AN ACCESS CONTROL MODEL FOR DATA ARCHIVES

P. Bonatti[1] E. Damiani[1] S. De Capitani di Vimercati[2] P. Samarati[1]

(1) Dip. di Tecnologie dell'Informazione – Università di Milano - 20163 Crema, Italy

(2) Dip. di Elettronica per l'Automazione - Università di Brescia - 25123 Brescia, Italy

Abstract We present an approach to enforce access control at data archives that need to make their data selectively available on the Web. The paper discusses protection requirements and access control policies for regulating access to the stored data. It presents a model for enforcing access control regulations and a related language for expressing these regulations.

1. INTRODUCTION

Today's society places great demand on the dissemination and sharing of information. With the development and wide spread use of the Internet and the World Wide Web, that allow for convenient electronic data storage and distribution, organizations in the private and public sectors are more and more required to make their data available to the outside world. An ever increasing amount of data is today collected by statistical agencies and census bureaus for analysis and subsequent distribution to the general public or to specific organizations (e.g., research institutions, government offices). *Data producers* can release the data produced directly, as in the case of national statistical institutions, or exploit the mediation of archive institutions (*data publishers*) that collect data from various sources for their subsequent distribution.

This data distribution process is clearly selective: data cannot just be released to anybody. Rather, specific data can usually be released only to specific requesters or under specific conditions [2, 8]. For instance, there are sensitive data that can be released only to specific individuals and/or for specific purposes (e.g., health data collected from hospitals and which must be made available to health care institutions or related partners for research purposes). There are data which are subject to embargoes and can be released to the general public only after a specific time; there are data that can be released only for non-commercial purposes; and data which do not bear sensitivity, but whose release is subject to payment. Many and many other examples can be mentioned, but these few can already give an idea of the variety of protection requirements that may need to be enforced. This situation calls for the need of powerful and flexible access control systems able to capture and enforce the different requirements that the data producers (or publishers) may need to enforce on the data access. While flexible and expressive enough, the access control system should remain simple, easy to manage, and efficient. In particular, we have identified the following characteristics that the access control system should provide.

- The model should support access restrictions based on the typical abstractions used by data producers and publishers, which can define categorizations of users, purposes of use, types of operations, and data objects. These categories should be definable by the data publisher, and hierarchical structures [8] should be supported.
- The model on which the system is based should support restrictions based on conditions on metadata describing (meta)properties of the stored data and the users, which can be represented through profiles maintained at the system.
- The language to express access control rules should have a declarative form. The use of a declarative language makes it easier the task of specifying access restrictions and keeping control over them.
- The language should be simple and expressive. It should be simple to make the management task of specifying and maintaining the security specifications easy, as well as keeping syntax checking time reasonable. It should be expressive to make it possible to specify, in a flexible way, different protection requirements that may need to be imposed on different data.
- Last but not least, the language should be easy to use to nonspecialists in the field. We could imagine that often, the people specifying the security policies will be employees unfamiliar with procedural or logic-based languages. Therefore, while providing expressive power and unambiguity of these paradigms, the language should however be based on a high-level formulation of the access control rules, possibly close to natural language formulation.

Although many access control models and systems have been proposed [11], current proposals do not completely satisfy the characteristics above. For instance, while most regulations by data producers/publishers make data release conditioned on the use that the recipient will do of the data, use-based restrictions are not supported by current access control systems. While more recent logic-based authorization languages (e.g., [8]) could provide the expressive power to capture these requirements (or be enriched for that), the resulting system would be too complex to use and manage.

In this paper, we present an access control model regulating access to a data archive together with a language for the specification of security requirements. The language allows data publishers (in the case where data are being distributed by the producer directly, the publisher is the producer itself) to state to whom, how, and under which conditions specific data can be accessed. While expressive and flexible enough to capture the different protection requirements that may need to be imposed on the data, the system remains simple and easy to use.

2. DATA MANAGEMENT AT THE ARCHIVE

The data archive maintains data collected from the different producers for their subsequent distribution. Besides these actual data, called *datasets*, the archive also maintains a collection of *metadata* representing information associated with datasets. We describe datasets and metadata in more details.

2.1. DATASETS

Datasets are data collected from the producers for distribution. Usually, they represent statistical information organized via tables (tabular data) and can be in the form of *microdata*, reporting information of individual respondents, or *macrodata*, e.g., aggregates combining data of different respondents [6]. For the purpose of this paper, we consider data to have already undergone the statistical disclosure control necessary to sanitize data by removing explicit identities of the data respondents, or the possibility of inferring them [7, 10]. Datasets can be organized in abstractions defining groups of datasets that can be collectively referred together with a given name. Groups can reflect the file system organization in directories and/or orthogonal abstractions defined by grouping datasets with common characteristics. Dataset groups need not be disjoint and can be nested. Datasets with their groups define a partial order that introduces a hierarchy [8]. This hierarchy can be depicted as a directed acyclic graph whose nodes are the datasets and groups thereof and an arc from node n_1 to node n_2 indicates a *direct* (i.e., explicitly defined) membership of n_1 in n_2. Figure 1 illustrates an example of datasets hierarchy, where, for simplicity, the datasets leaves are omitted. The hierarchy divides datasets into two groups: `Free_Datasets` (reporting public data) and `Restricted_Datasets` (which cannot be made available to the general public). In turn `Restricted_Datasets` are organized in `EU_Datasets` (reporting statistics of countries within the European Community) and `Non-EU_Datasets` (reporting statistics of other countries). In the following, we assume the hierarchy to be rooted, meaning there is one element to which all datasets belong. This assumption is not limiting (a dummy node to which all elements belong can be assumed) and is common in many systems [8].

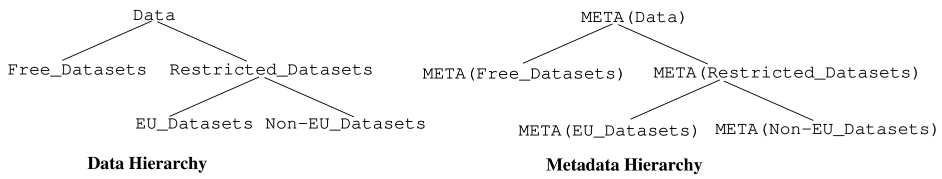

Figure 1 An example of data and metadata hierarchies

2.2. METADATA

Metadata represent data about data [1]. They are not part of the dataset content; they provide additional contextual information on datasets that can be provided to users and can help them in browsing through the system (searching for specific data). For instance, metadata can report to which study a dataset is referred, how and when it was obtained, by whom, and so on. Although several standards have been proposed for interoperable metadata interchange in the digital libraries domain (e.g., Z39.50, Dublin core, and RDF, [3]) few attempts have been made at employing such standards for the description of statistical information [5]. Rather, information retrieval techniques have been used to extract general-purpose descriptors from available data. In our approach, no assumption is made about metadata syntax and semantics; metadata are assumed to be available in the form of textual or semistructured documents (e.g., XML [1] or DDI [12]). Generic XML-based semistructured documents naturally support heterogeneous metadata formats as they have no fixed structure: the structure can be absent, irregular, or incomplete. Intuitively, a semistructured document can be seen as a set of element properties, possibly nested (element-subelement relationship). Whatever their form, metadata are, at a practical level, files associated with datasets. Metadata are associated only with specific datasets, and not with abstractions on them. Also, no hierarchy is explicitly defined on metadata. However, the abstraction hierarchy defined on the data reflects in an abstraction hierarchy on the corresponding metadata (see Figure 1). We assume a bijective function META() that makes the association between a dataset (or groups thereof) and its metadata (or groups thereof). For instance, given a dataset `dataset1`, function META(`dataset1`) refers to the metadata associated with it. Given a dataset group `Free_Datasets`, function META(`Free_Datasets`) denotes the set of all metadata of the datasets in the group. A metadata document can then be referenced either through its identifier or, via function META, through the identifier of the dataset with which it is associated.

For metadata browsing by users and (as we will see in Section 4) for the evaluation of conditions that may determine whether or not a given access to datasets can be allowed, it is useful to evaluate the content of metadata. For instance, a user may require access to all datasets produced in the current year (where year is a property in the metadata). The same property can be exploited in the specification of security restrictions by a rule limiting access to datasets produced in the current year to a restricted set of users. For semistructured metadata, we support these features by allowing reference to fine-grained content at the level of properties. Properties (elements and attributes, in the XML terminology) within a metadata document are referenced by means of *path expressions*, stated in an appropriate language, for example XPath [14]. Basically, a path expression is a sequence of element names separated by character `/` (slash): $l_1/l_2/\ldots/l_n$. Intuitively, semistructured documents can be seen as trees, where each node represents an element or attribute of the considered document and an edge between two nodes represents a containment relationship between them. A path expression $l_1/l_2/\ldots/l_n$ on a document tree then represents all the attributes or elements named l_n that can be reached by descending the document tree along the sequence of nodes named $l_1, l_2, \ldots, l_{n-1}$. For instance, path expression META(`dataset1`)`/codeBook/stdyInfo/subject` identifies the elements `subject` (describing the topic of a study) within element `stdyInfo` of element `codebook` in the metadata associated with dataset `dataset1`. Path expressions may also include conditions associated with the nodes of a path; in this case the path expression identifies the set of nodes that satisfy all the conditions. Conditions greatly enrich the power of the language, and are a fundamental component in the construction of a sophisticated authorization mechanism. The *conditional expressions* used to represent conditions may operate on the "text" of elements (i.e., the character data in the elements) or on names and values of attributes. Conditions are distinguished from navigation specifications by enclosing them within square brackets. Given a path expression $l_1/\ldots/l_n$ on a document tree, a condition may be defined on any label l_i, enclosing in square brackets a separate evaluation context containing a predicate that compares the result of the evaluation of the relative path expression with a constant or another expression. Conditional expressions may be combined via `and` and `or` operators to build boolean expressions. Multiple conditional expressions appearing in the same path expression are considered to be `and`ed (i.e., all the conditions must be satisfied). For instance, expression `/codeBook//styInfo[./subject/keyword = "private schools"]/sumDscr/[./collDate = "2000-07-05"]` identifies the `sumDscr` element of all the studies whose date of collection is July 5, 2000 *and* one of the salient aspects of the studies's content is `private schools`.

2.3. ACCESSING DATA

Datasets stored at the archive, and metadata associated with them, can be accessed by users via different actions that can be executed on the datasets/metadata. The specific actions supported by a server may vary depending on the functionalities provided on specific kinds of datasets. Among the actions supported, we can distinguish the following three categories (which can correspond to a single action or groups of them):

Browse to visualize and query metadata associated with datasets. With browsing operation, users can walk through the metadata to choose the actual dataset they are interested in.

Analyze-on-line to query datasets. On line analysis includes a set of pre-defined operations that perform on-line calculations on selected data. Available operations may vary depending on the kind of dataset under consideration, and may include basic statistical methods such as: n-way cross tabs, breakdown analysis, correlation, and regression [9].

Download to download data from the server. It allows users to save whole datasets on their local machine to perform off-line analysis.

Further abstractions (or specializations) can be defined on actions, to allow references to groups of actions via a single name. For instance, the three categories of actions above can all be grouped in a set called `access` and thus referred to as one. In this way, granting a user privilege `access` to a given dataset will give the user the privilege of executing any action on it.

3. SUBJECT CHARACTERIZATION

We now discuss the characterization of subjects (data requestors) to the purpose of enforcing restrictions on actions that they can execute on the datasets/metadata.

3.1. REQUESTORS

Subjects are entities requesting access to data. The basic concept for the characterization of a subject is the person presenting the request, which is usually referred to as *user*. Users are human entities that can connect to the system and make requests. Each user has associated an identifier (usually the user's login registered at the server), with which the user is referred to in the system.

Although requests are actually typed in by a human user, the decision of whether some data may or may not be released does not depend only on the requesting user's identity but also on the use that the user intends to do of the data being requested, and that can be declared by the user at the time of the request. As a matter of fact, from the analysis of traditional paper world and electronic-based practices at the data archives consulted, it appears clear that the use for which the data are being requested plays an important role in the decision of whether the data can or cannot be released. Although not supported in current access control systems, use-based access restrictions appear to be one of the basic requirements that should be addressed in data dissemination [13]. From an analysis of current practices, we have identified two ways in which the use can be characterized: *purpose* and *project*. Purpose is the reason for which data are being requested and will be used. Examples of purposes are: `Research`, `Commercial`, `Teaching`, or `Personal_interest`. A project is a named activity registered at the server, for which different users can be subscribed, and which may have one or more purposes. As an example, one or more organizations involved in a given research project can register the project to the archive so that all users working on it (as specified by the authority registering the project at the archive) can, in the execution of the project's activities, enjoy the project's privileges for accessing data maintained at the archive.

Accordingly, we characterize each subject making a request to the data publisher server with a triple $\langle$*user, project, purpose*$\rangle$ stating that *user* is requesting an access for a given *project* and/or a given *purpose*. Access requests are then characterized by the subject requesting access, the action requested, and the object on which the action is requested. Some elements within the subject triple may remain unspecified with respect to a given request. This is, for example, the case of requests made by users who do not belong to any project or who do not declare the purpose for which the data are being requested. Identity also can remain unspecified, as in the case of anonymous requests.

Example 3.1 Examples of access requests are as follows.

- $\langle$`tom.smith,FASTER,research`$\rangle$, `download`, `dataset1`

  user `tom.smith` requires to `download` `dataset1` for `research` purposes within the `FASTER` project.

- $\langle$`john.doe,_,commercial`$\rangle$, `download`, `dataset1`

  user `john.doe` requires to `download` `dataset1` for `commercial` purposes.

- $\langle$`_,_,_`$\rangle$, `browse`, `meta_dataset5`

  an anonymous user with undeclared project and purposes requires to `browse` metadata `meta_dataset5`.

3.2. SUBJECT ABSTRACTIONS AND PROFILING

Besides their identities or declared project and purpose (composing the request), subjects are characterized at the server by additional information, such as membership in groups or satisfaction of given properties, which may affect their ability to access data. We now discuss the definition and organization of subject related information.

3.2.1 Subject abstractions. Abstractions allow the grouping of users, projects, and purposes, respectively, with common characteristics, and referencing to the groups with a name. For instance, with respect to projects, abstractions can group together all the projects registered by a given organization, all the projects sponsored by a national institution, or all the projects with commercial goals. With respect to purposes, abstractions can correspond to generalization/specialization relationships. For instance, `pure research` and `applied research` can be seen as a specialization of `research`. With reference to the user domain, abstractions allow

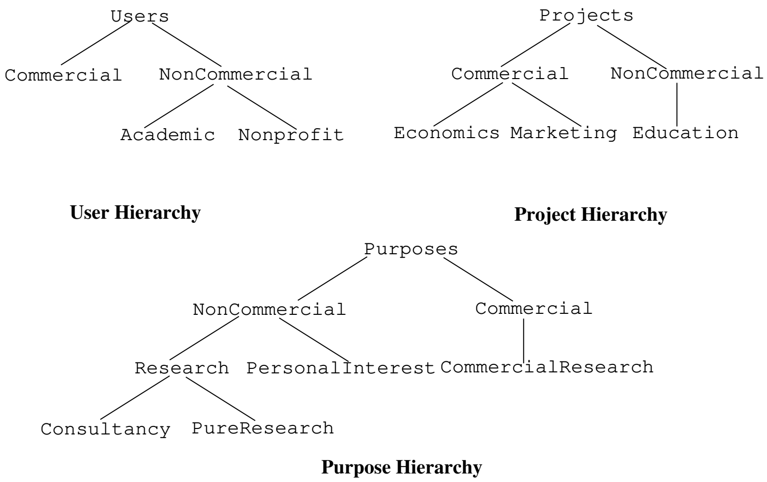

Figure 2 An example of user, project, and purpose hierarchies

the definition of groups, representing named sets of users, as usually supported in current access control systems [8, 11]. At a very high level, groups can distinguish the different communities of users who may need access to a data archive, such as: academic community, policy making community, mass media community, and commercial community [9]. Specializing these communities, we can obtain finer grained or orthogonal classifications of the users. For instance, going at a finer grain we can distinguish, within the academic community, groups `private_schools` vs `state_schools`, or `Faculty` and `Students`. At an orthogonal level, we could also classify users based on other aspects such as their citizenship (e.g., `EU_citizens` vs `Non-EU_citizens`).

Abstraction groups can be nested (i.e., groups can be defined as members of other groups) and need not be disjoint (e.g., a user can belong to more that one group). The membership relationship between abstraction groups introduces then a hierarchy (partial order) on the domains of users, projects, and purposes. Figure 2 illustrates an example of users, projects, and purpose hierarchies, where, for simplicity, only the abstractions are depicted and leaf nodes (corresponding to individual users, projects, and purposes) are omitted instead.

3.2.2 Users and projects profiles. The data publisher server recognizes only users and projects registered at the server. Each user and project is assigned an identifier that allows the server to refer to the user (project, resp.). Besides their identifiers, users and projects registered at the server usually have other properties associated with them. For instance, a user may have properties such as name, address, and occupation; a project may have properties such as title, abstract, and sponsor. To capture and reason about these properties we assume each user

User profile
Name
Login
Title/Position/Job
Address
Email
Telephone
Validation information
Groups
Purposes
Agreement
On line agreement
Registration Date

Project profile
ID
Title
Abstract
Objectives
Period
Purposes
Sponsor
Leaders
Participants
Responsible institution

Figure 3 An example of user and project profiles

and project is associated with a *profile*. Intuitively, profiles are to users and projects what metadata are to datasets. Properties in profiles allow the enforcement of access restrictions that traverse group boundaries.[1] To be as general as possible, we view profiles as semistructured documents (XML or RDF like [1]). The profile associated with a user (project, resp.) defines the name and value of the properties that characterize the user (project, resp.). The semistructured format of user and project profiles provides flexibility in the definition of meta properties associated with subjects (e.g., certain properties can be specified only for given classes of users). Figure 3 illustrates an example of profile for users and projects.

4. ACCESS CONTROL RULES AND ACU LANGUAGE

In the previous sections, we have discussed the form of access requests and the organization of data and subject-related information at the server. We now present the rules that establish access regulations for subjects to access data, and a language for expressing them. We start by introducing the components of the rules, we then give their format and semantics.

4.1. SPECIFICATION OF SUBJECT, OBJECTS, AND CONDITIONS

The first step in the specification of access control rules is the characterization of the *subjects*, *actions*, and *objects* to which each rule applies, and of possible *con-*

[1]In principle, every property could be supported through groups, but this would require the definition of as many groups as the cardinality of the property domain, with a result that would be rather awkward and impracticable.

Keywords	CAN WITH IF ONLY IF IN AND OR NOT FORMETA PURPOSES PROJECTS	
Reserved identifiers	**user**	bounded to the identity (if defined) of the user making a request
	project	bounded to the project (if defined) specified by the user making a request
	purpose	bounded to the purpose (if defined) specified by the user making a request
	dataset	bounded to the identifier of the dataset to which access is requested
	metadata	bounded to the identifier of the metadata document to which access is requested or associated with the dataset to which access is being requested.

Figure 4 List of keywords and reserved identifiers of the ACU language

ditions under which the specific access can be executed. Actions are characterized simply through the name of the operation or class of operations (in which case the rule applies to all operations in the class). Subjects and objects can also be specified simply by stating an identifier, specifying a given elementary value in the corresponding domain or a named abstraction of values. To provide expressiveness and flexibility, our language also allows the specification of subjects and objects through expressions, where each expression identifies a set of subjects (objects, respectively) that satisfy specific properties. To make it possible in these expressions to refer to the user, project, purpose, data, or metadata involved in the request being evaluated without need of introducing variables in the language [8], we provide the reserved identifiers listed in Figure 4. The appearance of one of such identifiers (e.g., **user**) in an expression is intended to be replaced with the actual parameter of the request (e.g., user requesting access) in the evaluation at access control time. The value is "undefined" in case no value has been declared. Object and subject expressions can also use the keywords listed in Figure 4. The meaning of some keywords is straightforward (e.g., AND, OR, and NOT are boolean operators, and IN denotes membership in abstraction), the meaning of the others will be clear in the following.

4.1.1 Object expressions.

The specification of the objects to which a rule applies is an *object expression* of the form

object-id [WITH *conditional-object-expression*]

where:

- *object-id* is either the identifier of a dataset (or group of datasets) or of a metadata document (or group thereof) together with an optional XPath expression identifying portions of the document. Metadata document can be identified explicitly via their identifier or, via function META, by specifying the name of the datasets (or group thereof) with which they are associated.
- *conditional-object-expression* is a boolean formula of conditions that can evaluate membership of the object in groups, values of properties on metadata, and so on.

Example 4.1 The following are examples of object expressions.

- `Free_Datasets` WITH META(**dataset**)/`producer=ACME`

  it denotes all datasets in the `Free_Datasets` class that are produced by ACME (the producer is specified as a property in the associated metadata).
- META(`Restricted_Datasets`)//`question_text`

  it denotes element `question_text` within the metadata documents associated with datasets in the `Restricted_Datasets` group.

The use of abstractions, reserved identifiers, and path expressions to query metadata provide a flexible and powerful means of identifying via a simple expression a whole set of datasets/metadata, which will turn very convenient in the specification of access rules [4]. In particular, given the reachness of the metadata usually supported [9], expressions allow the specification of access rules applicable only to datasets whose metadata satisfy some conditions. For instance, it allows the enforcement of embargo restrictions, where only datasets collected before a given year can be released.

4.1.2 Subject expressions.

In an analogous way, subjects to which a rule applies are specified as a *subject expression* in the form of a boolean formula of terms that evaluate conditions on the user, project, and purpose of the request. The rule will be applicable only to subjects that satisfy the given conditions, where conditions can evaluate the user's profile or its membership in groups, the project's profile or its membership in a group, and the purpose value or its inclusion in an abstraction. We assume profiles to be referenced with the identity of the corresponding users and projects. Single properties within users and projects profiles are referenced with path expressions denoting the path from the root to the property. For instance, `FASTER/sponsor/address` indicates the address of the sponsor of the FASTER project. Here, `FASTER` is the identity of the project (and therefore the identifier for the corresponding profile), and `sponsor/address` the path name of the address property. Expressions can make reference to the user, project, and purpose involved in the current request via the reserved identifiers **user**, **project**, and **purpose**, respectively (see Figure 4).

Example 4.2 Some examples of subject expressions are as follows.

- **user**/`citizenship=EC` AND (**project**/`sponsor=EC` OR **purpose** IN `research`)

  it denotes requests made by users who are European citizens and intend to use the data for research purposes or within an EC funded project.
- **user** IN `NonCommercial-users` AND **purpose** IN `research`

  it denotes requests made by users belonging to group `NonCommercial-users` that intend to use the data for research purposes.
- **user** IN `NonCommercial-users` AND **purpose** IN `research` AND **project**/`sponsor=EC`

it denotes requests made by users belonging to group `NonCommercial-users` that intend to use the data for research purposes within an EC funded project.

In the case of security constraints applicable to all users within a given group or that request access for a given project or purpose (or group thereof), the group, project, and/or purpose element can be explicitly factorized out of the subject expression and isolated. The explicit reference to users/groups, projects, and purposes allows the indexing of the ACU rules and consequently improves performances in the access control.

Intuitively, a subject expression of the form

- **user** IN user-id AND **project** IN project-id AND **purpose** IN purpose-id AND subject-expression

can be turned into an indexable expression of the form

- user-id OF project-id PROJECTS FOR purpose-id PURPOSES WITH subject expression

where the clauses "OF project-id PROJECTS", "FOR purpose-id PURPOSES", and "WITH subject expression" are optional and can be omitted.

4.1.3 Conditions.

Besides subjects, objects, and actions, access control rules can specify *conditions* defining constraints that the rule requires be satisfy for the request to be granted. Conditions evaluate membership of subjects and objects into classes or properties in their profiles and associated metadata. These are conditions similar to those appearing in subject and object conditional expressions, but which may need to be stated separately (as it will be clear in the next subsection).

4.2. ACCESS RULES

Our system supports two kinds of access rules: *authorizations* and *restrictions*.

Authorizations specify permissions for the access. They have the form

$$\langle subjects \rangle \text{ CAN } \langle actions \rangle \langle objects \rangle \text{ [IF } \langle conditions \rangle]$$

where *subjects*, *actions*, and *objects* identify the requests to which the authorization applies as discussed in the previous section, and *conditions* is a boolean expression of conditions whose satisfaction authorizes the access. Note that conditions can also be included in the expressions specifying the *subjects* and *object* for the rule. An access request is considered to be authorized if at least one of the authorizations that applies to the request is satisfied.

Restrictions specify requirements that *must* be satisfied for an access to be granted. They have the form

$$\langle subjects \rangle \text{ CAN } \langle actions \rangle \langle objects \rangle \text{ ONLY IF } \langle conditions \rangle$$

where *subjects*, *actions*, and *objects* identify the requests to which the restriction applies as discussed in the previous section, and *conditions* is a boolean expression of conditions that every request to which the restriction applies must satisfy; lack to satisfy any of the conditions in restrictions that apply to a given request implies that the request will be denied. Unlike for authorizations, conditions cannot be all incorporated in the subject and object expressions of the rules as this would change the semantics of the restrictions. While conditions appearing in the *conditions* field impose constraints that if not satisfied imply that the access should be denied, conditions in the subject (object) expressions simply limit the requests to which the restriction is applicable. As an example, notice the difference between statements like "Users can access data1 *only if* they are non-commercial and have signed an agreement" and "Users who are non-commercial can access data1 *only if* they have signed an agreement". While the first rule prohibits access to commercial users, the second rule does not.

Authorizations correspond to traditional (positive) rules usually enforced in access control systems [11]. If multiple authorizations are applicable to a given access request, the request can be granted only if at least the conditions in one authorization are satisfied. Therefore, lack to satisfy the conditions in an authorization applicable to a request simply makes the authorization ineffective; but it does not imply that the access will be denied. Intuitively, this means that different authorizations are considered as combined in OR.

The only support of authorizations (traditional open policy) would result however limiting. As a matter of fact, by looking at the specifications of several partners we noticed that often access restrictions are stated in a *restrictive* form, rather than in the *inclusive* positive form just mentioned. By restrictive form we mean rules that state conditions that *must* be satisfied for an access to be granted and such that, if at least one condition is not satisfied, the access should not be granted. For instance, a rule can state that "access to dataset1 can be allowed *only* to citizens". It is easy to see that such a restriction cannot be simply represented as an authorization stating that citizens are authorized. In fact, while the single authorization brings the desidered behavior, its combination with other authorizations may not, leading the *only* constraint to be not satisfied anymore. The combined use of authorization and restrictions easily support both requirements: restrictions specify requirements of the exclusive *only if* form, while authorizations specify requirements in the traditional positive *if* form. Intuitively, restrictions play the same role as negative authorizations (denials) supported by recent access control systems (a restriction is equivalent to a negative authorization where the condition is negated). However, we decided to introduce restrictions as their format appears to be closer to the intuitive formulation of protection requirements in the policies examined. Restrictions are also easier to understand because of the clear separation between subjects to which a restriction applies on the one side and necessary conditions that these subjects must satify on the other side (which, in traditional approaches, would be collapsed into a single field).

As visible from the example below, the specification of authorizations and restrictions while ensuring non-ambiguity and a clear semantics, results very intuitive and close to the natural language formulation of the requirements.

Example 4.3 The following are examples of security requirements and corresponding ACU rules enforcing them.

Rule 1) Everybody can access `Free_Datasets`.

- `Users` CAN `access` `Free_Datasets`

Rule 2) Access to datasets not in Free_Datasets allowed *only* to UK citizens.

- `Users` CAN `access` `data` WITH NOT **dataset** IN `Free_Datasets` ONLY IF **user**/citizenship='UK'

Rule 3) NonCommercial users can download Standard_Datasets if project is Educational and its sponsor is a non-profit organization.

- `NonCommercial-users` OF Educational PROJECTS CAN `download` `Standard_Datasets` IF **project**/sponsor='non-profit'

Rule 4) Users within NonCommercial projects who are employed as faculty members can access Standard_Datasets.

- `Users` OF NonCommercial PROJECTS CAN `download` `Standard_Datasets` IF **user**/title = 'faculty'

4.3. ACCESS CONTROL ENFORCEMENT

The Access Control Unit (ACU) component mediates all the access requests to datasets/metadata and evaluates them against the access rules. As already discussed in Section 3, each access request is characterized by three elements: the *subject* that makes the request (composed of the triple ⟨*user,project,purpose*⟩), the *object* on which the request is made, and the *action* that the subject wishes to perform on the object. For each request received, the access control system first determines all the rules that apply to the request, that is, the rules for which the action field is equal or is an abstraction of the action in the request, and whose subject (object, respectively) expressions are satisfied by the subject (object respectively) of the request. This rule collection process is followed by a conditions packing and evaluation process as follows. All the conditions appearing in the applicable rules are evaluated. According to the given semantics, for the access to be granted *all* the (*only if*) conditions in the restrictions must be satisfied *and* the (*if*) conditions of *at least one* authorization must be satisfied. The system therefore evaluates the satisfaction of the resulting combined condition, substituting true or false for conditions that can be evaluated against profiles and metadata If the required conditions are satisfied the access is granted, it is denied otherwise.

Example 4.4 Consider the access control rules in Example 4.3 and a request by user `Alice` to `download` `dataset1` for `Commercial` purpose within project `Al_Marketing`. Suppose that `dataset1` is a `Free_Datasets`. The access will be granted with no condition according to rule 1.
Consider now a request by user `Bob` to `analyze on line` `dataset2` for `Research` purpose within `Educational` project. Suppose `dataset2` belongs to `Standard_Datasets`. Authorizations 3 and 4 and restriction 2 apply to the request. Accordingly, the access can be granted only if the conditions in the restriction (**user**/citizenship='UK')) *and* the conditions in at least one of the authorizations (**project**/sponsor='non-profit' or **user**/title = 'faculty') are satisfied. Suppose that, according to the profile information, Bob is UK citizen, the sponsor of the project is a non-profit organization (conditions in rules 2 and 3 are true), and `Bob` is a student (condition in rule 4 is false). The restriction and at least one authorization are satisfied and therefore access is granted.

5. CONCLUSIONS

We have presented a model to regulate access to data to be made available for controlled distribution over the Web. The approach is based on a flexible access control model based on a fully-declarative, simple, and expressive language able to express the different protection requirements that may need to be enforced. We are currently extending the language to the consideration of dynamic conditions (e.g., sign agreements) and support of user-system dialog. A prototype implementation is also being developed.

Acknowledgements

The work presented in this paper has been carried out in the context of the FASTER project (`www.faster-data.org`), a project funded by the European Community targeted to the design and development of a secure Web-based system for the selective distribution of data on the Web. We would like to thank all the partners of the FASTER project,[2] and Titto Assini in particular, for useful discussions on protection requirements of data archives and for providing actual access control policies to be enforced.

[2]Partners of the project are: The Data Archive at University of Essex (UK), University of Milan (Italy), Norwegian Social Science Data Services (Norway), Dansk Data Arkiv (Denmark), Centraal Bureau voor de Statistiek (Netherlands), Central Statistics Office (Ireland), Statistik Sentralbyra (Norway), Centre d'Informatisation des Donnees Socio-Politiques (France).

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Academic Press/Morgan Kaufmann, 1999.

[2] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *Proceedings of 7th ACM Computer and Communication Security*, Athens, Greece, November 2000.

[3] Communications of the ACM, April 1998.

[4] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Design and implementation of an access control processor for XML documents. *Computer Networks*, 33(1–6):59–75, June 2000.

[5] C.S. Dippo. Fedstats promotes statistical literacy. *Communications of the ACM*, 41(4):58–60, April 1998.

[6] N. Kirkendall et. al. Report on statistical disclosure limitation methodology. Statistical policy working paper, no. 22, Washington: Office of Management and Budget, 1994.

[7] A. Hundepool and L. Willenborg. μ- and τ-Argus: Software for statistical disclosure control. In *Third International Seminar on Statistical Confidentiality*, Bled, MA (USA), 1996.

[8] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. A Unified framework for supporting multiple access control policies *ACM Transactions on Database Systems*, 2000. To appear.

[9] S. Musgrave and J. Ryssevik. The social science dream machine: Resource discovery, analysis and delivery on the web. In *Social Science Computing Review*. To appear.

[10] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Data and Knowledge Engineering*, 2001. To appear.

[11] P. Samarati and S. Jajodia. Data security. In J.G. Webster, editor, *Wiley Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons, February 1999.

[12] The data documentation initiative codebook DTD - version 1.0, March 2000. http://www.icpsr.umich.edu/DDI/CODEBOOK.TXT.

[13] B. Thuraisingham, S. Jajodia, P. Samarati, J. Dobson, and M. Olivier. Privacy issues in www and data mining: Panel discussion. In S. Jajodia, editor, *Database Security XII - Status and Prospects*. Kluwer, 1999.

[14] World Wide Web Consortium (W3C). *XML Path Language (XPath) Version 1.0*, November 1999. http://www.w3.org/TR/xpath.