec}(A)) = \{\top_{CT}\}\}$
2. **For** each $A \in \mathcal{AB}'$:
For each $cp \in Concepts(\text{co-spec}(\text{ent-spec}(A)))$:
For each attribute a appearing in $\text{crd-spec}(A)$: Add (cp, a) to *Client_Info*
endfor
endfor
3. **return** *Client_Info*

Fig. 7. An algorithm for computing *Client_Info*.

Example 6.3. Consider the conceptual hierarchy in Fig. 1. Suppose \mathcal{AB} consists of the following authorizations:

```
(X.age > 25, Tax Exemption, view-all, +);
(X.nationality = Italy, (Import Control ∨
Tax-Incentive), view, -);
((X.nationality = US ∧ X.national origin = US),
(Import-Export ∧ Tax-Incentive), view, -);
(LLOC employee(X), Tax Exemption, view, -).
```

The corresponding *Client_Info* table built by Algorithm 6.1 is shown in Table 4.

When the SSA has to determine the information it needs from the client, it first computes the set of concepts characterizing the requested *dlo* (this set corresponds with the set $C(dlo)$ introduced in Section 4.1). For each of those concepts, it selects the corresponding entry in *Client_Info* and adds the attributes in that entry to the list of information it needs from the CSA. Note that this operation is very efficient since it only requires access to *Client_Info* for each concept in $C(dlo)$. Then, the SSA considers the authorizations in the \mathcal{AB} explicitly containing the ID of the requested object to verify whether additional information is needed for their evaluation, besides the ones already in the list, using the same method in Algorithm 6.1. If this is the case, such information is added to the list. Note, however, that, this latter operation is not computationally expensive because, in general, authorizations referring to a particular object are a small portion of the \mathcal{AB} .

6.3 Client and Server Access Control Protocol

Under our approach, all global access requests from users are mediated by the local site. Secure communications between the local site (the client) and the global site (the server) are enforced by means of private and public keys using the digital signature and encryption algorithms [18], [27]. Under our approach, a SSA need not be aware of the public keys of all global users, but only need to deal with the public keys of the different clients. The CSA at the local site, in turn, authenticates local users, according to what-

ever approach is locally used. Authentication of local users is not, however, of any concern to the global sites.

Moreover, note that we make the assumption that the SSA trusts the CSA to submit all the credentials needed for the evaluation of an access request. For this reason, we assume that each server site maintains a list of *trusted client sites*; these sites are the only ones from which access requests can be submitted.

The interaction between a CSA and a SSA can be divided into two main phases. The first is the *introduction phase* that takes place only once when a server decides to include a given client in the list of the sites from which it is accessible.

The second phase is the *access control phase* that takes place each time the CSA submits an access request to the SSA on behalf of one of its users.

The activities in both the above phases are described below. Assume E_c (D_c) and E_s (D_s) denote the public (private) keys of the CSA and the SSA, respectively. We assume that these keys are generated by a Certification Authority (CA) (see [18], [27] for more details).

The Introduction Phase:

1. The CSA sends the SSA a certificate containing its name and its public key after signing it with D_c . This message is encrypted with E_s . When the SSA receives the certificate, it decrypts it with D_s , verifies the public key of the CSA (E_c) from the Certification Authority, and verifies CSA's signature using E_c . The SSA adds the client name to the list of clients authorized to access it.

TABLE 4
Table *Client_Info* for the \mathcal{AB} of Example 6.3.

Concept	Attributes
Import Control	Nationality
Import-Export	Nationality, National Origin
Tax-Exemption	Age
Tax-Incentive	Nationality, National Origin

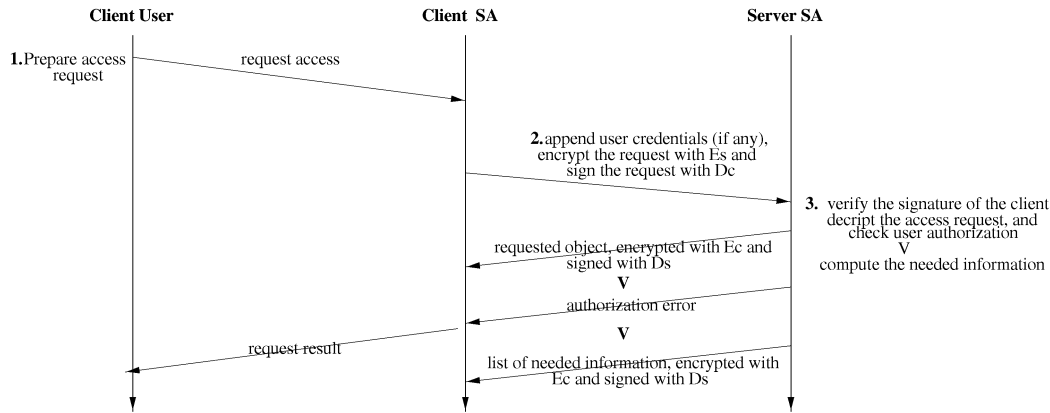


Fig. 8 Interactions between the CSA and the SSA during access control.

2. The SSA sends an acknowledgment to the CSA by signing it with D_s and the CSA then verifies the authenticity of the messages received by the SSA using E_s .
3. If the SSA enables access only to client site's users on the basis of credentials, a proper mapping will be established between the client and the server credential types, as described in the previous section.

The Access Control Phase:

1. When a user u wishes to access a dlo , he/she sends the access request to his/her local AS. If the requested object is local, then the AS evaluates the access request using the strategy illustrated in Section 6.1. Otherwise, if a mapping has been established between the server and the client credentials, the CSA submits the user credential along with the access request. Because of the mapping performed during the introduction phase, the submitted credential would be of one of the types in the server credential-type hierarchy. Otherwise, no further information is submitted to the SSA. Since the user credential can contain confidential information, the access request is first encrypted by the CSA with E_s and then signed by D_c .
2. When the SSA receives the access request, it first verifies whether it is from an authorized client; if not, it rejects the access request. Then, it verifies the signature of the CSA using E_c and decrypts the message with D_s .
3. If the access request contains a security credential, the SSA evaluates the request against such credential and returns the result to the CSA (the result is encrypted using E_c and signed using D_s). Otherwise,
4. the SSA analyzes the authorizations involving the requested object to determine which are the user information needed to evaluate such authorizations. It then returns to the CSA a list of information it may submit to have a high probability to access the object. This information is computed using Algorithm 6.1.
5. The CSA can then decide to provide some or all of the information to the server or to give up.
6. If the CSA provides all the information to the SSA, then the SSA evaluates the client request against these information, and it returns the answer to the CSA.
7. If the CSA sends only part of the information, the SSA evaluates the access request and returns the answer to the CSA. If the access is granted or the CSA has provided all the information to the SSA, then the interaction between the client and the SSA halts. Otherwise, Steps 5 and 6 are executed once again, until either the client decides to terminate the interaction with the server or it provides all the information requested by the server.

The interactions between the client and the SSA during the access control phase are graphically illustrated in Fig. 8 (in the figure, symbol "V" denotes the logical connector OR).

7 IMPLEMENTATION OF DLAM

We have implemented a first prototype system for the access control model presented in this paper. The prototype system, which we refer to as *Digital Library Authorization Module—DLAM* has been developed on top of the Oracle 8.03 DBMS, using Delphi 3 client/server.

Since our authorization model relies on the use of conceptual hierarchies for supporting content-based access control for $dlos$, DLAM has been integrated with a mechanism that extracts concepts from objects, builds a hierarchy of concepts and then classifies new documents into one or more concept-classes. The mechanism we use for concept extraction is based on the work reported in [16]. The interaction between DLAM and the Document Classification Module is illustrated in Fig. 9. Each time a new document is acquired by the DL, it is first processed by the document classification module, which extracts the relevant concepts from this object. Information on the relevant concepts are then stored into Oracle tables and used by DLAM to perform content-based access control on $dlos$.

DLAM implements all the functions of the access control model presented in this paper, except the evaluation of global user credentials. An extension of DLAM to deal with credential evaluation in a heterogeneous environment is currently being developed. In the following sections, we briefly illustrate the support DLAM provides for the

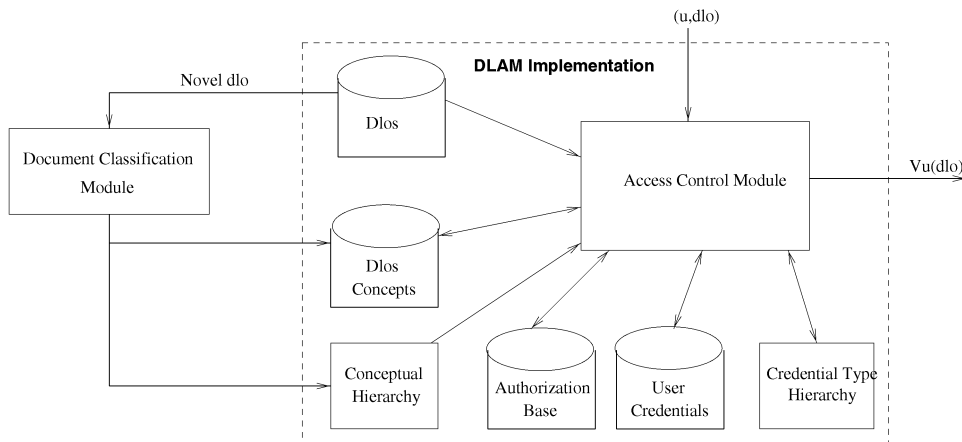


Fig. 9. The system architecture.

management of administrative operations and access control. More details on DLAM implementation can be found in [2].

7.1 Administrative Operations in DLAM

DLAM allows the Security Officer (SO) to easily perform administrative operations such as the insertion and deletion of an authorization, or to fill in the credential of a new user. For each administrative operation supported by our model, DLAM provides a graphical interface that supports the SO in performing the administrative operation. For instance, new authorizations can be entered into the \mathcal{AB} using the screen shown in Fig. 10. In this screen, the SO enters general information on the new authorization, such as the sign and the privilege of the authorization, the type of its credential specification (that is, whether it is a set of user identifiers or a credential expression), and the type of the entity specification. After this preliminary information is entered, the SO must specify both the credential and the entity specification of the new authorization. These operations are performed using a graphical interface that dynamically changes according to the type of credential and entity specification that must be entered.

Moreover, DLAM provides graphical tools to define and update the credential-type hierarchy and to structure *dlos* into slots. Similar functions are provided to insert/delete a link in a *dlo*.

7.2 Access Control in DLAM

Access control in systems like DLAM can be rather expensive since evaluating an access request often requires the evaluation of several credentials and entity specifications at runtime. To reduce this cost, we have adopted authorization precomputation strategies. More precisely, each time an authorization is inserted, we calculate and store the set of objects to which it applies. Thus, access control becomes more efficient since there is no difference in costs between content-dependent and content-independent authorizations. As far as users are concerned, we adopt a slightly different strategy. Precalculating the set of users to which an authorization applies is not always convenient since credentials may frequently change. Therefore, our strategy is to statically evaluate only those credential expressions containing attributes that seldom

change, and evaluating all the other credential expressions at runtime.

We are currently developing a prototype system for testing our materialization strategy. However, we believe we will obtain results similar to the ones we have obtained for the temporal access control model proposed by us in [5] since the two models have almost the same complexity. For the model presented in [5], experiment tests on the materialization strategy have shown that the materialization strategy is always preferable to a run-time evaluation approach when administrative operations are less than 15-20 percent of the number of access requests submitted to the system [13]. This is a reasonable assumption in real situations since, generally, access requests are considerably more frequent than administrative requests.

Access control in DLAM is based on the conflict resolution policy presented in Section 5.2, which basically solves conflicts by establishing a priority among the various hierarchies on which our model relies and considers the credential-type hierarchy as prevailing with regard to the conceptual hierarchy. When conflicts are not solved by the hierarchies, the negative authorizations are considered as prevailing. However, note that, to provide more flexibility, DLAM allows the SO to modify such default conflict resolution policy by assigning different weights to the criteria used to solve conflicts. More precisely, the SO can decide to consider negative authorizations as always prevailing (in such case, the hierarchies are not considered for conflict resolution), or to assign greater priority to the conceptual hierarchy with regard to the credential-type hierarchy. In such cases, the algorithms we use to enforce access control are a slightly modification of the algorithm presented in Section 6.1.

8 CONCLUSIONS AND FUTURE WORK

Digital Libraries (DLs) introduce several challenging requirements with respect to the formulation, specification, and enforcement of adequate data protection policies. Traditional authorization models are not adequate to meet access control requirements typical of a DL environment. In this paper, we have proposed a content-based authorization model suitable for a DL environment. The model also supports flexible specification of authorizations based on

Fig. 10. Authorization specification in DLAM.

the qualifications and characteristics of users (including positive and negative) and varying granularity of authorization objects ranging from sets of library objects to specific portions of objects.

We plan to extend the work reported in this paper along several directions. A first direction deals with content-based access control for images and videos. One of the problems is related to the difficulty of automatic recognition of image and video contents. We plan to extend our model based on the state-of-the-art in multimedia. However, note, that our access control model naturally extends to other media since it is based on concepts associated with objects. In particular, our access control model can be coupled with any system able to associate information in form of concepts to different media objects, such as systems supporting metadata information [26]. Moreover, we plan to extend our model to support *parametric authorizations*, that is, authorizations that contain variable fields and can thus be used as a compact way to denote a set of authorizations.

Then, we plan to extend this work to address issues such as copyright, privacy, and integrity that are of critical importance for the DL environment.

Moreover, note that the access control models presented in this paper can be intended as a *core mechanism* by which many protection policies can be enforced. A very important research direction concerns the development, on top of our access control mechanisms, of tools for the specification of such authorization policies and for their automatic mapping into a set of authorization rules. Such tools are particularly crucial when dealing with sophisticated authorization models like the one presented in this paper. This area,

however, has not been so far widely investigated. We plan to invest a major effort in this direction.

ACKNOWLEDGMENTS

The authors would like to thank Nick J. Kozura, the Library of Congress and Dr. Rubens Medina, Law Librarian, Library of Congress, for their suggestions and comments on issues related to GLIN. Thanks are also due to Dr. Richard Holowczak for his help in integrating DLAM with his concept extractor and to Ugo Capuozzo for the implementation of DLAM. The work of V. Atluri was supported in part by the US National Science Foundation under grant IRI-9624222. The work of Elisa Bertino and Elena Ferrari is partially supported by CIMIC, Rutgers University, Newark, NJ, by the MURST under the Project Interdata and by the CSELT under the Project Advanced Data Management Functions for Web Data.

REFERENCES

- [1] N. Adam, V. Atluri, E. Bertino, and E. Ferrari, "A Content-Based Authorization Model for Digital Libraries," technical report, Computer Science Dept., Univ. of Milano, 1998.
- [2] E. Ferrari, N. Adam, V. Atluri, and E. Bertino, "An Authorization System for Digital Libraries," *The VLDB Journal*, Year?
- [3] N. Adam et al., "Strategic Directions in Electronic Commerce and Digital Libraries: Towards a Digital Agora," *ACM Computing Surveys*, vol. 28, no. 4, 1996.
- [4] V. Atluri and W.K. Huang, "An Authorization Model for Workflows," *Proc. Fourth European Symp. Research in Computer Security (ESORICS '96)*, 1996.

- [5] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati, "A Temporal Access Control Mechanism for Database Systems," *IEEE Trans. Knowledge and Data Eng.*, vol. 8, no. 1, pp. 67–80, Feb. 1996.
- [6] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati, "An Access Control Model Supporting Periodicity Constraint and Temporal Reasoning," *ACM Trans. Database Systems*, vol. 23, no. 3, pp. 231–285, 1998.
- [7] E. Bertino, F. Buccafurri, E. Ferrari, and P. Rullo, "A Logic-Based Approach for Enforcing Access Control," *J. Computer Security*, vol. 8, nos. 2 and 3, pp. 109–139, 2000.
- [8] E. Bertino, F. Buccafurri, E. Ferrari, and P. Rullo, "A Logical Framework for Reasoning on Data Access Control," *Proc. 12th IEEE Computer Security Foundations Workshop*, pp. 175–189, June 1999.
- [9] E. Bertino, P. Samarati, and S. Jajodia, "A Flexible Authorization Mechanism for Data Management Systems," *ACM Trans. Information Systems*, vol. 17, no. 2, pp. 101–140, 1999.
- [10] E. Bertino, P. Samarati, and S. Jajodia, "An Extended Authorization Model," *IEEE Trans. Knowledge and Data Eng.*, vol. 9, no. 1, pp. 85–101, Jan./Feb. 1997.
- [11] S. Castano and V. De Antonellis, M.G. Fugini, and B. Pernici, "Conceptual Schema Analysis: Techniques and Applications," *ACM Trans. Database Systems*, vol. 23, no. 3, pp. 286–333, 1998.
- [12] C.K. Baru and A. Rajasekar, "A Hierarchical Access Control Scheme for Digital Libraries," *Proc. Third ACM Int'l Conf. Digital Libraries*, pp. 275–276, 1998.
- [13] E. Ferrari, E. Bertino, C. Bettini, A. Motta, and P. Samarati, "On Using Materialization Strategies for a Temporal Authorization Model," *Proc. Post-SIGMOD Workshop Materialized Views: Techniques and Applications*, pp. 34–41, 1996.
- [14] A. Glenn and D. Millman, "Access Management of Web-Based Services An Incremental Approach to Cross-Organizational Authentication and Authorization," *D-Lib Magazine*, 1998.
- [15] E. Gudes, H. Song, and E.B. Fernandez, "Evaluation of Negative, Predicate, and Instance-Based Authorization in Object-Oriented Databases," *Database Security, IV: Status and Prospects*, 1991.
- [16] R. Holowczak, *Extractors for Digital Library Objects*, PhD thesis, Rutgers Univ., Dept. of MS/CIS, 1997.
- [17] D. Jonscher and K.D. Dittrich, "An Approach for Building Secure Database Federations," *Proc. 20th Very Large Database Conf.*, 1994.
- [18] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*. Prentice Hall, 1995.
- [19] W. Kim, N. Ballou, J.F. Garza, and D. Woelk, "A Distributed Object-Oriented Database System Supporting Shared and Private Databases," *ACM Trans. Office Information Systems*, vol. 9, pp. 31–51, 1991.
- [20] U. Kohl, J. Lotspiech, and M.A. Kaplan, "Safeguarding Digital Library Contents and Users Protecting Documents Rather Than Channels," *D-Lib Magazine*, 1997.
- [21] T. Lunt, "Access Control Policies: some Unanswered Questions," *Computer & Security*, vol. 8, no. 1, 1989.
- [22] D. Millman, "Cross-Organizational Access Management A Digital Library Authentication and Authorization Architecture," *D-Lib Magazine*, 1999.
- [23] *Proc. First ACM Workshop Role-Based Access Control*, 1996.
- [24] F. Rabitti, E. Bertino, W. Kim, and D. Woelk, "A Model of Authorization for Next-Generation Database Systems," *ACM Trans. Database Systems*, vol. 16, no. 1, pp. 88–131, 1991.
- [25] P. Samarati, E. Bertino, and S. Jajodia, "An Authorization Model for a Distributed Hypertext System," *IEEE Trans. Knowledge and Data Eng.*, vol. 8, no. 4, pp. 555–562, Aug. 1996.
- [26] V.S. Subrahmanian, *Principles of Multimedia Database Systems*. Morgan-Kaufmann, 1997.
- [27] R.C. Summers, *Secure Computing: Threats and Safeguards*. Mc-Graw Hill, 1997.
- [28] R. Sandhu et al., "Role-Based Access Control Models," *Computer*, pp. 38–47, 1996.
- [29] B. Thuraisingham, "A Tutorial in Secure Database Systems," MITRE technical report, 1992.
- [30] S.H. Von Solms and I. Van Der Merwe, "The Management of Computer Security Profiles using a Role-Oriented Approach," *Computers & Security*, vol. 13, no. 8, pp. 673–680, 1994.
- [31] P. Weinstein and W.P. Birmingham, "Creating Ontological Metadata for Digital Library Content and Services," *Int'l J. Digital Libraries*, 1998.

- [32] M. Winslett, N. Ching, V. Jones, and I. Slepchin, "Using Digital Credentials on the World-Wide Web," *J. Computer Security*, vol. 5, 1997.



Nabil R. Adam is a professor of computers & information systems and the director of the Rutgers CIMIC Center in Newark. Dr. Adam has published numerous technical papers in such journals as *IEEE Transactions on Software Engineering*, *IEEE Transactions on Knowledge and Data Engineering*, *ACM Computing Surveys*, *Communications of the ACM*, *Information Systems*, *Journal of Management Information Systems*, and *International Journal of Intelligent and Cooperative Information Systems*. He coauthored/coedited nine books including *Electronic Commerce: Technical, Business, and Legal Issues*, Prentice Hall, 1998, *Databases Issues in GIS*, Kluwer Academic Publisher, 1997, and *Electronic Commerce* (1996), published as part of the Springer Verlag Lecture Notes Series in Computer Science. Dr. Adam is editor-in-chief of the *International Journal on Digital Libraries* and serves on the editorial board of the *Journal of Management Information Systems* and the *Journal of Electronic Commerce*. He served as a guest editor for the *Communications of the ACM*, *Operations Research*, and *Journal of Management Information Systems*. He is the cofounder and current chair of the IEEE Technical Committee on Digital Libraries. He served as general chair and program chair of various international conferences related to digital libraries and electronic commerce. Dr. Adam was elected as a distinguished speaker in the IEEE Computer Society's Distinguished Visitors Program (DVP) for the period 1997–2000. He was invited to lecture on digital libraries, electronic commerce, and other related topics at several national and international institutions.



Vijay Atluri received the BTech degree in electronics and communications engineering from Jawaharlal Nehru Technological University, Kakinada, India, the MTech degree in electronics and communications engineering from Indian Institute of Technology, Kharagpur, India, and the PhD degree in Information Technology from George Mason University, Virginia. She is currently an assistant professor of computer information systems in the MSIS Department at Rutgers University. Dr. Atluri's research interests include information systems security, database management systems, workflow management, and distributed systems. She has published more than 35 technical papers in the refereed journals and conference proceedings in these areas and is the coauthor of the book, *Multilevel Secure Transaction Processing*, Kluwer Academic (1999). She served as program chair for the 2000 ACM Workshop on Role-Based Access Control, and program cochair for the 1999 IFIP WG11.3 Working Conference on Database Security. In 1996, she was a recipient of the US National Science Foundation CAREER Award to investigate issues related to incorporating multilevel security into database management systems for advanced application domains such as office information systems, CAD/CAM, workflow systems. Dr. Atluri is a member of the IEEE Computer Society and the ACM.



Elisa Bertino is currently a professor of database systems in the Department of Computer Science of the University of Milan, where she is currently the chair of the department. She has also been on the faculty in the Department of Computer and Information Science of the University of Genova, Italy. Until 1990, she was a researcher for the Italian National Research Council in Pisa, Italy, where she headed the Object-Oriented Systems

Group. She has been a visiting researcher at the IBM Research Laboratory (now Almaden) in San Jose, at the Microelectronics and Computer Technology Corporation in Austin, Texas, at Rutgers University, and at Purdue University. Her main research interests include object-oriented databases, distributed databases, deductive databases, multimedia databases, interoperability of heterogeneous systems, integration of artificial intelligence and database techniques, database security. In those areas, Professor Bertino has published several papers in all major refereed journals and in proceedings of international conferences and symposia. She is a coauthor of the books *Object-Oriented Database Systems—Concepts and Architectures*, 1993 (Addison-Wesley), *Indexing Techniques for Advanced Database Systems*, 1997 (Kluwer Academic), and *Intelligent Database Systems*, 2000 (Addison-Wesley). She is or has been on the editorial boards of the following scientific journals: *ACM Transactions on Information and Systems Security*, the *IEEE Transactions on Knowledge and Data Engineering*, the *International Journal of Theory and Practice of Object Systems*, the *Very Large Database Systems (VLDB) Journal*, the *Parallel and Distributed Database Journal*, the *Journal of Computer Security, Data & Knowledge Engineering*, and the *International Journal of Information Technology*. She has been a consultant to several Italian companies on data management systems and applications and has given several courses to industries. She has been also involved in several European projects sponsored by the EEC under the ESPRIT programme. Professor Bertino is a senior member of IEEE and a member of ACM and AICA and has been named a Golden Core Member for her service to the IEEE Computer Society. She has served as a program committee member of several international conferences, such as ACM SIGMOD and VLDB, as program chair of the 1996 European Symposium on Research in Computer Security (ESORICS'96), as general chair of the 1997 International Workshop on Multimedia Information Systems, as program cochair of the 1998 IEEE International Conference on Data Engineering (ICDE), and as program chair of the 2000 European Conference on Object-Oriented Programming.



Elena Ferrari received the MS degree in information sciences and the PhD degree in computer science from the University of Milano, Italy, in 1992 and 1998, respectively. She is now an assistant professor in the Department of Computer Science at the University of Milano, Italy. She has also been a visiting researcher at George Mason University, Fairfax (VA) and at Rutgers University, Newark (NJ). Her main research interests include: database security,

access control models, multimedia databases, and temporal object-oriented data models. On these topics, she has published several papers in refereed journals and conference proceedings. She has served as PC cochair of the ECOOP '99 Workshop on Object-Oriented Databases and the ECOOP'00 Workshop on XML and Object Technology (XOT '2000), and as PC member of the 1997 International Workshop on Multimedia Information Systems, the 1999 International Workshop on Information Technologies for Electronic Commerce, the 1999 International Workshop on Electronic Commerce and Security, and the 2001 ACS International Conference on Computer Systems and Applications.