# An Access Control Mechanism for Large Scale Data Dissemination Systems*


Elisa Bertino
Computer Science Department
University of Milano, Italy
bertino@dsi.unimi.it

Elena Ferrari
Computer Science Department
University of Milano, Italy
ferrarie@dsi.unimi.it

Evaggelia Pitoura
Computer Science Department
University of Ioannina, Greece
pitoura@cs.uoi.gr



## Abstract

*Automatic data dissemination systems are becoming increasingly relevant in internet-based information systems. In such systems, users subscribe to the dissemination service by providing interest profiles. These profiles are then used to determine which information should be delivered to which users, whenever new information is entered into the system. A main shortcoming of existing dissemination systems is the lack of any access control mechanisms, enabling selective information dissemination based for example on security or other regulatory policies. In this paper, we present an access control model suitable for dissemination systems. The model is based on user profiles containing both user interests and user credentials (i.e., a set of attributes characterizing users for access control purposes). Information is then filtered before being delivered to users on the basis of both the user interests and credentials. To make authorization manageable when dealing with very large information systems, our model supports authorization domains, that allow one to group together information objects to which the same access control policies apply. Finally, in addition to formally defining our model and developing the related access control algorithm, we outline an implementation strategy.*


## 1 Introduction

The large volumes of data available for general access and the rapidity with which these data change make very difficult for users to track the data they are interested in. Thus, tools and systems for information dissemination are becoming crucial in a variety of application environments in both Internet and intranets [10]. In general, users subscribe to such systems by providing lists of topics they are interested in. Whenever new data are available for distribution, the system notifies the users who may be interested in such data based on the user preferences. Several such systems have been developed [2, 3, 10]. A main shortcoming of existing systems is the lack of any access control facility. In general, such a facility allows an administrator to specify rules stating which user can access which data for which purpose. Such facilities would be instrumental in enhancing the use of data dissemination systems. For example, access control facilities can be used to deny access to particular users; material promoting gun or drug use may not be distributed to minors, and confidential documents should be delivered only to authorized users. Furthermore, such facilities can be used to selectively disseminate information based on subscription rates or other special agreements.

The goal of our work is thus to define an access control mechanism, suitable for information dissemination systems. Our approach is based on the use of *user profiles*; such profiles contain two different kinds of information. The first kind refers to user interests. User interests can be specified either explicitly, by listing the information objects[1] the user is interested to receive, or through a concept query that describes which concepts the objects should (or should not) contain to be of interest to the user.

The second kind of information is in the form of user credentials, that is, a set of attributes characterizing users for access control purposes. Based on user credentials, suitable access control policies can be established and enforced. Information objects are then filtered before being delivered to users on the basis of both the user interests and credentials in such a way that a user receives only the information he/she is interested in and for which he/she has an appropriate authorization. We describe a language according to which access control policies can be stated and outline an implementation of access control, based on indexing techniques, on top of the filtering module.

Scalability is a central requirement in dissemination systems due to the huge number of both users and information


*The work of Elisa Bertino and Elena Ferrari has been partially supported by a grant from Microsoft Research. The work of Evaggelia Pitoura has been partially supported by GSRT PENED Grant 99-495


[1] By *information objects* (objects for short), we denote the information to be protected, that is, units of information which can be disseminated.

objects. To make authorization manageable, we group objects to which the same access policies apply into authorization domains. Authorization domains are defined both statically and dynamically. Dynamic specification is attained through concept queries that specify which objects belong to a domain based on the objects' content.

Other features of our model include: (i) two different browsing access modes: a notification mode to inform users for the availability of new objects and a propagation mode to disseminate the objects to the users, (ii) a credential specification language and an inheritance hierarchy for structuring credentials and (iii) indexing techniques for improving performance.

Since data dissemination is a new and emerging area of research, there has been very little prior work that addresses security for such systems. An access control model for Web documents has been presented in [7]. Although we borrow from [7] the hypertext model used to organize information objects, our work substantially differs in that we support a flexible user specification by which users are qualified by predicates on credentials instead of user's id and authorization based on the document contents instead of only on documents identifiers.

Winslett et al. [9] have first identified the need of user credentials. Building on their work we formalize the specification of user credentials by providing a credential specification language.

The concept of credential has same similarity with that of role which is the basis for a class of models known as Role-based Access Control (RBAC) models [6]. Roles can be seen as a set of actions or responsibilities associated with a particular working activity. 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, all users with a given age).

The remainder of this paper is structured as follows. Section 2 presents the system model. Section 3 illustrates the access control model, whereas Section 4 deals with access control. Finally, Section 5 concludes the paper.

# 2 The System Model

## 2.1 System Architecture

The overall architecture of our system is illustrated in Figure 1. Users are entitled to access the system services, upon subscription to the system. A user subscribes to the system by interacting with the *Subscription Module* (step 1). During the subscription phase, the user specifies his/her *profile*, which describes the information objects he/she is interested in. Additionally, the user is required to give some personal data (such as age or nationality) which are included

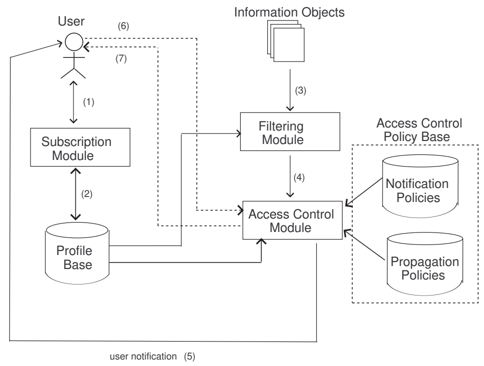


**Figure 1. System architecture**

in the profile. It may be required that such data are encoded in a certificate, issued by a proper Certification Authority, to ensure their authenticity.

Data collected during the subscription phase are subsequently used by the *Access Control Module* to verify which information objects each user can access. A profile may include additional parameters for example to determine how often the user has to be notified of new information.

Data about users collected during the subscription phase are stored into the *Profile Base* (step 2) and can then be updated by further interactions with the Subscription Module.

The system also provides a form of *implicit subscription*, by which the user is notified of the existence of new information objects without necessarily having subscribed to the system. For instance, attendants of a database conference are notified when new database-related information is available. For these users, the profiles are automatically generated by the system (for example, in the case of a database conference attendant, the profile will contain database as topic of interest, and a minimum set of personal data such as name, affiliation and email address). This kind of users as well as conventional subscribers may require the system not to send unsolicited information by sending an un-subscribe request.

As new objects are acquired by the system, or existing ones are modified, they are processed by the *Filtering Module* (step 3) that, on the basis of the profiles stored into the Profile Base, selects only those information objects that are relevant to the subscribers. The output of the filtering phase is thus a set of objects along with the users to which such objects are relevant. After the filtering phase, the *Access Control Module* is activated (step 4) that, on the basis of the access control policies stored into the *Access Control Policy Base* and on the user data stored into the Profile Base, verifies whether the users have the rights to be notified of the existence of the objects selected by the Filtering Module. In such case, the users are notified of the existence of some potentially relevant information objects (step 5). The user can then require some of those objects (step 6) and, if the user has the necessary authorizations, he/she will receive a copy of the objects (step 7).

The notification step helps people from being overwhelmed with too much information. Furthermore, users that do not have explicitly requested to subscribe are given only notification rights. However, they can be entitled to access the objects upon a further interaction with the Subscription Module.

### 2.2 Information Objects and Concepts

We assume that information objects are organized according to the hypertext model proposed in [7].[2]

At the base level, two kinds of objects are stored: *basic objects* and *nodes*. A basic object is a container of information that can be included into an information object. A node is a frame describing an information object. Nodes are unstructured and can contain any kind of information, such as text, video, images or graphics. Links are used to connect a node to other nodes or basic objects. The model supports two different types of links: *inclusion links* (links between nodes and basic objects) and *navigation links* (links between nodes). An inclusion link between a node and a basic object means that the basic object should be included into the information object described by the node. By contrast, a navigation link between two nodes means that a relationship exists between the two nodes and thus between the corresponding information objects. 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 identifiers of the two objects connected by the link.

Each information object has a set of concepts associated with it, describing its information content. Such concepts could be terms, such as in Information Retrieval approaches. The concepts associated with an information object can also have weights, representing the relevance of the concept for the information object. The set of concepts associated with an information object is a superset of the concepts associated with all basic objects connected to the corresponding node through inclusion links. The weights are the sum of weights. For instance, if a video is included in an information object, then the concepts associated with the information object should include all concepts associated with the video. The methods used to extract concepts depend on their type. If the object is a text file then concept extraction can be done using text retrieval methods [4], if it is a video file, concept extraction can be based on video retrieval [5], and so on. The methods used to extract concepts from basic objects are not directly relevant for the present discussion. We assume that there are some tools in place for concept extraction.

In the following, we denote with $\mathcal{CP}$ the set of concepts that can appear in information objects, whereas $\mathcal{W}$ denotes a set of weights that can be associated with concepts in $\mathcal{CP}$. An information object is thus described by a set of concepts and associated weights and by the set of links it contains, and it is characterized by a unique identifier.

**Definition 2.1 (Information Object Specification)** *An information object specification is a triple $(id, links, cps\_spec)$, where $id$ is the object identifier, $links$ is a set of link identifiers, and cps_spec is a set of pairs $(cp, w)$, with $cp \in \mathcal{CP}$ and $w \in \mathcal{W}$.*

**Example 2.1** $(\texttt{io}_1, \{(\texttt{Esprit Project}, 90), (\texttt{Esprit Project Budget}, 40)\})$[3] *is an example of information object specification referring to an information object dealing with Esprit projects and their budgets.*

## 3 The Access Control Model

### 3.1 Users, Credentials, and Profiles

Each user has a *profile* associated with him/her, consisting of two main components: a set of interest profiles and a set of user credentials. *Interest profiles* specify the topics the user is interested in. A *user credential* (credential for short) is a set of data about the user that are relevant for access control purposes. To make the task of credential specification easier, credentials with similar structures are grouped into *credential types*.

In the following we formalize the concepts of user credential, interest profile, and user profile. 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 $\mathcal{T}$. Moreover, we denote the set of credential type identifiers with $\mathcal{CTI}$, the set of credential identifiers with $\mathcal{CI}$, and the set of interest profile identifiers with $\mathcal{IPI}$. Finally, we use $\mathcal{U}$ to denote a set of account identifiers associated with the users who subscribed to the system. Credential types are formally defined as follows.

**Definition 3.1 (Credential type)** *A credential type is a pair (ct_id, attr), where ct_id $\in \mathcal{CTI}$ is the credential type identifier and attr is a set containing an item for each attribute of the credential type. Each element in attr is a triple (a_name,a_dom,a_type), where a_name $\in \mathcal{AN}$ is the attribute name, a_dom $\in \mathcal{T}$ is the attribute domain, and a_type $\in \{\texttt{opt}, \texttt{mand}\}$ indicates whether the attribute can assume a null value (a_type = "*`opt`*") or not (a_type = "*`mand`*").*

[2]Note, however, that our model can be applied also to information spaces that are not necessarily structured according to the hypertext model.

[3]For the sake of simplicity, links are omitted from the object specification.

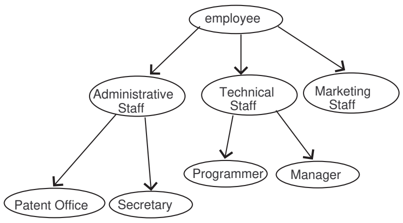


**Figure 2. A credential type hierarchy**

**Example 3.1** *The following is an example of credential type:*
(`programmer`, {(`name`,`string`,`mand`),(`project`, `string`,`mand`), (`project_type`,`string`,`mand`), (`manager`,`string`,`mand`), (`skills`, `set-of( string)`,`opt`)}).

Credential types are hierarchically organized according to a partial order $\prec_{\mathcal{CT}}$ (referred to as *credential type hierarchy*). Given, two credential types $ct_1$ and $ct_2 \in \mathcal{CTI}$ we say that $ct_2$ is a *subcredential type* (or shortly subtype) of $ct_1$ iff $ct_1 \prec_{\mathcal{CT}} ct_2$. Figure 2 shows an example of credential type hierarchy. 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 $ct \in \mathcal{CTI}$, $A(ct)$ denotes the set of attributes of the credential type instances. 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 3.2 (Credential)** *A credential $c$ is a triple $(c\_id, state, ct\_id)$, where $c\_id \in \mathcal{CI}$ is the credential identifier; state = ($a_1 : v_1, \ldots, a_n : v_n$), where $a_1, \ldots, a_n \in A(ct\_id)$ are the names of the attributes of c, $v_1, \ldots, v_n \in \mathcal{V}$ are their values, and $ct\_id \in \mathcal{CTI}$ is the identifier of the credential type of which c is an instance.*

**Example 3.2** *The following is an example of credential referring to the credential type hierarchy of Figure 2:*
($c_1$,(`name`:`'Bob'`,`division`:`'R&D'`,`projects`: {`'P125'`,`'R345'`}, `category`:`'senior'`, `benefits`:`null`),`manager`).

A user can describe the objects he/she is interested in by specifying one or more interest profile. An interest profile may either be specified by explicitly listing a set of information object identifiers or by means of *concept queries*, formally defined as follows.

**Definition 3.3 (Concept Query)** *A concept query is a boolean combination of predicates of the form $c[w]$ or c, where $c \in \mathcal{CP}$ and $w \in \mathcal{W}$.*

**Example 3.3** `'Esprit Project'[60] AND NOT 'Esprit Project Budget'` *is a concept query identifying all the information objects dealing with Esprit projects but not with their budgets. The query also specifies a weight for the first term, meaning that an information object satisfies the concept query if it contains the concept* `Esprit Project` *and the weight of this concept in the object is not less than* `60`.

Interest profiles are formally defined as follows.

**Definition 3.4 (Interest Profile)** *An interest profile is a pair $(ip\_id, ip\_spec)$, where $ip\_id \in \mathcal{IPI}$ is the interest profile identifier, and $ip\_spec$ is an interest profile specification, defined according to the following rules: i) each set of information object identifiers is an interest profile specification; ii) each concept query is an interest profile specification; iii) if $ps_1$ and $ps_2$ are interest profile specifications, then ($ps_1$ AND $ps_2$), ($ps_1$ OR $ps_2$), and NOT $ps_1$ are interest profile specifications.*

**Example 3.4** *By submitting the interest profile* (`ip`$_1$`, 'Esprit Project'[60] AND NOT 'Esprit Project Guideline'`) *the user declares to be interested in receiving information objects in which the concept* `Esprit Project` *has weight not less than* `60`, *but that do not contain the guidelines for project proposals.*

Besides interests and credentials, user profiles also contain additional data, such as the lifetime of the profile (that is, the subscription duration), the frequency of updates, that is, a measure of how often the user wants to be notified of new relevant information objects, and the amount of information to be received. This last information is specified by giving a *relevance threshold*, which is the minimum similarity score that an information object must have against the interest profile in order to be returned to the user.

**Definition 3.5 (User Profile)** *A user profile (profile for short) is a tuple $(u, lifespan, update\_freq, rel\_threshold, cs, ip)$, where $u \in \mathcal{U}$ is a user identifier, $lifespan$ is a time interval [$t_b$,$t_e$], with $t_b$, $t_e \in \mathbb{N}$, $t_b \leq t_e$ representing the lifespan of the profile, $update\_freq \in \mathbb{N}$ is the frequency of user updates, $rel\_threshold \in \mathbb{N}$ is the relevance threshold, cs is a set of credentials specified according to Definition 3.2, and $ip$ is a set of interest profiles specified according to Definition 3.4.*

The *Profile Base*, denoted as $\mathcal{PB}$, is the set of profiles associated with users in the system.

Profiles are assigned when a user subscribes to the system. Alternatively, users can be assigned a default profile in case they do not explicitly require to subscribe the system but they are entitled to access the system services (for instance because they belong to some mailing lists). In this case, the users are assigned a default credential, instance of the credential type *Non-Subscribers*, which contains only a minimum set of personal data and a default interest profile, which is automatically generated by the system on the basis of the available information.

### 3.2 Access Modes

The model supports two different types of browsing access modes: *(i)* a *notification access mode*: the user has the right to be notified that an information object is available (but he/she does not receive the object), and *(ii)* a *propagate access mode*: the user has the right of receiving the object and, eventually, navigating trough its links. To better protect objects containing navigation links, the propagate access mode is further specialized into: `selective_propagate`, which allows a user to see the information in an object but not the links it contains, and `total_propagate`, that allows a user to see the information in an object including all the links it contains. An additional access mode, called `link_propagate`, to authorize a user to see the existence of a specific link is considered in [1] but is omitted here due to space limitations. The propagation access mode subsumes the notification access mode. Note that, since we distinguish between notification and propagation access modes, once a user has been notified for a new information object, he/she has to explicitly request that information object to get it and such request is authorized only if the user has a (selective or total) propagation access mode on the object. If the user has a propagation authorization on the information object, he/she will receive a copy of the object with all the basic objects it includes. Link display depends on whether the propagation privilege is total or selective. When links are displayed, requests to navigate them are processed by the Access Control Module, and the request is authorized only if the requesting user has a propagation authorization on the information object referenced by the link. We denote with $\mathcal{M}$ the set of access modes.

### 3.3 Access Control Policies

Access control policies specify the access modes users can exercise on the objects in the system. They are specified by imposing a set of conditions on user credentials. Such conditions are expressed by means of *credential expressions*.

**Definition 3.6 (Credential Expression)** *The set of credential expressions is defined according to the following rules: i) each element in $\mathcal{CTI}$ is a credential expression; ii) if $a \in \mathcal{AN}$, $v \in \mathcal{V}$, and* OP $\in \{=, \neq, >, <, \geq, \leq, \in, \notin \subseteq, \not\subseteq, \subset, \not\subset, \supset, \supseteq, \not\supset, \not\supseteq\}$, *then $a$* OP *$v$ is a credential expression;*[4] *iii) if $ce_1$ and $ce_2$ are credential expressions, then ($ce_1$ AND $ce_2$), ($ce_1$ OR $ce_2$), and NOT $ce_1$ are credential expressions.*

**Example 3.5** *The credential expression (*`programmer` *AND* `project_type = 'Esprit'`*) denotes all the programmers working in an Esprit project.*

Since many information objects are usually subject to the same access control policies, it is useful to have mechanisms to group them. To this purpose, we introduce *authorization domains*. Authorization domains can be defined either *statically*, by listing the identifiers of objects composing the domain, or *dynamically*, by imposing conditions on the concepts an object should (or should not) contain to be included into the authorization domain, or by a combination of the above.

**Definition 3.7 (Authorization Domain)** *An authorization domain is a pair $(ad\_id, ad\_spec)$, where $ad\_id$ is the authorization domain identifier, and $ad\_spec$ is an authorization domain specification defined according to the following rules: i) each set of object identifiers is an authorization domain specification; ii) each concept query is an authorization domain specification; iii) if $as_1$ and $as_2$ are authorization domain specifications, then ($as_1$ AND $as_2$), ($as_1$ OR $as_2$) and NOT $as_1$ are authorization domain specifications.*

**Example 3.6** `(ad`$_1$`, 'Esprit Project Budget' OR 'Italian Project Budget')`, `(ad`$_2$`,`$\{io_1, io_6\}$`)`, *are examples of authorization domains, where $io_1, io_6$ are information object identifiers.*

Authorization domains are defined by the Security Administrator. To regulate information object inclusion into domains, each authorization domain can have an associated list of credentials, denoting the users that are authorized to post their information objects into that domain. Moreover, a user can ask to be notified of the authorization domains in which objects he/she creates can be potentially included. Such domains are the ones whose associated concept query is satisfied by the information object and for which the user has an authorization to post information. The owner of the object can then decide to include his/her object in all the authorization domains, or only in some of them. Due to lack of space, we do not address these aspects in the paper. More details can be found in [1]. In the following $\mathcal{AD}$ denotes the set of authorization domains in the system.

**Definition 3.8 (Access Control Policy)** *An access control policy is a triple ($cred\_expr$,*

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

$ad\_id, m$), *where* $cred\_expr$ *is a credential expression denoting the users to whom the access is granted,* $ad\_id$ *is an authorization domain identifier denoting the objects to which the policy refers to, and* $m \in \mathcal{M}$ *is the mode under which the access is granted.*

**Example 3.7** *The following are examples of access control policies:*

- `((manager AND category = 'senior'), 'Esprit Project Budget' OR 'Italian Project Budget',`[5] `Total_propagate)`*: senior managers can receive objects dealing with Esprit or Italian project budget and see all the links they contain;*
- `((manager AND category='Junior'), 'Esprit Project Budget' OR 'Italian Project Budget', Notify)`*: junior managers can be notified of objects dealing with Esprit or Italian project budget but they cannot access such objects;*
- `((programmer AND project_type = 'Esprit'), ('Esprit Project' OR 'Italian Project') AND NOT ('Esprit Project Budget' OR 'Italian Project Budget'), Notify)`*: programmers can be notified of objects dealing with Esprit or Italian projects as long as those objects do not deal with the budget.*
- `(marketing staff, 'Esprit Project Exploitation Plans', Notify)`*: marketing staff can be notified of objects dealing with exploitation plans;*
- `(patent office, 'Esprit Project Technical Results', Total_propagate)`*: patent office members can access information objects dealing with Esprit project technical results and see the existence of all the links they contain.*
- `(secretary, 'Esprit Project Guideline' AND NOT 'Esprit Project Budget', Selective_propagate)`*: secretaries can access objects dealing with Esprit project guidelines (except the links they eventually contain) but not containing information about the project budget.*

The Access Control Policy Base, $\mathcal{ACPB}$, is the set of access control policies of the system.

# 4 Access Control

When new information objects are acquired or existing information objects are modified, the Filtering Module is activated, which, on the basis of the interest profiles stored in the Profile Base, selects the users that should be notified. The output of the Filtering Module is thus a set of pairs $(u, O)$, where $u$ is a user identifier and $O$ is a set of object identifiers. After the filtering phase, the notification part of the access control module is activated. This part of the module decides for which of the objects in $O$, user $u$ should receive a notification, on the basis of the credentials associated with user $u$ and of the policies stored in the $\mathcal{ACPB}$. After being notified of the new or modified information objects of interest, a user may explicitly request to receive specific objects. The user will receive a copy of a requested object, only if the user has a propagation access mode on the object.

Due to space limitations, we do not discuss further the propagation algorithm [1], but focus on the notification algorithm which involves a large number of users and objects. In the rest of this section, we first introduce auxiliary data structures for the efficient implementation of access control and then present the notification algorithm.

## 4.1 Auxiliary Data Structures for Access Control

For each user $u$, we maintain a list $AD(u)$ that contains the authorization domains $u$ belongs to. In particular, let $CT(u)$ be the credentials associated with $u$ in $\mathcal{PB}$. For each user $u$, $AD(u) = \{\ ad\_id \mid \exists$ $(cred\_expr, ad\_id, m) \in \mathcal{ACPB}$ such that the credentials in $CT(u)$ satisfy $cred\_expr\ \}$. $AD(u)$ is computed for each user $u$, when $u$ subscribes. $AD(u)$ is expected to be seldomly updated when either (a) the credentials of user $u$ are modified, or (b) there is some change in the $\mathcal{ACPB}$.

To avoid re-evaluating the credential expressions for all triples in $\mathcal{ACPB}$, for computing $AD(u)$, when a user $u$ enters the system for the first time or modifies his/her credentials, we maintain indexes on the credential expressions in the $\mathcal{ACPB}$. In effect, we index the $\mathcal{ACPB}$ based on the type of credentials that appear in the credential expressions of each triple in the base. Then, when a user subscribes or updates its credentials, only the affected triples are checked. In particular, for each credential type $ct\_id$ $in\ \mathcal{CTI}$, we maintain the set $ACP(ct\_id)$ of the associated access control policy triples, i.e., the triples in which $ct\_id$ or its attributes appear. Special care must be given to negative terms of the form NOT $ct\_id$ ($ct\_id \in \mathcal{CTI}$). Such terms are satisfied by users not having a credential of type $ct\_id$ or of a subtype of $ct\_id$. Thus, $ACP(ct\_id)$ is defined as follows: $ACP(ct\_id) = \{(cred\_expr, ad\_id, m) \mid$ $\{(cred\_expr, ad\_id, m) \in \mathcal{ACPB}$ and either $ct\_id$ appears in $cred\_expr$ or an attribute name $a \in \mathcal{AN}$ of a non-inherited attribute of $ct\_id$ appears in $cred\_expr\} \cup$ $\{(cred\_expr, ad\_id, m) \in \mathcal{ACPB}$ and a term NOT $c$, $c\ in$ $\mathcal{CTI}$, appears in $cred\_expr$ where $c \neq ct\_id$ and $c$ is not a subtype of $ct\_id\}$.

**Example 4.1** *Figure 3 depicts the part of the index for the access control policies of Example 3.7. Now assume that a new user* $u'$ *with a single credential type* `secretary` *enters the system. To evaluate* $AD(u')$*, we just evaluate the*

[5] For the sake of clarity, we use the concept query specifying the authorization domain instead of the authorization domain identifier.

*credential expressions in the access control policies indexed by the entry* `secretary` *and its supertypes.*

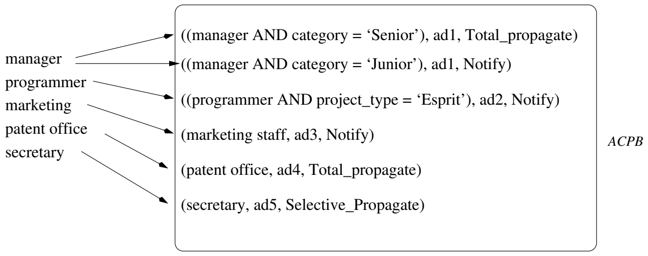


**Figure 3. Indexing the $\mathcal{ACPB}$ . Authorization domains identifiers are used; concept queries for the domains are as in the access control policies in Example 3.7**

Analogously, to efficiently update the $AD(u)$ sets when there is an update in the $\mathcal{ACPB}$, we may index users based on the credential types in their profiles. In this case, when there is an update in a credential expression of a triple $\in \mathcal{ACPB}$ or a new triple is added, we use the index to re-evaluate the sets $AD(u)$ only for the users $u$ that have a related credential type in their profiles.

**Example 4.2** *Assume that we add the access control policy* `(programmer, ('IST Project' OR 'Italian Project'), Notify)` *to the $\mathcal{ACPB}$. If there were no indexes, the sets $AD(u)$ should have to been re-evaluated for all users $u$ in the system. With the index, we re-evaluate the set $AD(u)$ only for those users that are programmers.*

A computationally intensive step of the notification algorithm is determining the set of authorization domains to which a new or modified object belongs to. The straightforward method is to evaluate (or re-evaluate) the concept queries of all authorization domains for each object $o$. To reduce the number of concept queries that need to be evaluated, we maintain inverse indexes similar to those used in tools like SIFT [8]. In such tools, the filtering module uses an inverted index for the interest profiles so that for each new information object $o$ only a subset of the profiles is tested. The idea is, for each concept, to collect in a list all interest profiles (administration domains in our case) that refer to it. Thus, an administration domain with $K$ associated concepts will appear in $K$ lists. When processing an object $o$, we just need to check those authorization domains in the inverted lists of concepts appearing in $o$.

Several index structures are possible [8]. For example, assume that a concept query $cq$ is in disjunctive normal form, $cq$: $cq_1$ OR $cq_2$ OR ... $cq_n$. Then we could select just one concept, a key, from each $cq_i$ and insert the concept query only in the inverted list of these key concepts. For

**Algorithm 4.1 An Implementation of the Notification Algorithm**

INPUT: 1) A set $S$ of pairs $(u, O)$, where $u \in \mathcal{U}$ and $O$ is a set of object identifiers and the set $\mathcal{IO}$ of updated or new objects ($\mathcal{IO} = \cup_{(u,O) \in S} O$).
2) $\mathcal{AD}$, $\mathcal{PB}$, and $\mathcal{ACPB}$
3) For each $u \in \mathcal{U}$, the set $AD(u) = \{\, ad\_id \mid \exists\, (cred\_expr,\ ad\_id,\ m) \in \mathcal{ACPB}$ such that the credentials in $CT(u)$ satisfy $cred\_expr \}$.
OUTPUT: A set $S'$ of pairs $(u', O')$, such that $(u, O) \in S$, $u = u'$ and $O' \subseteq O$ is the set of objects in $O$ for which $u$ has to be notified
METHOD:

**Variation 1**

$S' := \emptyset$
1. **For** each new or updated information object $o \in \mathcal{IO}$:
   compute $\Pi_o(\mathcal{AD}) = \{ad\_id \mid (ad\_id,\ spec) \in \mathcal{AD}$ and $o$ satisfies $spec\}$
2. **For** each $(u, O) \in S$:
   (a) $P' := \emptyset$
   (b) **For** each $o \in O$:
      Compute $A := \Pi_o(\mathcal{AD}) \cap AD(u)$
      If $A \neq \emptyset$ add $o$ to $O'$
   **endfor**
   (c) Add $(u, O')$ to $S'$
**endfor**

**Variation 2**

$S' := \emptyset$; **For** each $ad\_id$, $(ad\_id,\ spec) \in \mathcal{AD}$, $\Pi_{ad\_id}(\mathcal{IO}) := \emptyset$
1. **For** each new or updated information object $o \in \mathcal{IO}$:
   If $o$ satisfies $spec$ in $(ad\_id,\ spec) \in \mathcal{AD}$:
      add $o$ to the set $\Pi_{ad\_id}(\mathcal{IO})$
2. **For** each $(u, O) \in S$:
   (a) $L := \emptyset$
   (b) **For** each $ad\_id \in AD(u)$: $L := L \cup \Pi_{ad\_id}(\mathcal{IO})$
   (c) $O' := L \cap O$
   (d) Add $(u, O')$ to $S'$
**endfor**

**Figure 4. The notification algorithm**

each concept query, the key concepts may be selected randomly or by using additional information, e.g., it may be the least frequently appearing concepts. Reducing the number of concept queries to be evaluated is harder when the concept query involves negation. If a concept query includes at least one non-negative term in each $cq_i$ we can select this as the key concept. However, indexing a query when there is no such term may not be possible. We do not consider such queries in our system. We expect concept queries to be used to grant access to information objects based on user credentials and negation to be used just to exclude some of the qualifying information objects. Finally, notice that the inverted indexes in tools like SIFT are built on interest profiles while in our case are built on authorization domains. Since, the number of authorization domains is expected to be much smaller than the number of interest profiles, evaluating concept queries in our case is a less time-consuming task.

## 4.2 The Notification Algorithm

Given the set $AD(u)$ and the evaluation of concept queries, we now provide an outline of the notification algorithm. The algorithm receives as input the set $S$ of pairs $(u, O)$ returned by the Filtering Module, where $u$ is a user and $O$ is a set of identifiers of objects that are of interest to user $u$. In addition, the algorithm receives as input the set $\mathcal{IO}$ of the object identifiers of all new or updated information objects, the set of authorization domain identifiers $\mathcal{AD}$, the Profile Base $\mathcal{PB}$, the Access Control Policy Base $\mathcal{ACPB}$, and sets $AD(u)$ for each user $u$. The algorithm returns for each user in $S$ the objects in $O$ for which he/she has a notification right.

Two variations of the algorithm are depicted in Figure 4 based on different outputs produced during the evaluation of concept queries (step 1). In Variation 1 of Algorithm 4.1, we assume that the output of the first step is: for each new or updated object $o \in \mathcal{IO}$, the set $\Pi_o(\mathcal{AD})$ of the authorization domains $o$ belongs to. That is, $\Pi_o(\mathcal{AD}) = \{ ad\_id \mid (ad\_id, spec) \in \mathcal{AD}$ and $o$ satisfies $spec\}$. In Variation 2, we assume that the output of the first step is: for each affected authorization domain $ad\_id$, $(ad\_id, spec) \in \mathcal{AD}$, the set $\Pi_{ad\_id}(\mathcal{IO})$ of all new or updated objects that are added to it. That is, $\Pi_{ad\_id}(\mathcal{IO}) = \{o \mid o \in \mathcal{IO}$, $o$ satisfies $spec$ of $(ad\_id, spec) \in \mathcal{AD}$ $\}$.

We compare the performance of the second step of the two variations. We assume that the input sets of the set operations (step 2(b) of Variation 1 and steps 2(b) and 2(c) of Variation 2) are ordered so that the operations take time linear in the size of the input sets.

Let $M$ be the number of objects of interest, $N$ the number of users and $D$ the number of authorization domains. Assume that in the average, each object belongs to $k$ authorization domains, that is, the average size of $\Pi_o(\mathcal{AD})$ is $k$. Then, the average size of $\Pi_{ad\_id}(\mathcal{IO})$ is equal to $(k\,M)/D$. Further, assume that for each user, we have an average number $m$ of objects of interest and in the average, each user belongs to $d$ authorization domains.

The complexity of the second step of Variation 1 of Algorithm 4.1 is: O($N\,[m\,(k+d)]$).

The complexity of the second step of Variation 2 of Algorithm 4.1 is: O($N\,[((d\,k\,M)/D) + m]$).

Assume that from the $M$ new objects, in the average, each user is interested in a percentage $c$ ($0 \leq c \leq 1$) of them, that is, $m = c\,M$. Then, if we also consider $d$, $D$, and $k$ to be constants, the complexity of both algorithms is $O(N\,m)$, which is the optimal, since in the worst case the size of the output is $N\,m$, that is, each user is entitled to be notified for all objects of interest. In practice, the first variation works better when $c$ is small, while the second one works better when both $c$ and $D$ are large.

**Example 4.3** *Let us demonstrate how the algorithm works for pair $(u_1, O) \in S$ with $O = \{o_1, o_2, o_3, o_4\}$, $AD(u_1) = \{ad_2, ad_3\}$ and $\mathcal{IO} = \{o_1, o_2, o_3, o_4, o_5, o_6\}$. Let the output of the first step of Variation 1 be: $\Pi_{o_1}(\mathcal{AD}) = \{ad_1, ad_4\}$, $\Pi_{o_2}(\mathcal{AD}) = \{ad_1, ad_4\}$, $\Pi_{o_3}(\mathcal{AD}) = \{ad_2, ad_5\}$, and $\Pi_{o_4}(\mathcal{AD}) = \{ad_1, ad_2, ad_3\}$. Correspondingly, the output of the first step of Variation 2 would be: $\Pi_{ad_1}(\mathcal{IO}) = \{o_1, o_2, o_4\}$, $\Pi_{ad_2}(\mathcal{IO}) = \{o_3, o_4, o_5\}$, $\Pi_{ad_3}(\mathcal{IO}) = \{o_4\}$, and $\Pi_{ad_4}(\mathcal{IO}) = \{o_1, o_2, o_5, o_6\}$. During the second step of Variation 1, for each $o \in O$, we compute $\Pi_o(\mathcal{AD}) \cap AD(u_1)$ and notify $u_1$ about $o$ only if the result is not the empty set. For instance, $\Pi_{o_1}(\mathcal{AD}) \cap AD(u_1) = \emptyset$, thus $u_1$ is not notified for $o_1$, while $\Pi_{o_3}(\mathcal{AD}) \cap AD(u_1) = \{ad_2\}$ and $u_1$ is notified for $o_3$. During the second step of Variation 2, we first compute set $L$. $L$ is computed by taking the union of sets $\Pi_{ad\_id}(\mathcal{IO})$ for all $ad\_id$ in $AD(u_1)$. For this example, $L = \Pi_{ad_2}(\mathcal{IO}) \cup \Pi_{ad_3}(\mathcal{IO}) = \{o_3, o_4, o_5\}$. User $u_1$ will be notified for the objects in $L \cap O$ that is for $\{o_3, o_4\}$.*

## 5 Conclusion

In this paper, we have presented an access control model which provides a set of features specifically tailored for data dissemination systems, such as authorization domains, and a flexible specification of authorizations based on the characteristics of users and on information object contents. We are currently extending our work in several directions. First, we are implementing the proposed model to investigate its performance. Then, we plan to develop an enhanced version of the model which provides support for XML documents.

## References


[1] E. Bertino, E. Ferrari, and E. Pitoura. An Access Control Mechanism for Large Scale Data Dissemination Systems. Technical Report, Extended version of this paper, Department of Computer Science, University of Milano, 2000.

[2] D. Gifford, R. Baldwin, S. Berlin, and J. Lucassen. An Architecture for Large Scale Information Systems. In *Proc. of the Symposium on Operating System Principles*, pages 161–170, 1985.

[3] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of ACM*, 35(12):61–70, 1992.

[4] R. Holowczak. Extractors for Digital Library Objects. Phd thesis, Rutgers University, MS/CIS Department, 1997.

[5] H. Jiang and A. Elmagarmid. WVTDB - A Semantic Content-Based Video Database System on the World Wide Web. *IEEE TKDE*, 10(6):947–996, 1998.

[6] Proc. of the 1st ACM Workshop on Role-Based Access Control. Fairfax (VA), 1996.

[7] P. Samarati, E. Bertino, and S. Jajodia. An Authorization Model for a Distributed Hypertext System. *IEEE TKDE*, 8(4):555–562, 1996.

[8] H. G.-M. Tak W. Yan. Index Structures for Selective Dissemination of Information Under the Boolean Model. *ACM TODS*, 19(2):332–364, 1994.

[9] M. Winslett, N. Ching, V. Jones, and I. Slepchin. Using Digital Credentials on the World-Wide Web. *Journal of Computer Security*, 5, 1997.

[10] T. W. Yan and H. Garcia-Molina. The SIFT Information Dissemination System. *ACM TODS*, 24(4):529–565, 1999.