# Authorization-Transparent Access Control for XML under the Non-Truman Model

Yaron Kanza, Alberto O. Mendelzon, Renée J. Miller, and Zheng Zhang

Department of Computer Science
University of Toronto,
Toronto, Canada
{yaron, mendel, miller, zhzhang}@cs.toronto.edu

**Abstract.** In authorization-transparent access control, users formulate their queries against the database schema rather than against authorization views that transform and hide data. The Truman and the Non-Truman are two approaches to authorization transparency where in a Truman model, queries that violate the access restrictions are modified transparently by the system to only reveal accessible data, while in a Non-Truman model, such queries are rejected. The advantage of a Non-Truman model is that the semantics of user queries is not changed by the access control mechanism. This work presents an access control mechanism for XML under the Non-Truman model. Security policies are specified as parameterized rules formulated using XPath. The rules specify relationships between elements that should be concealed from users. Hence, not only elements, but also edges and paths within an XML document, can be concealed. The access control mechanism authorizes only *valid queries*, *i.e.*, queries that do not disclose the existence of concealed relationships. The additional expressive power, provided by these rules, over element-based authorization techniques is illustrated and algorithms that check the validity of queries are provided. The proposed access control mechanism can either serve as a substitute for views or as a layer that verifies the specific relationships are concealed by a view.

## 1 Introduction

Access control is a fundamental part of databases systems. The purpose of access control is to protect private or secret information from unauthorized users. Given the status of XML as a standard for storing and exchanging data, the need for XML access control has been recognized and has received a lot of attention [6, 12, 14, 17].

In authorization-transparent access control, users formulate their queries against the database schema rather than against authorization views that transform and hide data. Rizvi *et al.* [24] present two basic approaches to access control in authorization transparent systems. The first approach is referred to as the *Truman model* and the second as the *Non-Truman model* [24]. In the Truman model, an access control language (often a view language) is used to specify what data is accessible to a user. User queries are modified by the system so that the answer includes only accessible data. Let $Q$ be a user query, $D$ be a database and $D_u$ be the part of $D$ that the user is permitted to see, then query $Q$ is modified to a safe query $Q_s$ such that $Q_s(D) = Q(D_u)$.

*Example 1.* Consider a database that contains information on courses in a university. For each course, the system stores information about the students who are enrolled in a course and the grades that they have received. Suppose that a Truman access control model is used to specify that each student is permitted to see only her grades (not the grades of other students). If student Alice poses a query that asks for the highest grade received in one of the courses she is enrolled in, say *Databases 101*, the system will modify the query to return the highest grade that Alice has received in *Databases 101*.

As Rizvi *et al.* [24] point out, using a Truman access control model, the answers to queries may be misleading. A user cannot tell if an answer to a query is correct over the entire database. In our simple example, Alice may be misled into thinking she is the best in the class (after all, she asked for the highest grade over all students).

Misleading answers are prevented by the Non-Truman model, an alternative, authorization transparent model. In the Non-Truman model, a query that violates access control specifications is rejected, rather than modified. Only non-violating, or *valid* queries are answered. Hence, query answers are always the result of applying the user query to the entire database. The Non-Truman model has the desirable property that the semantics of a query is independent of the access control specification. For example, if in Example 1 the system uses a Non-Truman access control model, then the query of Alice will be rejected. Alice will only received answers to queries that are *valid* with respect to the access control policy.

In Non-Truman access control, a fundamental question is the definition of validity. Rizvi *et al.* [24] use a model in which the accessible data is defined using views. Given a database $D$, a query $Q$ is validated by checking whether it could be rewritten using only the authorized views $V$. The rewritten query needs to be equivalent to $Q$ either for all possible database states (referred to as *unconditional equivalence* [30] since it is independent of the current database state $D$) or for only those database states $D'$ for which $V(D) = V(D')$ (termed *conditional equivalence* [30]).

Certainly, such an approach is possible for XML as well. However, results on answering queries using views for small fragments of XML query languages are still emerging [27], and may be undecidable even for the relational model [24]. Furthermore, a view is a positive statement about what data is accessible and it is up to the designer of the view to decide what can be put in the view while still hiding the desired private data. Regardless of the form of the view or access control mechanism, we would like to be able to make statements about what information is *concealed* from a user. In our work, we will specifically consider what it means to conceal a relationship in an XML document.

Information disclosure has been studied formally. Miklau and Suciu [22], define disclosure as exposing information that increases the probability that a user can guess concealed information. There are cases, however, where rejecting a query just because its answer decreases the user's uncertainty about the concealed data is too restrictive [28]. If we consider a relationship, it may be sufficient to ensure that a user cannot distinguish between the current document and other documents that differ from the current document only in the concealed relationship.

*Example 2.* Consider an XML document $D$ that contains information about departments and employees in a company. There is an edge from each department element

$d$ to an employee element $e$ whenever $e$ works in $d$. A company may have an access control policy that permits access to all employees and departments, but that restricts access to the `works-in` relationship. That is, a user should be able to ask queries about employees and departments, but the company may not wish to reveal who works in which department. Perhaps this information may reveal strategic information about the direction of the company.

Intuitively, a query conceals a potential relationship if the query answer does not reveal the presence (or absence) of a relationship in the document. To understand our semantics, consider the following example.

*Example 3.* Continuing our example where the relationship between departments and employees is concealed. Consider a query $Q_1$ that looks for the employees in a specific department $d$, and a query $Q_2$ that looks for all the employees in the company, regardless of their department. Only the answer to $Q_1$ depends on the `works-in` relationship.

In this work, we propose a precise semantics for what it means to *conceal* a relationship. We propose a mechanism for testing whether an XPath query *conceals* a relationship or set of relationships. In particular, we can test whether a view, specified by an XPath query, conceals a relationship.

Our model controls access to relationships. This approach provides a finer granularity than restricting access to elements. On one hand, restricting access to an element is possible in our approach. This is done by concealing all the relationships (edges and paths) to that element. On the other hand, in our approach it is possible to conceal a relationship without restricting access to any of the elements in the relationship. Returning to our example, our rules will permit access to employees and departments while restricting only access to the works-in relationship.

The main contributions of our work are the following.

- The first authorization-transparent, Non-Truman access-control model for XML. Our mechanism is fine-grained and enforces access control at the level of ancestor-descendent relationships among elements.
- A new semantics for concealing relationships in an XML document where a relationship is defined by an edge or a path in the document. To define relationships, we use rules, each containing a pair of XPath expressions.
- We define two forms of query validity. A query is *locally valid* for a document and a set of rules, if it conceals all relationships defined by the rules. Queries may be executed only if they are locally valid. For documents conforming to a schema, we define a stronger form of validity. A query is *globally valid* for a set of rules and a schema if the query is locally valid for the rules and each document that conforms to the schema.
- We consider the problem of testing query validity. Since in our semantics we need to consider only finitely many possible documents, testing is always decidable (unlike in previous approaches [24]) and is polynomial (in the size of the document and the query) for a large class of queries.
- Finally, we show that indeed valid queries do not reveal information about concealed edges.

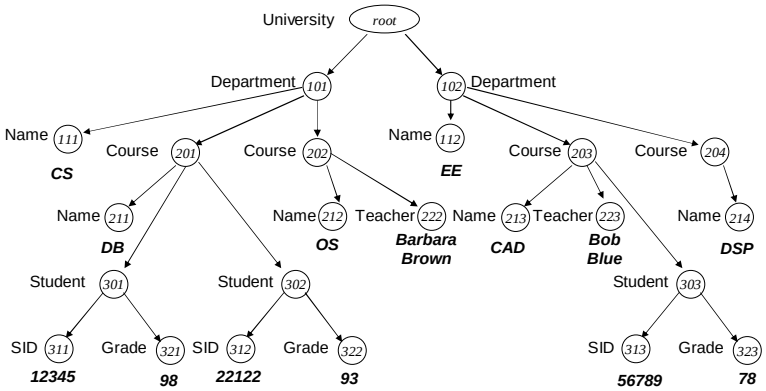


**Fig. 1.** A document that contains information on courses, students and grades in a university.

## 2 Related Work

Most of the work on XML access control uses non-authorization transparent models. For example, Miklau and Suciu define access control rules as XQuery expressions [21]. A user is given a modified document (one in which data is encrypted) and queries are posed on this modified document. They present a new query semantics that permits a user to see only authorized data. An alternative approach proposed by Damiani *et al.* [12, 13] is to compute a secure view by labeling and pruning secure data. Fan *et al.* [14] specify security by extending the document DTD with annotations and publishing a modified DTD. Similarly, work by Bertino *et al.* [4–6] and Finance *et al.* [15] provides XML-based specification languages for publishing secure XML document content, and for specifying role-based access control on XML data [7, 23, 26]. Fundulaki and Marx [16] survey a number of approaches that permit access control to be specified on nodes within a document. Restricting access to nodes has also been used in XACML [18] and XACL [19], two proposed industrial standards. Since non-authorization-transparent models may change the query semantics, other work has focused on efficient query processing for secure queries [9, 29].

In contrast, we present the first authorization-transparent, Non-Truman model for XML. Queries are posed on the original document, so we do not present a model for publishing secure data. Our access control model is a very simple specification of relationships in a document that should be concealed. Our work extends work on information disclosure [3, 28]. In Section 4, we define precisely the relationship with common models of information disclosure such as k-anonymity. Our main focus is to provide a test of query validity that ensures that valid queries effectively conceal any secure relationships. Unlike the non-authorization transparent approaches, we do not change the query semantics.

## 3 Data Model

In this section, we introduce our data model. We assume that the reader is familiar with the notion of a rooted labeled directed graph. We present a rooted labeled directed graph $G$, over a set $L$ of labels, by a 4-tuple $(V, E, r, \textit{label-of}_G)$, where $V$ is a set of nodes, $E$ is a set of edges, $r$ is the root of $G$ and $\textit{label-of}_G$ is a function that maps each node to an element of $L$.

**Document** Let $L$ be a finite set of labels and $A$ be a finite set of atomic values. An *XML document* is a rooted labeled directed tree over $L$ with values of $A$ attached to atomic nodes (*i.e.*, to nodes that do not have outgoing edges). Formally, a document $D$ is a 5-tuple $(X, E_D, \textit{root}_D, \textit{label-of}_D, \textit{value-of}_D)$, where the tuple $(X, E_D, \textit{root}_D, \textit{label-of}_D)$ is a rooted labeled directed tree over $L$, and $\textit{value-of}_D$ is a function that maps each atomic node to a value of $A$. The nodes in $X$ are called *elements*. In order to simplify the model, we do not distinguish between elements and attributes and we assume that all the values on atomic nodes are of type PCDATA (*i.e.*, *String*).

*Example 4.* Figure 1 shows a document that contains information on courses, students and grades. Elements are represented by circles and are numbered, for easier reference. Atomic values appear below the atomic nodes and are written with a bold font.

**XPath** In this work, we use XPath [11] for formulating queries and access control rules. XPath is a simple language for navigating in an XML document. XPath expressions are omnipresent in XML applications. In particular, XPath is part of both XSLT [10] and XQuery [8], the WWW-Consortium standards for querying and transforming XML. Due to lack of space, we do not present the syntax and the semantics of XPath in this paper and we refer the reader to the standard [11].

In XPath there are thirteen types of axes that are being used for navigating in a document. Our focus in this work is on the *child* axis (`/`) and the *descendent-or-self* axis (`//`) that are the most commonly used axes in XPath. Our model, however, can also be applied to queries that include the other axes.

## 4 Concealing Relationships

Before presenting our techniques, we first consider what it means to conceal a relationship. A *relationship* in a document is essentially two sets of elements. For example, consider the university document (shown in Figure 1). The pair $(S, G)$, where $S$ is the set elements labeled "Student" and $G$ is the set of elements labeled "Grade", is a relationship between students and grades. Concealing the relationship between students and grades means that for every grade in the document, the user will not be able to infer (with certainty), from query answers, below which student the grade is. We will want this to be true for all authorized queries (*i.e.*, all *valid queries*).

We also want to have some measure of the uncertainty that is gained by concealing a relationship. Thus, we use a definition that is a variation of $k$-*anonymity* [25] to relationships in a document. In the $k$-anonymity model, the goal is to provide a guarantee that each element cannot be distinguished from at least $k - 1$ other elements. In our case, suppose that we want to conceal in a document $D$ the relationship $(A, B)$. Given

an element $b \in B$, we want to have a subset $A_k$ of $A$ such that the following conditions hold. First, in $A_k$, there are $k$ elements of $A$, and no element is an ancestor of another element. Second, the user will not be able to infer, from answers to valid queries, which one among the $k$ elements of $A_k$ is an ancestor of $b$. To make this more precise, we present a formal definition.

**Definition 1 ($k$-Concealment of a Relationship).** *Consider a set of valid queries $\mathcal{Q}$, a document $D$, and two sets $A$ and $B$ of elements in $D$. The relationship $(A, B)$ is* $k$-concealed *if for every $b \in B$ there exist $k$ elements $a_1, \ldots, a_k$ of $A$ and $k$ document $D_1, \ldots, D_k$ over the element set of $D$, such that the following conditions hold.*

1. *For every $a_i$ and $a_j$ among $\{a_1, \ldots a_k\}$, $a_i$ is not an ancestor of $a_j$.*
2. *Each element $b \in B$ appears in $D_i$ below $a_i$.*
3. *$Q(D) = Q(D_i)$, for every valid query $Q \in \mathcal{Q}$.*

We consider a relationship to be concealed as long as *some* uncertainty remains about the ancestor-descendent relationships. Thus, in the rest of this paper, we will use the phrase "concealing a relationship" for 2-concealment.

Given the definition of concealing relationships, we now turn to the logistics of specifying sets of relationships over XML documents. We will use pairs of XPath expressions for this purpose. Each pair will form an access control rule. The two expressions will define a pair of sets, *i.e.*, a relationship, that should be concealed.

## 5 Access Control Rules

Our approach to access control in XML documents is based on rules rather than views. While views are normally "positive" in the sense that they specify what the user is allowed to know, our rules are "negative" and specify what should be concealed from the user. Our access-control rules specify pairs of elements in the document and by this designate the relationships between these elements as being restricted. In this section, we first present the syntax of rules. Then, we explain why we use rules rather than views. We provide the semantics of rules in our model and define local and global validity. Finally, we briefly discuss the complexity of testing validity.

### 5.1 The Syntax of Rules

Rules are formulated using XPath expressions. Each rule consists of two expressions. The two expressions specify pairs of ancestor and descendent elements that the relationship between them should not be revealed to users. The syntax of a rule is

`for` *path*$_1$ `exclude` *path*$_2$

where *path*$_1$ and *path*$_2$ are XPath expressions. Notice *path*$_2$ is a relative XPath expression w.r.t. *path*$_1$. In the following, we show how to use rules over the university document presented in Figure 1.

*Example 5.* Suppose that we want to prevent queries from disclosing information about what grades were given to which students. This restriction can be specified by the following rule: `for //Student exclude /Grade`.

*Example 6.* Suppose that in the CS department, queries should not disclose information on grades of students as well as the course grades. To set this restriction, two rules are used—the rule from the previous example and the following rule:
`for /Department[Name='CS']/Course exclude //Grade`.

In many scenarios, different users have different accessibilities over the same data. For example, an institution could have a policy that a teacher of some course can have access to all the grades in the course while students can only see their own course grades. Access control rules are parameterized with information specific to a session, such as the user-id, location, date, time, *etc.*, which are instantiated by the system before access control is performed. Parameters are written with a preceding dollar sign.

*Example 7.* Suppose that `$userid` is instantiated to be the current user identification. Consider a policy that only a course teacher is permitted to see the students' grades in her course. This policy is set by the following rule:
`for //Course[not(Teacher=$userid)]/Student exclude /Grade`.

### 5.2 Rules versus Views

We now explain why we use rules instead of views for XML access control in the Non-Truman model. The first reason is that there are many cases where using rules is simpler and requires a more succinct formulation than using views. The following example illustrates such a case.

*Example 8.* Suppose that we only want to prevent users from knowing which student (her SID) is enrolled in which course. Obviously, it is required to have the rule `for //Course exclude //SID`. However, we need another rule `for //Course exclude //Student` to prevent users from inferring the relationship between some course and some SID from the relationships: course-student and student-SID.

Note that these rules should not prevent evaluation of queries that "jump" over a restricted edge. For example, a query that returns the grades of a specific course does not violate the rules. Neither does a query that returns the grades of a specific student.

It is not easy to formulate an XQuery view that preserves all the relationships in the document except for the course-student and course-SID relationships. One example for such a view is a query $Q_{cut}$ that reconstructs the whole document with the following changes. First, student elements should be moved, with all their content, to be below their department element. This cuts the relationship between students and courses but keeps the relationships between departments and students. Second, grade elements should be copied and pasted below their ancestor course elements. We need to duplicate grades, because we need grades to be related to both courses and students. Note $Q_{cut}$ would not work if in the original document, courses have an element named "Grade" as a child. A comprehensive solution [15] is proposed to protect relationships by restructuring the document as well as duplicating and renaming elements of the original document. The resulting view over the document is complicated and cumbersome to formulate in XQuery. Hence it is more error-prone to define access control policies by views than rules in many cases.

The second reason why choosing rules instead of views is that when using views it is difficult to verify that what we want to conceal is indeed concealed.

*Example 9.* Let us continue Example 8. Suppose the queries asking grades of the courses or students are allowed and are defined as views. Do these views really conceal all the relationships between courses and students? Apparently not. Suppose that there is a grade, say 78, that appears only once in the document. Then, knowing who received this grade and in which course this grade was given, it is possible to infer a relationship between a student and a course.

Later in this paper we will present the notion of a coherent set of rules and we will show that when using a coherent set of rules, we can guarantee that restricted relationships are indeed concealed.

The third reason for not using authorization views is that in the Non-Truman model, when using views, testing validity is defined as the problem of answering queries using views. However, query answering using views is not always decidable and has a very high time complexity, as explained in the introduction.

### 5.3 Local Validity

In the Non-Truman model, queries are put through a validity test and evaluated only if the test is passed. We now define the local validity test for queries, given a document and a set of rules. We start by providing some necessary definitions and notations.

**Document Expansion** Consider a document $D = (X, E_D, \mathit{root}_D, \mathit{label\text{-}of}_D, \mathit{value\text{-}of}_D)$. An *expansion* of $D$, denoted as $D''$, is a labeled directed graph that is created by replacing $E_D$ with a new set of edges $E'$, and adding to $D$ a set $E''_D$ of new edges. We call the edges of $E'$ *child edges* and the edges of $E''_D$ *descendent edges*. Formally, the expansion of $D$ is a tuple $((X, E', \mathit{root}_D, \mathit{label\text{-}of}_D, \mathit{value\text{-}of}_D), E''_D)$, where $E'$ is a set of child edges and $E''_D$ is a set of descendent edges. Note that the expansion is not necessarily a tree and is not even required to be connected.

**Transitive Closure** The *transitive closure* of a document $D$, denoted as $\bar{D}$, is a document expansion $(D, E''_D)$, such that in $E''_D$ there is an edge between every two nodes that are connected by a directed path in $D$. The direction of the edge is the same as the direction of the path. Also, $E''_D$ contains an edge from every node to itself. Note that the original edge set of $D$ is not being replaced. As an example, Figure 2(b) shows the transitive closure of the document in Figure 2(a). Child edges are drawn with a solid lines and descendent edges with dashed lines.

The evaluation of an XPath expression over a document expansion is by *following* a child edge whenever a child axis occurs and following a descendent edge whenever a descendent-or-self axis occurs. We explain this in the following example.

*Example 10.* Consider the XPath query `//Department[Name='CS']//Course` over a document expansion $D''$. This query returns course elements $c$ that satisfy the following. There are a department element $d$ and a descendent edge in $D''$ from the root to $d$. There is an element $n$ with label "Name", with value "CS" and there is a child edge in $D$ from $d$ to $n$. Finally, there is a descendent edge in $D''$ from $d$ to $c$. Note that to satisfy the `//` axis we require the existence of a descendent edge rather than the existence of a path between the relevant nodes.

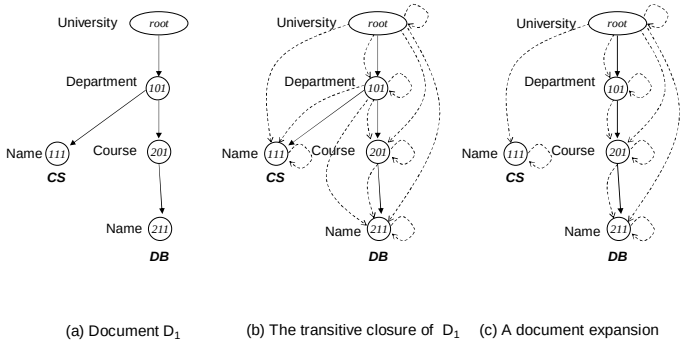


**Fig. 2.** A document $D_1$, the transitive closure of $D_1$ and a document expansion.

It is easy to see that posing an XPath query $Q$ to a document $D$ is equivalent to evaluating $Q$ over the transitive closure of $D$. However, when evaluating $Q$ over a document expansion that is not the transitive closure of $D$, we may get an answer that is different from the answer to $Q$ over $D$.

**Pruning of a Document Expansion** Given a set $R$ of access control rules, a *pruning* of a document expansion $D''$ is a new document expansion, denoted $\textit{prune}_R(D'')$, that is created by removing from $D''$ all the edges (both child edges and descendent edges) that connect a *restricted* pair of nodes. By restricted pair, we mean two nodes whose relationship should be concealed according to $R$. For example, the pruning of $D_1$ (Figure 2(a)) by the rule `for //Department exclude //Name` is depicted in Figure 2(c).

We represent a rule $\rho$ of the form `for` $x_1$ `exclude` $x_2$ as a pair $(x_1, x_2)$. By $x_1x_2$ we denote the XPath expression that is created by the concatenation of the expressions $x_1$ and $x_2$. In a document $D$, $\rho$ specifies as restricted all the pairs $(e_1, e_2)$ of elements of $D$ such that $e_1 \in x_1(D)$ (*i.e.*, $e_1$ is in the answer to $x_1$ over $D$) and $e_2 \in x_1x_2(D)$. For example, the rule `for //Student exclude //Grade` specifies as restricted all the pairs of a student element and a grade of the student. A set of rules specify as restricted all the pairs that are restricted according to at least one of the rules in the set.

Intuitively, given a rule $\rho = (x_1, x_2)$ we want to conceal whether (or not) there is a path between any two restricted elements. We can think of the existing paths in $D$ as defining a subset $P$ of $x_1(D) \times x_1x_2(D)$. We will define as valid only those queries that do not permit a user to distinguish whether $D$ contains the subset $P$ or another possible subset of $x_1(D) \times x_1x_2(D)$. This motivates the following definition.

**Universe of Expansions** Consider a document $D$ and a set of access control rules $R$. Let $\bar{D}$ be the transitive closure of $D$ and let $\textit{prune}_R(\bar{D})$ be the pruning of $\bar{D}$ using the rules of $R$. The *universe of expansions* (universe, for short) of $D$ under the concealment of $R$, is the set of all document expansions $D''$ such that $\textit{prune}_R(\bar{D}) = \textit{prune}_R(D'')$. In other words, the universe contains all the document expansions that are created by

adding to $\mathit{prune}_R(\bar{D})$ some edges that connect restricted pairs of nodes. We denote the universe of $D$ by $\mathcal{U}_R(D)$.

**Definition 2 (Local Validity).** *Given a document $D$ and a set of rules $R$, a query $Q$ is* locally valid *if $Q(D) = Q(D')$ for any document expansion $D'$ in the universe $\mathcal{U}_R(D)$.*

We now explains why we need to consider, in Definition 2, all the document expansions in the universe $\mathcal{U}_R(D)$ instead of just considering the single document expansion $\mathit{prune}_R(\bar{D})$ (the pruning of the transitive closure of $D$) that contains only edges between non-restricted pairs, which we call *pseudo-validity*. A query $Q$ is pseudo-valid if $Q(D) = Q(\mathit{prune}_R(\bar{D}))$. By Definition 2, the condition of pseudo-validity is necessary but not sufficient for $Q$ to be locally valid. The following example demonstrates a situation that secure information may be leaked due to authorizing pseudo-valid queries.

*Example 11.* Consider the courses-grades document $D$ of Figure 1 and the rule $\rho$ in Example 5 to conceal relationships between students and grades. Suppose we authorize pseudo-valid queries such as $Q_i$ : `//Student[SID='12345' and Grade=`$i$`]`, for $i = 0, 1, \ldots, 100$. In all the 100 cases where $i \neq 98$, the query will be authorized and return an empty result. For $i = 98$ (*i.e.*, the grade of the student in the DB course), the query will not be authorized. This reveals the grade of a student in some course.

Such information leakage does not occur when only locally valid queries are authorized. To see why this is true, consider the document expansion $D'$ constructed as follows. Let $D'$ be the result of removing two edges and adding two new edges to the transitive closure $\bar{D}$. The removed edges are the two Student-Grade edges that connect Node 301 to 321 and Node 302 to 322. The two added edges are Student-Grade edges that connect Node 301 to 322 and Node 302 to 321. All these added and removed edges are Student-Grade edges and thus, are removed in a pruning w.r.t. $\rho$. That is, $\mathit{prune}_\rho(\bar{D}) = \mathit{prune}_\rho(D')$. Yet, evaluating $Q_{93}$ w.r.t. $D'$ provides a different answer from the answer to $Q_{93}$ over $D$. Thus, $Q_{93}$ is not valid. $Q_{78}$ is also not valid by a similar construction. All the three queries $Q_{78}$, $Q_{93}$ and $Q_{98}$ are rejected. Thus, a user could only tell the grade of Student '12345' is one of the grades 78, 93, 98; however, this is what she could have learned from the result of the valid query `//Grade`.

The definition of local validity has a number of properties that are important in practice. For example, if two documents are equal (that is, isomorphic) except for their restricted edges, then a locally valid query will not be able to distinguish them.

**Proposition 1.** *Consider a set of rules $R$ and let $D_1$ and $D_2$ be two documents such that $\mathit{prune}_R(\bar{D}_1) = \mathit{prune}_R(\bar{D}_2)$. If a query $Q$ is locally valid w.r.t. $D_1$ and $R$ then $Q$ is also locally valid w.r.t. $D_2$ and $R$. Furthermore, $Q(D_1) = Q(D_2)$.*

### 5.4 Global Validity

For documents conforming to a schema, we define a more restrictive form of validity called *global validity*. First, we formally define the notion of a schema.

**Schema** In our model, a *schema* is represented as a rooted labeled directed graph. Our schema representation is a simplification of common XML schema-definition languages

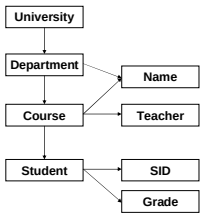


**Fig. 3.** A University schema.

such as DTD [1] and XSchema [2]. A schema can be used to provide a succinct description of a document structure, or as a constraint on the structure of documents in a repository. Formally, a schema $S$, over a finite set of labels $L$, is a rooted labeled directed graph $(\mathit{Names}_S, E_S, \mathit{root}_S, \mathit{label\text{-}of}_S)$ over $L$, where the nodes are uniquely labeled. A document *conforms* to a schema if there exists a *homomorphism* from the graph of the document to the schema. An example of a schema is given in Figure 3(c). The document in Figure 1 conforms to this schema.

**Definition 3 (Global Validity).** *A query $Q$ is globally valid for a set of rules $R$ and a schema $S$, if, given $R$, $Q$ is locally valid for every document $D$ that conforms to $S$.*

*Example 12.* Let $R$ contain the rule given in Example 5. This rule rejects queries that use the relationship between students and grades. Suppose a query $Q$ asking for the grades of the student with id '00000' (*i.e.*, `//Student[SID='00000']//Grade`) is posed on to the document in Figure 1. If there was a student with id '00000' in the document, then the query would not be considered locally valid and would not be authorized. Since there is no student with id '00000', there is no edge to prune and the query is locally valid. Note that the query does not reveal the grade of any existing student. Although $Q$ is locally valid, it is not globally valid if we consider the schema $S$ shown in Figure 3. It is possible to construct a document $D'$ that conforms to $S$ and contains a student with id '00000'. Hence, the query will not be locally valid for $D'$ and $R$. Thus, $Q$ is not globally valid for schema $S$.

In some cases, global validity could be too restrictive; however, it does have some advantages over local validity. Suppose that there is a collection of documents and all the documents conform to the same schema. In this case, if a query is globally valid, then we do not need to check the validity of the query over each document. Furthermore, after a document is updated, if the new document still conforms to the schema, we do not need to revalidate queries.

### 5.5 Complexity of Testing Validity

Due to lack of space, presenting algorithms for efficient validity testing is beyond the scope of this paper. However, it is important to notice that our model has the following advantages. First, local validity is always decidable. Secondly, for large classes of

queries, *e.g.*, queries that do not use the logic operator `not`, local validity can be tested in polynomial time (in the size of the query, the rules and the document), in a test where the query is actually evaluated over only two document expansions. Thirdly, in important cases such as when queries and rules are conjunctive expressions with a bounded number of * symbols, it is possible to test global validity in polynomial time in the size of the query, the rules and the schema. Such a test, which is independent of the data, has high efficiency. For details, see [20].

## 6 A Coherent Set of Rules

Our goal is to allow users to conceal relationships between elements and let them be sure that what they want to conceal is truly concealed. Unfortunately, given an arbitrary set of relationships to be concealed, it cannot be guaranteed that all the relationships are really concealed by making them hidden. Sometimes, it is possible to infer an hidden relationship from the relationships that are not concealed. In this section, we characterize a set of rules whose designated relationships are indeed concealed.

We say a set of rules is *coherent* if it is impossible to infer any hidden relationships from the relationships that are not pruned by the rules. Before providing the formal definition for a coherent set of rules, we give an example of two cases where a relationship can be inferred from a pair of non-concealed relationships.

*Example 13.* Suppose that in the university document it is known that the CAD course (Node 203) is given in the EE department (Node 102) and student 56789 (Node 303) is registered in the CAD course. In this case, the relationship between Node 102 and 303 can be derived from other relationships, thus, there is no point in concealing it alone.

Suppose that it is known Student 12345 (Node 301) studies in the CS department (Node 101) and is registered in the DB course (Node 211). Knowing that the document is a tree allows a user to infer that the DB course is given in the CS department (*i.e.*, Node 201 and Node 301 are related).

We now define when a set of rules is coherent. Consider a document $D$ and a set of rules $R$. The set $R$ has an *incomplete concealment* in a document $D$ if one of the following two cases occurs. *(1)* Lack of transitive: $D$ has three elements $e_1$, $e_2$ and $e_3$ such that $\mathit{prune}_R(\bar{D})$ (the pruning of the transitive closure of $D$ by $R$) has an edge from $e_1$ to $e_2$ and an edge from $e_2$ to $e_3$, but $\mathit{prune}_R(\bar{D})$ does not have an edge from $e_1$ to $e_3$. *(2)* Lack of reverse transitivity: there are three elements $e_1$, $e_2$ and $e_3$ in $D$, such that $\mathit{prune}_R(\bar{D})$ has an edge from $e_1$ to $e_3$ and an edge from $e_2$ to $e_3$; however, $\mathit{prune}_R(\bar{D})$ does not have an edge from $e_1$ to $e_2$.

**Definition 4 (A Coherent Set of Rules).** *Given a document $D$, a set of rules $R$ is* coherent *if an incomplete concealment does not occur in $D$. Given a schema $S$, a set $R$ is coherent if $R$ is coherent for every document that conforms to $S$.*

### 6.1 Coherence for Documents

There is a simple and efficient test for verifying that a set of rules $R$ is coherent for a document $D$. The test starts by computing the pruning of the transitive closure of $D$

according to $R$, and considering the edge set of *prune*$_R(\bar{D})$ as a relation $r$. There is a lack of transitivity if and only if the algebraic expression $\pi_{\$1,\$4}(r \bowtie_{\$2=\$1} r) - r$ is not empty. There is a lack of reverse transitivity if and only if the algebraic expression $\pi_{\$1,\$3}(r \bowtie_{\$2=\$2} r) - r$ is not empty.

Furthermore, we can give intuitive conditions for constructing coherent sets of rules. Our conditions consider how relationships specified by different rules are related. We say that an edge $(e_1, e_2)$ in a transitive closure $\bar{D}$ is *encapsulating* an edge $(e'_1, e'_2)$ if there is a path $\phi$ in $\bar{D}$ that goes through the two edges $(e_1, e_2)$ and $(e'_1, e'_2)$ such that one of the following three cases holds: *(1)* $e_1$ appears on $\phi$ before $e'_1$ and $e'_2$ appears before $e_2$. *(2)* $e_1 = e'_1$ and $e'_2$ appears on $\phi$ before $e_2$. *(3)* $e_1$ appears on $\phi$ before $e'_1$ and $e_2 = e'_2$. The following is a necessary condition for the coherency of a set of rules.

**Proposition 2.** *Given a document $D$, if a set of rules $R$ is coherent, then the following condition holds. For every descendent edge $(e_1, e_2)$ in $\bar{D}$, which is removed in the pruning of $\bar{D}$ by $R$, there is an edge $(e'_1, e'_2)$ in $\bar{D}$ such that $(e'_1, e'_2)$ is encapsulated by $(e_1, e_2)$ and $(e'_1, e'_2)$ is also removed in the pruning of $\bar{D}$.*

Consider two edges $(e_1, e_2)$ and $(e_1, e'_2)$ that are outgoing edges of the same node. We say that these two edges are *parallel* in $\bar{D}$ if either there is a path from $e_2$ to $e'_2$ or vice-versa. That is, these two edges do not lead to two disjointed parts of $\bar{D}$. We will use this definition in the next proposition to provide a sufficient condition for coherency.

**Proposition 3.** *Let $R$ be a set of rules and $D$ be a document. If the following condition holds, then $R$ is coherent w.r.t $D$. For every edge $(e_1, e_2)$ that is removed in the pruning of $\bar{D}$ w.r.t. $R$, all the edges $(e_1, e'_2)$ that are parallel to $(e_1, e_2)$ are also removed in the pruning of $\bar{D}$.*

### 6.2 Coherence for Schemas

Given a schema, we can generalize, using containment of XPath expressions, the condition presented in Proposition 3 for coherency w.r.t. a document. Since the generalized condition is technical and does not contribute to the understanding of the paper, we do not present it here. However, a special case of the generalized condition is that for every document $D$ that conforms to the schema, for each element $e$ in $D$, either all the outgoing edges of $e$ in $\bar{D}$ are removed in the pruning or none of them is removed.

A simple way to ensure this condition is to allow only rules that have one of the following two forms: `for` *path* `exclude //*` or `for` *path* `exclude /`*label*`[`*condition*`]//*`, where *path* can be any XPath expression, *label* can be any label and *condition* can be any XPath condition.

*Example 14.* Consider the schema $S$ in Figure 3. Suppose we want to conceal information of students in courses. We can apply the following two rules. (1) `for //Course exclude /Student` and (2) `for //Course exclude /Student//*`. These rules are coherent w.r.t. the schema $S$.

## 7 Effectiveness of Concealment

In this section, we prove the effectiveness of a coherent set of rules in concealing relationships. The presence of a schema and the fact that documents are trees impose limitations on the relationships that we are able to conceal. These limitations will be discussed in the first part of this section.

### 7.1 The Singleton-Source Disclosure

The *singleton-source disclosure* occurs when a user can infer that two elements $e_1$ and $e_2$ are related, from the following two pieces of information. *(1)* The path from the root to $e_2$ must go through an element of *type* $T$. *(2)* The only element in the document of type $T$ is $e_1$. The problem is illustrated by the following three examples.

*Example 15.* Consider a university document that conforms to the schema in Figure 3 and that contains only a single department element. Consider the rule
`for //Department exclude /Course`,
which presumably conceals the relationships between departments and courses. A user that is familiar with the schema of the document and knows that the document contains only a single department can infer that every course element in the document is below the only department element.

*Example 16.* Consider the document in Figure 1 and the rule
`for //Department[Name='CS'] exclude /Course`.
Suppose that $Q_1$ is a query that looks for all the courses in the document and $Q_2$ is a query that looks for all the courses in departments other than "CS". Both queries are locally valid w.r.t. the document and the rule. By applying set difference to the answers to $Q_1$ and to $Q_2$, it is possible to infer what are the courses in the "CS" department.

*Example 17.* Suppose that there are several departments, in the university document, but only one department element, denoted $d$, has course elements below it. In this case, according to the schema shown in Figure 3, every student in the document must be a descendent of $d$.

We use the notion of $k$-concealment (recall Definition 1) in order to define the singleton-source disclosure.

**Definition 5 (Singleton-source disclosure).** *Consider a document $D$ and a set of rules $R$. A singleton-source disclosure occurs when there is a rule $\rho = (x_1, x_2)$ in $R$ such that the relationship $(x_1(D), x_1x_2(D))$ is not 2-concealed.*

### 7.2 Verifying $k$-Concealment for a Coherent Set of Rules

We will describe now an algorithm that given a document $D$ and a coherent set of rules $R$, tests if a singleton-source disclosure occurs. Essentially, the algorithm computes, for each rule $\rho = (x_1, x_2)$ in $R$, the maximal $k$ for which the relationship $(x_1(D), x_1x_2(D))$ is $k$-concealed. If $k > 1$ for all the rules of $R$, then a singleton-source disclosure does not occur. Otherwise, a singleton-source disclosure does occur.

Before presenting the algorithm that computes $k$, we provide an example that illustrates part of the work of that algorithm.

*Example 18.* Consider the document in Figure 1 and a coherent set of rules $R$. Suppose $R$ contains the rule $\rho$, and $\rho$ hides the relationship between courses and students. There are other rules in $R$ that we do not describe. Suppose $R$ does not hide the relationship between departments and students. We now discuss the computation of $k$ for $\rho$.

There are three students and four courses that we need to consider. For each student $s$, we need to count the number of courses $c$ for which $s$ might be related to $c$. There is a possibility that $s$ is related to $c$ if, and only if, there exists a document $D_{sc}$ for which the following conditions hold. (1) The element $s$ is a descendent of the element $c$, in $D_{sc}$. (2) Every locally valid query $Q$ provides the same answer when posed to $D_{sc}$ and when posed to the original university document.

Intuitively, we can think of a document $D_{sc}$ as a result of moving some subtrees of the original document from one part of the document to another. Thus, for Node 301, we can either leave it below Node 201 or move it to be below Node 202. However, we cannot move Node 301 to be below Node 203. On the conceptual level, this is because Node 203 is a course that belongs to a different department from the department that Node 301 is related to. On the technical level, this is because if we move Node 301 to be below Node 203, we will have two ancestors to Node 301 (Node 101 and Node 102) that are not on the same path. In a tree this should not happen.

In the computation, we check for each student, how many courses it can be related to, as was done for Node 301. In our example, each student has two courses that she can be related to. Thus, in $D$ there is a 2-concealment for $\rho$.

We present now the algorithm—Compute-$k$—that for any coherent set of rules $R$ and a document $D$, computes a maximal value $k$ for which $k$-concealment can be guaranteed for the relationships that are specified by the rules of $R$.

**Compute $k$ ($D$, $R$)**
**Input**: a document $D$ and a coherent set of rules $R$;
**Output**: a maximal $k$ such that there is a $k$-concealment for each rule in $R$

Initially, we set $k = \infty$. We iterate over all the rules of $R$. For each rule $\rho = (x_1, x_2)$ in $R$, we denote by $A$ the set $x_1(D)$; and by $B$ the set $x_1x_2(D)$. We iterate over all the elements of $B$. For each element $b \in B$ we count the number of nodes $a \in A$ such that we can move $b$ to be below $a$ (shortly, we will explain how). Let $k_b$ be this count. Then, if $k_b < k$, we replace $k$ by $k_b$. At the end of all the iterations, $k$ is returned.

We now explain how we count, for a given $b \in B$, the number of nodes $a \in A$ such that we can move $b$ to be below $a$. We do that by iterating over the elements of $A$ and trying to "attach" $b$ below each one of these elements. The test for $b$ and $a$ is as follows. We start by computing the pruning by $R$ of the transitive closure of $D$—$prune_R(\bar{D})$. We will then try to connect $b$ to $a$ using only edges between restricted pairs, *i.e.*, we will add only edges that will be removed in the pruning by $R$. This will produce an expansion $D''$ of $D$ such that $Q(D'') = Q(D)$, for every query $Q$ that is locally valid for $R$ and $D$.

The following observations are important. First, for every $a' \in A$, in $prune_R(\bar{D})$ there does not exist any edge from $a'$ to $b$. This is because of the rule $\rho$. Furthermore,

since $R$ is coherent, in *prune*$_R(\bar{D})$, there is no path from $a'$ to any of the following three: $b$, an ancestor of $b$, or a descendent of $b$. Hence, there is a subtree $T_b$ in *prune*$_R(\bar{D})$ that contains $b$ and is disconnected from any $a' \in A$. What we need to test is the possibility to connect the root of $T_b$ to $a$ or to a descendent of $a$ (using only edges that are removed in the pruning) and to make sure the following two things. First, we want to make sure that we will eventually produce a tree or a graph that can be extended to be a tree. Secondly, we want to make sure that by adding $T_b$ below $a$ we do not create a relationship (*i.e.*, an ancestor-descendent pair) between two nodes that were not related in $D$ and do not form a restricted pair.

To ensure that we are able to extend the new graph to be a tree, we need to verify that the nodes of $T_b$ do not have two ancestors that are not on one path. To that end, we define $\phi$ to be the path, in *prune*$_R(\bar{D})$, from the highest ancestor of $a$ to the root of $T_b$. The highest ancestor of $a$ can be the root, but is not necessarily the root, *e.g.*, if in the pruning, $a$ is disconnected from the root. Now, if there are a node $n$ in $T_b$ and a node $m$ that is neither on $\phi$ nor in $T_b$, then the test fails, *i.e.*, we will not be able to create a tree. In this case, we do not increase $k_b$.

We describe now the test for verifying whether, by adding $T_b$ below $a$, we create a relationship between two nodes that were not related in $D$ and are not a restricted pair. In the test, we simply check that for every pair of nodes connected by a path, after moving $T_b$, either they are connected by an edge in *prune*$_R(\bar{D})$ or they are a restricted pair according to $R$. If this test fails, we do not increase $k_b$. Otherwise, we increase $k_b$ by one.

The algorithm counts for each $b \in B$ all the $a \in A$ that are possible ancestors of $b$. That is, there exists a document $D'$, for which the pruning of the transitive closure—*prune*$_R(\bar{D}')$—is in the universe $\mathcal{U}_R(D)$, and in $D'$, $b$ is a descendent of $a$. When we count for $b \in B$ the elements $a \in A$ that are possible ancestors of $b$, we should not count twice two elements that are on the same path. This requires a simple modification to the algorithm, but we did not include this change to the presentation here in order not to further complicate the description of the algorithm.

**Theorem 1.** *Given a document $D$ and a coherent set of rules $R$, Algorithm Compute-$k$ computes a value $k$ such that the followings hold.*

1. *All the relationships that are defined by rules of $R$ are $k$-concealed.*
2. *There is a rule in $R$ that defines a relationship which is not $k+1$-concealed.*

Theorem 1 shows that when a coherent set of rules is used, it can be tested for a given document $D$ whether 2-concealment, or even $k$-concealment for some $k > 2$, is provided. When $k$-concealment is provided for $D$ and $R$, the following holds. Suppose that $e_1$ and $e_2$ are two elements such that the association between them should be concealed, *i.e.*, there is a rule in $R$ that specifies the relationship $(A, B)$, where $e_1 \in A$ and $e_2 \in B$. Then, a user who sees the answers to locally valid queries will not be able to tell with certainty if the two elements $e_1$ and $e_2$ are connected in $D$. This is because 2-concealment guarantees that there are two documents $D_1$ and $D_2$ such that in $D_1$ the two elements $e_1$ and $e_2$ are connected, while in $D_2$, the two elements $e_1$ and $e_2$ are not connected. Furthermore, $Q(D_1) = Q(D_2)$, for any locally valid query.

An important advantage of the algorithm Compute-$k$ is that it has a polynomial time complexity.

## 8 Conclusion

We presented an authorization-transparent access control mechanism for XML under the Non-Truman model. Our mechanism uses rules, which are formulated using XPath expressions, for specifying element relationships that should be concealed. The proposed access control mechanism has finer granularity than mechanisms that only conceal nodes. We defined the semantics of rules with respect to a document and with respect to a schema. Coherency of a rule set was defined and discussed. A set of rules is coherent if concealed relationships cannot be inferred from non-concealed relationships. We showed how to construct a coherent set of rules. Finally, we presented the notion of $k$-concealment, which is a modification of $k$-anonimyty to our model. When $k$-concealment is provided (for $k > 2$), a user cannot reveal certain information on concealed relationship from the result of merely valid queries. We showed that when access control is performed using a coherent set of rules, $k$-concealment can be tested efficiently.

Future work includes implementing a system that supports our access control rules, and integrating this system with existing XPath query processors. Implementation should help develope optimization techniques to our validity tests. An important challenge is to adapt our mechanism to XQuery and XSLT. Another important issue that we leave for future work is dealing with queries that include aggregate functions.

## References


1. XML. The World Wide Web Consortium (W3C). Available at `http://www.w3c.org/XML`.
2. XML Schema. The World Wide Web Consortium (W3C). Available at `http://www.w3c.org/XML/Schema`.
3. R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st ICDE*, pages 217–228, 2005.
4. E. Bertino, S. Castano, and E. Ferrari. On specifying security policies for web documents with an xml-based language. In *Proceedings of the 6th SACMAT*, pages 57–65, 2001.
5. E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and enforcing access control policies for xml document sources. 3:139–151, 2000.
6. E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. *ACM TISSEC*, 5(3):290–331, 2002.
7. R. Bhatti, E. Bertino, A. Ghafoor, and J. Joshi. Xml-based specification for web services document security. In *IEEE Computer*, volume 4 of *37*, pages 41–49, 2004.
8. D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. XQuery version 1.0: An XML query language, June 2001. Available at `http://www.w3.org/TR/xquery`.
9. SungRan Cho, S. Amer-Yahia, L. V.S. Lakshmanan, and D. Srivastava. Optimizing the secure evaluation of twig queries. In *Proceedings of the 28th VLDB*, pages 490–501, 2002.
10. J. Clark. XSL Transformations (XSLT) version 1.0. Available at `http://www.w3.org/TR/xslt`, 1999.
11. J. Clark and S. DeRose. XML Path Language (XPath) version 1.0. Available at `http://www.w3.org/TR/xpath`, 1999.
12. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *ACM TISSEC*, 5(3):169–202, 2002.

13. E. Damiani, S. Samarati, S. di Vimercati, and S. Paraboschi. Controlling access to xml documents. *IEEE Internet Computing*, 5(6):18–28, 2001.
14. W. Fan, C. Chan, and M. Garofalakis. Secure XML querying with security views. In *Proceedings of the 23rd ACM SIGMOD*, pages 587–598, 2004.
15. B. Finance, S. Medjdoub, and P. Pucheral. The case for access control on xml relationships. Technical report, INRIA, 2005. Available from http://www-smis.inria.fr/dataFiles/FMP05a.pdf.
16. I. Fundulaki and M. Marx. Specifying access control policies for XML documents with XPath. In *Proceedings of the 9th ACM SACMAT*, pages 61–69, 2004.
17. A. Gabillon and E. Bruno. Regulating access to xml documents. In *Proceedings of the 15th IFIP WG11.3*, pages 299–314, 2001.
18. S. Godik and T. Moses. eXtesible Access Control Markup Language (XACML) Version 1.0. Available at `http://www.oasis-open.org/committees/xacml`, 2003. OASIS Standard.
19. S. Hada and M. Kudo. XML Access Control Language: provisional authorization for XML documents. Available at `http://www.trl.ibm.com/projects/xml/xacl`.
20. Y. Kanza, A. O. Mendelzon, R. J. Miller, and Z. Zhang. Authorization-Based Access Control for XML. Technical Report CSRG-527, University of Toronto, Department of Computer Science, 2005. Available from ftp://ftp.cs.toronto.edu/csrg-technical-reports.
21. G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *Proceedings of the 29th VLDB*, pages 898–909, 2003.
22. G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *Proceedings of the 23rd ACM SIGMOD*, pages 575–586, 2004.
23. S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM TISSEC*, 3(2):85–106, 2000.
24. S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *Proceedings of the 23rd ACM SIGMOD*, pages 551–562, 2004.
25. L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.
26. J. Wang and S. L. Osborn. A role-based approach to access control for XML databases. In *Proceedings of the 9th ACM SACMAT*, pages 70–77, 2004.
27. Wanhong Xu and Z. M. Özsoyoglu. Rewriting xpath queries using materialized views. In *Proceedings of the 31st VLDB*, pages 121–132, 2005.
28. C. Yao, X. S. Wang, and S. Jajodia. Checking for k-anonymity violation by views. In *Proceedings of the 31st VLDB*, pages 910–921, 2005.
29. T. Yu, D. Srivastava, L. V.S. Lakshmanan, and H. V. Jagadish. Compressed accessibility map: efficient access control for xml. In *Proceedings of the 28th VLDB*, pages 363–402, 2002.
30. Z. Zhang and A. O. Mendelzon. Authorization views and conditional query containment. In *Proceedings of the 10th ICDT*, pages 259–273, 2005.