# Efficient Database-Driven Evaluation of Security Clearance for Federated Access Control of Dynamic XML Documents

**Erwin Leonardi**[†] **Sourav S Bhowmick**[†] **Mizuho Iwaihara**[§]

[†]School of Computer Engineering
Nanyang Technological University, Singapore

[§]Graduate School of Information, Production and System
Waseda University, Japan
lerwin|assourav@ntu.edu.sg, iwaihara@aoni.waseda.jp

**Abstract**

Achieving data security over cooperating web services is becoming a reality, but existing access control architectures for XML databases do not consider this federated service computing. In this paper, we consider a federated access control model, in which *Data Provider* and *Policy Enforcers* are separated into different organizations; the *Data Provider* is responsible for evaluating criticality of requested XML documents based on co-occurrence of security objects, and issuing security clearances. The *Policy Enforcers* enforce access control rules reflecting their organization-specific policies. A user's query is sent to the *Data Provider* and he/she needs to obtain a permission from the *Policy Enforcer* in his/her organization to read the results of his/her query. The *Data Provider* evaluates the query and also evaluate *criticality* of the query, where evaluation of sensitiveness is carried out by using *clearance rules*. In this setting, we present a novel approach, called the DIFF approach, to evaluate security clearance by the *Data Provider*. Our technique is build on top of relational framework and utilizes pre-evaluated clearances by taking the differences (or deltas) between query results. Our experimental results show that the DIFF approach outperforms the SCAN approach, that evaluates the whole content of each requested results.

# 1 Introduction

As companies transact business over the Internet, letting authorized customers to access and even modify data stored in XML documents over the Web offers many advantages in terms of cost, accuracy, and timeliness. However, this raises an important question on security as due to the sensitive nature of business data, access should be given to the requester in a selective manner. The requester should not be aware of the information within a document hidden from him/her. In other words, a requester should be allowed to view only those information that (s)he has the right to access and nothing else. Hence, access control for XML documents is important to ensure access control policies on who can access what part of the documents. Proposed access control models for XML documents [6, 8, 9, 16] put emphasis on fine-grained access control of the document structure of XML and mainly focus on efficient evaluation of access control rules. For example, static analysis [23] and other methods [10] [20] utilize containment of XPath expressions between queries, access control rules, and DTDs. More recently, XML stream firewall [5] takes a different approach for rule evaluation. All these efforts assume a centralized database server, where a single point is responsible for enforcing policies.

Increasingly, data and services are becoming decentralized in nature. For example, the architecture of web services is becoming more decentralized; a number of servers stretching over different locations/organizations are orchestrating together to provide a unified service, sometimes referred to as *cloud computing* [15]. In this setting, access control for protecting sensitive data should also be cross-organizational, where a user, an access requester, and the *Data Provider* holding sensitive data, belong to different organizations. For example, in the medical domain, patient records of a hospital may be accessed by a user from another hospital or pharmacy. In this federated access control scenario, a *Policy Enforcer* at the user's organization shall be created for handling user authentication and administrating users' privileges. The two organizations should agree on *global security policies* for handling sensitive data, and this agreement should be legally powerful enough to sustain trustability between the two parties. In this case, we need to consider efficiency of access control through distribution of functionalities to co-operating servers.

Figure 1 shows a conceptual depiction of our federated access control model. The *Data Providers* and *Policy Enforcers* are separated into different organizations; each *Data Provider* is responsible for evaluating criticality of requested XML documents based on co-occurrence of security objects, and issuing security clearances. The *Policy Enforcers* enforce access control rules which reflect their own organization-specific policies. We assume that these organizations have agreed on *global* security policies for information exchange.

Let us illustrate the architecture with an example depicted in Figure 2(a) containing a part of a travel plan produced by a travel agency. We assume that the travel agency respond to request from clients and users using XML documents. These documents may contain sensitive information. Suppose that an XML document contain-

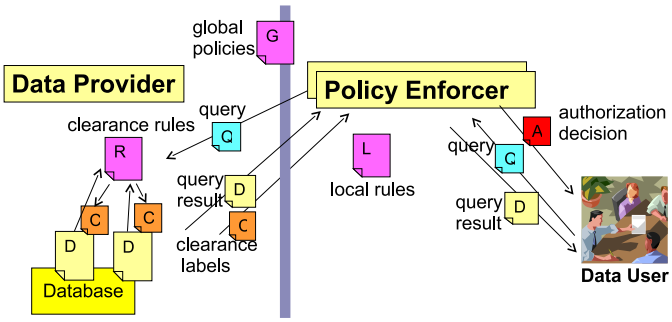
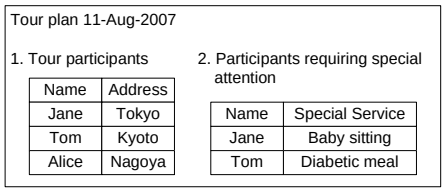



Figure 1: Overview of federated access control.

ing relevant results is requested by a user in an airline company providing flights for the tour. The user needs to obtain a permission from the *Policy Enforcer* in his/her organization to read the document. The user query is sent to the *Data Provider* (in this case, the travel agency). The *Data Provider* evaluates the query and also evaluate criticality of the query, where evaluation of sensitiveness is carried out by using *clearance rules R*. The rules consist of queries pointing to *security objects* in the documents, together with *clearance labels* that define necessary security clearance the user should have. Figure 2(b) illustrates a sample of clearance rules represented as a table. If we apply the clearance rules to the document shown in Figure 2(a), we obtain the clearance labels $L1, L2$, and $L3$. For instance, the objects `Alice` and `Nagoya` appear in the document and matches the rule `(Alice, Nagoya,` $L1$`)`. Likewise, `Jane` and `Tokyo`appear in the document and matches the rule `(Jane, Tokyo,` $L2$`)`. The co-occurrence of `Tom`, `Kyoto`, `Diabetic` meal raises the label $L3$. A partial order between labels can be introduced to indicate the degree of sensitiveness of the labels (e.g., $L1 < L2 < L3$). Note that assigning the label $L3$ to co-occurrence of `Tom`, `Kyoto`, `Diabetic` meal is appropriate because privacy breach of a person's health condition is often more serious than that of a person's address.

Finally, the *Policy Enforcer* receives the clearance labels $C$, and decides whether the user is eligible for the clearance by mapping the labels $C$ to its local roles, and checking whether the user is assigned to one of these roles. Observe that by issuing clearance, the *Data Provider* can export the task of access authorization to the *Policy Enforcer*, thus realizing federated access control. The above architecture has the following advantages over the centralized access control model.

- **Separation of global and local policies:** Access control rules can be categorized into rules at the *Data Provider* side and at the *Policy Enforcer* side.

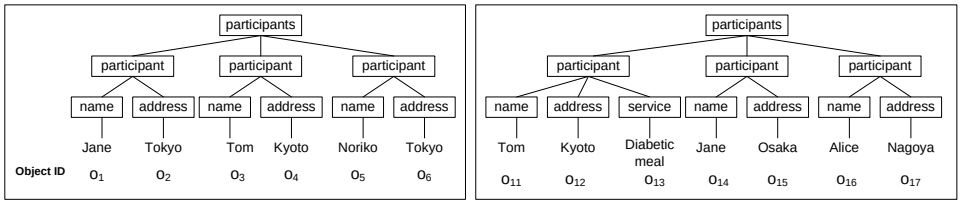



(a) A Published Document

| objects | label |
|---|---|
| {Alice, Nagoya} | *L1* |
| {Jane, Tokyo} | *L2* |
| {Tom, Kyoto, Diabetic meal} | *L3* |

(b) Clearance Rules

Figure 2: Example.




(a) Document $q_1$ (b) Document $q_2$

Figure 3: Example of results evaluated by the *Data Provider*.

This enables the *Policy Enforcer* to translate the global policies into their local access control rules. Since the *Policy Enforcer* has its own organizational structure, its implementation of the global policies into local rules can reflect their local organizational structure, so that the local policies vary among these user organizations.

- **Security for users:** Since access authorization is carried out at the *Policy Enforcer*, each user does not have to submit their authentication information such as account/password, as well as other personal information, to the *Data Provider*.
- **Load balancing:** Evaluating access control rules over data requests can be distributed between the two parties. Decentralization also allows caching of the outcomes of access control policy evaluation. For example, if different users are requesting the same data, the *Data Provider* needs to evaluate its rules for the requests only once. Further, for dynamic documents evolving over time, cached outcomes can be applied for unchanged parts of the data.

***In this paper, we focus on an issue of Data Provider module that is orthogonal to the* XML *access control problem in the collaborative or federated environment.*** In particular, we present a novel *diff-based* approach for security clearance by the *Data Provider* in a ***dynamic*** environment.

The rest of the paper is organized as follows. In Section 2, we present the motivation and overview of our work. In Section 3, we provide preliminary concepts relevant to the security clearance problem. We present the naïve approach of clearance evaluation, called the SCAN approach, in Section 4. Next, in Section 5, we discuss in detail the algorithm for the DIFF approach. We discuss the results of our

experimental evaluation in Section 6. Related work will be covered in Section 7. Finally, the last section concludes this paper.

## 2 Motivation and Overview

There has been a number of efforts to realize federated access control [2, 3, 7, 18, 21]. The Enterprise Role-Based Access Control Model [18] handles two-layered (enterprise-level and department-level) access control policies. Liberty Alliance [2] is a leading industrial standard for federated identity management, where *service providers* and *identity providers* form *circle of trust*, a mutual trust relationship, and *identity providers* take responsibility on user authentication. OpenID [3] is another example of federated identity management. XACML is a standard language for expressing access control rules/policies in XML [4]. Its has decentralization features such as separated *Policy Decision Point* and *Policy Enforcement Point*. Lin et al. [21] proposed a framework to decompose global policies into local policies over a set of collaborating parties without compromising the autonomy or confidentiality requirement of these parties. However, to the best of our knowledge, none of these efforts have undertaken a systematic study of the security clearance technique supported by the *Data Provider* in a dynamic environment where the underlying XML documents may evolve with time. In this paper, we focus on *a database-driven strategy for efficient evaluation of security clearance by the Data Provider in* ***dynamic*** *and* ***federated access control environment***. Our proposed technique ***compliments existing research on federated access control strategies for XML documents***.

Since the access control policies are realized through integration of the clearance rules at the *Data Provider* and the local rules at *Policy Enforcers*, at first glance, it may seem that we could take the strategy of pre-evaluating clearance rules at the *Data Provider* as much as possible and cache obtained clearance labels. The advantage of this strategy is that clearance evaluation for repeated queries can be avoided. However, this approach is not feasible due to the following reasons. Firstly, the underlying XML documents are dynamic in nature and hence document updates will invalidate cached clearances. Secondly, since effective clearances vary according to retrieved security objects by the query, utilizing pre-evaluated clearance labels becomes a challenging security problem. Let us illustrate this issue with an example.

Consider the documents $q_1$ and $q_2$ sent by the *Data Provider* to a *Policy Enforcer* in response to a user's queries at times $t_1$ and $t_2$, respectively, where $t_1 < t_2$. We assume that the *Data Provider* represents the query results in XML format and the set of clearance rules in Figure 2(b) must be satisfied by the documents. It is quite possible for $q_1$ and $q_2$ to share some data objects due to the following reasons: (a) $q_1$ and $q_2$ are results of the *same* query that is issued at times $t_1$ and $t_2$. The results may not be identical as the underlying data have evolved during this time period; (b) $q_1$ and $q_2$ are results of two different queries. However, some frag-

ments of the underlying data may satisfy both the queries. Consequently, only the second rule in Figure 2(b) is valid for $q_1$. However, in $q_2$ this rule does not hold anymore. On the contrary, now the first and third rules are valid for $q_2$. In other words, updates to the underlying data invalidates caching of the clearance rules of $q_1$.

In this paper, we take a novel approach for evaluating security clearance by exploiting the overlapping nature of query results. Specifically, we investigate taking differences (deltas) of XML representations of the query results, so that valid clearance labels can be detected and reused. We compute the clearance labels of the first result ($q_1$) by scanning the entire result. Subsequently, labels of subsequent results are computed efficiently by analyzing the differences between the results. We refer to this strategy as the DIFF approach. Since we store the clearance rules and XML results in a RDBMS, the DIFF approach detects differences in the query results and clearance rules using a series of SQL statements. Our experimental results show that the proposed diff-based approach has superior performance compared to the approach that scans the entire resultset for every request (referred to as the SCAN approach).

In summary, the main contributions of this paper are as follows:

- We present a novel technique for security clearance evaluation by the *Data Provider* in the dynamic federated access control environment. Importantly, our proposed algorithm exploits the overlapping nature of query results and is capable of working with any off-the-shelf RDBMS without any internal modifications.
- Through an extensive experimental study, we show that our proposed DIFF approach is efficient and scalable and can outperform technique that requires to scan the entire results for every request from the *Policy Enforcers*.

# 3 Preliminaries

Clearance rules and labels have been proposed decades back in the context of multi-level security (MLS). We adopt them in the context of federated access control framework for XML documents. We begin by defining the *clearance rule* as follows.

**Definition 1** *A clearance rule r is a 2-tuple* $[O, L]$*, where O is a set of objects existing in* XML *documents and L is a clearance label.*

For example, in Figure 2(b), we have three clearance rules: `[Alice, Nagoya,` $L1$`]`, `[Jane, Tokyo,` $L2$`]`, and `[Tom, Kyoto, Diabetic meal,` $L3$`]`. The objects in $O$ are the sensitive information whose co-occurrences raise security cautions. Note that these objects can be results of a set of XPath queries. We use $O$ instead of a set of queries in the definition of clearance rule as the process of generating these objects *is orthogonal to our proposed security clearance technique*

discussed later. Indeed, we can use any existing sophisticated XML access policy-based technique for generating security objects. The level of security caution is defined by the *clearance label*. Formally,

**Definition 2** *Let $B = \{b_1, b_2, \ldots, b_n\}$ be a bag of objects in a query result q. Let $r = [O, L]$ be a clearance rule. A clearance rule r raises a security caution defined by clearance label L iff $O \subseteq B$.*

For instance, the third clearance rule in Figure 2(b) raises a security caution $L_3$ for the document depicted in Figure 2(a).

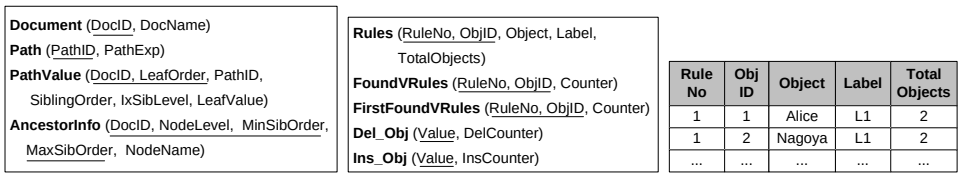
### 3.1 Context-free Occurrences of Security Objects

An object may appear as a string in any part of an XML tree (query results). In this paper, we limit the scope of the objects to be text nodes of an XML tree. This is because very often the sensitive objects are the data values in an XML document rather than elements that define the structure of the document. Note that string matching for such object detection can be exact, flexible, or domain-dependent. Flexible matching can be realized by a similarity constraint $sim(o_1, o_2) \geq \beta$ such that if objects $o_1$ and $o_2$ satisfy the constraint, then $o_1$ and $o_2$ are matching objects. For the similarity function $sim(o_1, o_2)$, we can use the inverse of the edit distance of the strings of $o_1$ and $o_2$. The variable $\beta$ is a user-defined threshold. Similarity-based matching is useful for detecting slightly-modified sensitive objects. As for domain-dependent matching, we can utilize existing techniques on identifying domain values such as persons' names [24], addresses, and phone numbers. In this paper $n$ text objects appearing in any part of a document is regarded as a co-occurrence. Note that there are situations where $n$ objects in a document match a clearance rule but these objects may not be *semantically* related. We leave a more precise analysis of semantically related objects for future work, and take a conservative approach of the above context-free model of co-occurrence.

### 3.2 Clearance Label

A *clearance label* is a symbol representing security clearance required to read/receive the document. In the local rules, the *Policy Enforcer* should assign roles that have privilege. In the widely-accepted role-based access control (RBAC) model [11,16], the mapping of privileges to users is decomposed into the mapping from privileges to roles and the mapping from roles to users.

Note that clearance labels can have a partial order of priority, such as: $L1$ for *non-sensitive*, $L2$ for *sensitive*, $L3$ for *confidential*, and $L4$ for *strictly confidential*. A partial order such as $L1 < L2 < L3 < L4$, where '$A < B$' means that $B$ is superior or more cautious than $A$. A query may raise a set of clearance labels, but if a priority order between labels is defined, a label that is dominated by another superior label can be ignored.




(a) SUCXENT Schema (b) Additional Relations

| Rule No | Obj ID | Object | Label | Total Objects |
|---|---|---|---|---|
| 1 | 1 | Alice | L1 | 2 |
| 1 | 2 | Nagoya | L1 | 2 |
| ... | ... | ... | ... | ... |

(c) The Rules Table (Partial view)

Figure 4: Relational schemas.

# 4 The SCAN Approach

In this section, we first present the naïve approach, called the SCAN approach, for evaluating clearance for a batch of results in XML format. We begin by first presenting the relational schema that we use for storing results and clearance rules in the database for both SCAN and DIFF approaches.

## 4.1 Relational Schema

As the results requested by a *Policy Enforcer* are represented in an XML format, we can use any existing techniques for XML storage built on top of a RDBMS [13] to store these results. We use the SUCXENT schema [25] depicted in Figure 4(a) for storing the request results in a RDBMS. SUCXENT is a tree-unaware approach for storing and querying XML documents in relational databases. Particularly, in this paper, only the `LeafValue` attribute of the `PathValue` table is used for security clearance evaluation. The `PathValue` table stores the textual content of the leaf nodes of an XML tree in the `LeafValue` column. Hence, we do not elaborate on the remaining attributes and tables in Figure 4(a). The reader may refer to [25] for details related to the SUCXENT schema.

The clearance rules are stored in the `Rules` table (Figure 4(b)). The `RuleNo` attribute is used as an unique identifier of a rule. The `TotalObjects` attribute maintains the total number of sensitive objects in a rule $r$ whose co-occurrences raise security cautions. The level of security caution is stored in the `Label` attribute. The `ObjID` and `Object` attributes store the identifier and value of the text objects in the query results, respectively. For example, Figure 4(c) depicts how the first rule `(Alice, Nagoya,` $L1$`)` in Figure 2(b) is stored in the `Rules` table. The remaining tables in Figure 4(b) are used for the DIFF approach and we shall elaborate on them in the next section.

## 4.2 The SCAN Algorithm

In this approach, we scan each results to determine the sensitive objects and their clearance labels. Given a sequence of query results $Q = \langle q_1, q_2, \ldots, q_n \rangle$ and clearance rules $R = \{r_1, r_2, \ldots, r_k\}$, the SCAN algorithm invokes an SQL query depicted in Figure 5 for each $q_i \in Q$. The objective of this algorithm is to find $C \subseteq R$ where $\forall c_w \in C$ raises a security caution. The *did* in Line 6 is replaced by the

```
01  SELECT DISTINCT C.RULENO, C.CLABEL
02  FROM
03    (SELECT C.RULENO,
04            COUNT(DISTINCT C.OBJID) AS TOTAL
05     FROM RULES C, PATHVALUE L
06     WHERE L.DOC_ID = did
07       AND CHARINDEX(' ' + C.OBJECT + ' ',
                 ' ' + L.LEAFVALUE +' ') >0
08     GROUP BY C.RULENO) X,
09    RULES C
10  WHERE C.RULENO = X.RULENO
11    AND X.TOTAL = C.TOTALOBJECT
```

Figure 5: The SQL query in the SCAN approach.

id of requested results. Line 7 is used to find sensitive objects that appear in requested results. Note that we extend this SQL query using more sophisticated string matching method. Lines 3-8 are used to determine how many sensitive objects of a particular rule appeared in the requested results. Line 11 ensures that all sensitive objects of a particular rule are in the result requests. Thus, the rule raises security cautions (Definition 2). The SQL query in Figure 4(c) will return pairs of rules that raise security cautions and their clearance labels.

# 5 The DIFF Approach

The SCAN approach evaluates clearance rules every time a new query arrive at the *Data Provider*. In this section, we present a more efficient strategy, called the DIFF approach, for evaluating clearance rules.

## 5.1 Efficient Clearance Evaluation Through Document Difference

Recall from our preceding discussion the following opportunities for more efficient processing of security clearance.

- Since the organizational policies are implemented at the local rules, the clearance rules can be pre-evaluated into clearance labels prior to access requests, and produced clearance labels can be cached. Upon access requests, the *Data Provider* just needs to return these pre-evaluated clearance labels if the labels are available and correct.

- Cache hits occur in a high rate for hotspot data, which are accessed by many users from different organizations. The cached clearance labels can be reused because the labels are the same if the queried objects are the same.

However, the above approach raises the following issues: (a) When a document in the database is updated, its corresponding clearance labels become invalid. (b) Applicable clearance labels vary according to the set of retrieved security objects. The first issue of cache invalidation can be addressed by managing timestamps of

query results and clearance labels as a naive approach. But since an update can be limited within a small region of a document, discarding all the clearance labels of the results is wasteful. Regarding the second issue, consider a set of query results $Q = \{q_1, q_2, \ldots, q_n\}$ in XML format. We refer to these results as *versions* in the sequel. Assume that the clearance labels for $q_1$ are cached, but no cache entry exists for the remaining results $q_i$ where $i > 1$. How can the *Data Provider* evaluate clearance labels for the remaining $(n-1)$ versions efficiently? In the DIFF approach, we take advantage of the significant overlaps between $q_1$ and remaining results by reusing cached clearance labels whenever possible, and re-evaluate the clearance rules that are only affected by the changes (deltas) to the results. Note that often the size of the deltas are typically smaller than the size of $q_i$.

## 5.2 Effects of the Changes

Suppose we have two query results, namely $q_1$ and $q_2$, and a set of clearance rules $C = \{c_1, c_2, \ldots, c_n\}$. After evaluating $q_1$, let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of clearance rules that match with $q_1$ where $R \subseteq C$. Let $O_{q1}$ and $O_{q2}$ be the bags of objects in $q_1$ and $q_2$, respectively.

Let us now discuss the effects of the changes to the query results on the clearance rules. In this paper, we focus on two types of change operations to the query results: *deletion* and *insertion* of text objects. Note that the *update* of a text object can be represented as a sequence of delete and insert operations. An object $o_{del}$ is a **deleted** object iff $o_{del} \in O_{q1}$ and $o_{del} \notin O_{q2}$. Property 1 describes the effects of a deletion of an object on the clearance rules.

**Property 1** A deletion of an object $o_{del}$ will cause the removal of clearance rule $r \in R$ iff co-occurrence $o_{del}$ with $o_k \in O_{q1}$ forms the clearance rule $r \in R$, and there does not exist $o_{k'} \in O_{q1}$ such that $value(o_{k'}) = value(o_{del})$ where $value(o)$ is the text value of object $o$.

Similarly, an object $o_{ins}$ is an **inserted** object iff $o_{ins} \notin O_{q1}$ and $o_{ins} \in O_{q2}$. Property 2 explains the effects of an insertion of an object to the clearance rules.

**Property 2** An insertion of an object $o_{ins_1}$ will cause an addition of clearance rule $r$ into $R$ if o-occurrence of $o_{ins_1}$ with $o_j \in O_{q1}$ forms a clearance rule $r \in C$, or co-occurrence of $o_{ins_1}$ with another inserted object $o_{ins_2}$ forms a clearance rule $r \in C$.

For example, consider the two query results $q_1$ and $q_2$ as depicted in Figure 3 and the set of clearance rules $C$ in Figure 2(b). In $q_1$, only the second rule in Figure 2(b) is added into $R$. The objects with ids 2, 5, and 6 (denoted as $o_2$, $o_5$ and $o_6$, respectively) are deleted objects. The deletion of $o_2$ results in the removal of the second rule from $R$. Meanwhile, deletions of $o_5$ and $o_6$ do not cause any rule deletion. Furthermore, objects $o_{13}$, $o_{16}$, and $o_{17}$ are inserted in $q_2$. Insertion of $o_{13}$ will trigger the addition of a new rule `[{Tom, Kyoto, Diabetic meal},`$L3$`]` into $R$. Similarly, insertions of $o_{16}$ and $o_{17}$ cause an addition of the rule `[{Alice,Nagoya},`$L1$`]`

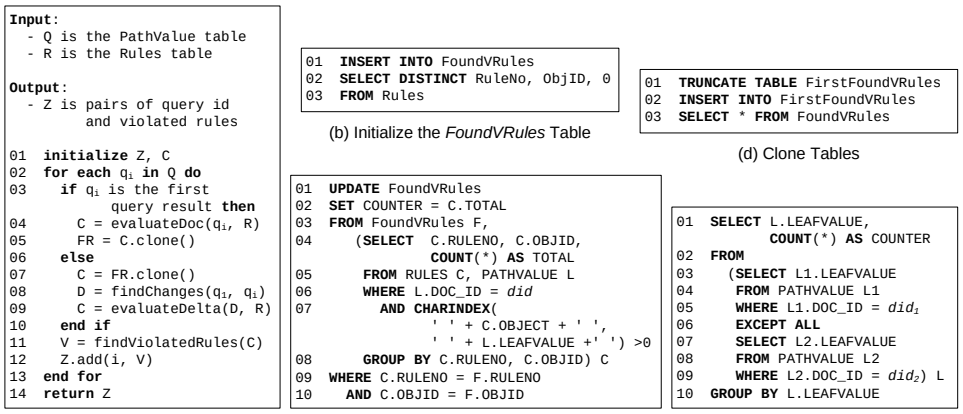



Figure 6: The DIFF algorithm and SQL queries.

into $R$. Based on the above properties, the DIFF approach will evaluate the query results. We now describe the algorithm in detail.

## 5.3 The DIFF Algorithm

The DIFF algorithm is depicted in Figure 6(a). The input to the algorithm are two relational tables, namely the `PathValue` table (denoted as $Q$ in Figure 6(a)) and the `Rules` table (denoted as $R$ in Figure 6(a)). Note that the requested results are stored in the `PathValue` table. The first step is to initialize the `FoundVRules` table (denoted as $C$ in Figure 6(a)) by invoking an SQL query depicted in Figure 6(b) and a list $Z$. Note that the `FoundVRules` and `FirstFoundVRules` tables (Figure 4(b)) are used to keep track of the number of sensitive objects that appeared in the requested query results. The number of occurrences of $k$-th sensitive object of a rule $r$ is stored in the `Counter` attribute. For each query result, the algorithm will do the followings (Lines 02–13). If the current query result is the first one ($q_1$), then it evaluates the occurrences of sensitive objects in $q_1$ (Lines 03–05). The evaluation is done by executing the SQL query depicted in Figure 6(c). The objective of this query is to update the value of `Counter` attribute of the `FoundVRules` tables to the number of occurrences of a sensitive object in a particular rule (Lines 04-08, Figure 6(c)). Next, the algorithm clones the `FoundVRules` table into the `FirstFoundVRules` table. The `FirstFoundVRules` table stores the results generated by evaluation of $q_1$.

If the current requested query results is *not* the first one (denoted as $q_i$ where $i > 1$), then the algorithm will do the followings (Lines 06–10). First, it clones the `FirstFoundVRules` table into the `FoundVRules` table (Line 07) using the SQL query depicted in Figure 6(d). This step is important as we want to evaluate clearance for $q_i$ using the clearance of $q_1$. Next, the algorithm determines the differences between $q_1$ and $q_i$ by executing two SQL queries. The first SQL query is used to find the deleted objects (Figure 6(e)). Note that $did_1$ and $did_2$ will be

```
01  UPDATE FoundVRules SET COUNTER =  C.TOTAL
02  FROM  FoundVRules F,
03    (SELECT C.RULENO, C.OBJID,
              F.COUNTER - COUNT(*) AS TOTAL
04     FROM DEL_OBJ D, RULES C, FirstFoundVRules F
05     WHERE CHARINDEX(' ' + C.OBJECT + ' ',
                       ' ' + D.VALUE  +' ') >0
06       AND C.RULENO = F.RULENO
07       AND C.OBJID = F.OBJID
08     GROUP BY C.RULENO, C.OBJID, F.COUNTER) C
09  WHERE C.RULENO = F.RULENO  AND C.OBJID = F.OBJID
```

(a) Analyze the Changes

```
01  SELECT DISTINCT C.RULENO, C.CLABEL
02  FROM RULES C,
03    (SELECT F.RULENO,
              COUNT(F.OBJID) AS VOBJ
04     FROM FoundVRules F
05     WHERE F.COUNTER >0
06     GROUP BY F.RULENO) F
07  WHERE F.RULENO = C.RULENO
08    AND F.VOBJ = C.TOTALOBJECT
```

(b) Find Violated Rules

Figure 7: SQL queries used in DIFF approach.

| Dataset | Number of Leaf Nodes | Filesize (KB) |
|---|---|---|
| 1 | 258 | 13 |
| 2 | 427 | 21 |
| 3 | 703 | 34 |
| 4 | 1,437 | 70 |
| 5 | 2,151 | 104 |
| 6 | 3,734 | 180 |

(a) Data Set

| N | R | Total Objects |
|---|---|---|
| 2 | 50 | 100 |
| 4 | 50 | 200 |
| 6 | 50 | 300 |
| 8 | 50 | 400 |
| 10 | 50 | 500 |

| N | R | Total Objects |
|---|---|---|
| 2 | 500 | 1,000 |
| 4 | 500 | 2,000 |
| 6 | 500 | 3,000 |
| 8 | 500 | 4,000 |
| 10 | 500 | 5,000 |

| N | R | Total Objects |
|---|---|---|
| 2 | 5,000 | 10,000 |
| 4 | 5,000 | 20,000 |
| 6 | 5,000 | 30,000 |
| 8 | 5,000 | 40,000 |
| 10 | 5,000 | 50,000 |

(b) Clearance Rules Characteristics

Figure 8: Dataset and clearance rules characteristics.

replaced by the ids of $q_1$ and $q_i$, respectively. The result of this SQL query is stored in the `Del_Obj` table (Figure 4(b)). The second SQL query is used to detect the inserted objects. We use the same SQL query as shown in Figure 6(e); however, $did1$ and $did2$ will be replaced by the ids of $q_i$ and $q_1$, respectively. The results of this SQL query is kept in the `Ins_Obj` table (Figure 4(b)).

Having found the differences between $q_1$ and $q_i$, the algorithm analyzes the deleted and inserted objects based on the Property 1 and Property 2, respectively, in order to determine the clearance of $q_i$. The SQL query depicted in Figure 7(a) is executed to analyze the set of deleted objects. Line 3 is used to decrease the number of appearances of sensitive objects if the sensitive objects are deleted. Similarly, this query is slightly modified to analyze the inserted objects. The modifications are as following. The "-" in Line 3 is replaced by "+". In Line 4, we replace "DEL_OBJ" with "INS_OBJ".

The last step in evaluating each requested result $q_i$ is to find the rules that raise security cautions by querying the `FoundVRules` table (denoted by $V$). Figure 7(b) presents the SQL query for determining such rules and is based on Definition 2. Then, we add a pair of request ids $i$ and $V$ into $Z$. Finally, the algorithm returns $Z$ which may be analyzed further in order to determine which requested results are safe for publication.

| Data set | N = 2 | | | | N = 6 | | | | N = 10 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | Avg | 1 | 2 | 3 | Avg | 1 | 2 | 3 | Avg |
| 1 | 1.27 | 1.21 | 1.14 | **1.21** | 4.93 | 4.76 | 4.20 | **4.63** | 7.63 | 7.74 | 5.97 | **7.11** |
| 2 | 1.99 | 1.88 | 1.93 | **1.93** | 7.50 | 7.28 | 7.45 | **7.41** | 12.44 | 12.07 | 12.42 | **12.31** |
| 3 | 3.20 | 3.17 | 3.15 | **3.17** | 12.20 | 12.22 | 12.16 | **12.19** | 20.32 | 20.25 | 20.21 | **20.26** |
| 4 | 8.32 | 8.47 | 8.49 | **8.43** | 24.87 | 24.72 | 24.78 | **24.79** | 42.47 | 41.16 | 41.26 | **41.63** |
| 5 | 12.44 | 12.35 | 12.38 | **12.39** | 37.19 | 37.23 | 37.07 | **37.16** | 61.78 | 61.68 | 61.88 | **61.78** |
| 6 | 21.40 | 21.44 | 21.51 | **21.45** | 64.31 | 64.46 | 64.50 | **64.43** | 109.46 | 107.25 | 107.00 | **107.90** |

(a) R=500

| Data set | N = 2 | | | | N = 6 | | | | N = 10 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | Avg | 1 | 2 | 3 | Avg | 1 | 2 | 3 | Avg |
| 1 | 15.12 | 15.39 | 11.67 | **14.06** | 45.04 | 45.81 | 45.08 | **45.31** | 74.91 | 76.45 | 75.13 | **75.50** |
| 2 | 24.76 | 24.07 | 24.71 | **24.51** | 73.92 | 71.92 | 73.85 | **73.23** | 122.67 | 119.95 | 122.81 | **121.81** |
| 3 | 42.53 | 40.35 | 40.27 | **41.05** | 120.97 | 121.46 | 121.07 | **121.16** | 201.41 | 201.60 | 200.55 | **201.19** |
| 4 | 82.73 | 81.98 | 82.76 | **82.49** | 249.16 | 245.60 | 246.41 | **247.06** | 412.21 | 410.10 | 410.68 | **411.00** |
| 5 | 126.92 | 122.99 | 123.48 | **124.46** | 369.05 | 369.35 | 369.84 | **369.41** | 615.53 | 613.58 | 616.05 | **615.05** |
| 6 | 212.90 | 214.20 | 213.87 | **213.66** | 639.60 | 658.78 | 641.41 | **646.59** | 852.27 | 857.48 | 854.32 | **854.69** |

(b) R=5000

Figure 9: Experimental results: The SCAN approach (in seconds).

# 6 Experimental Results

In this section, we shall present experimental results of our proposed approaches. The experiments were conducted on a computer with Pentium 4 3GHz processor and 1GB RAM. The operating system was Windows XP Professional. All the approaches were implemented using Java JDK 1.6. We use Microsoft SQL Server 2005 Developer Edition as our backend database system. All the relations used by our approaches are appropriately indexed.

We use synthetic XML documents that are generated based on the DTD of SIGMOD Record XML [1]. We assume that these documents represent results requested by the *Policy Enforcers*. Each data set has three different versions. Figure 8(a) depicts the characteristics of our data sets. The clearance rules are generated by randomly choosing the objects that co-occur together. The numbers of clearance rules (denoted as $R$) are between 50 and 5,000 rules, and the number of objects in each rule (denoted as $N$) are between 2 and 10. Hence, the total number of sensitive objects in the clearance rules is between 100 and 50,000 (Figure 8(b)).

## 6.1 Execution Times

Figure 9 depicts the performance of the SCAN approach for $R = \{500, 5000\}$ and $N = \{2, 6, 10\}$. Note that the *Avg* column denotes the average execution time for analyzing three XML documents. We notice that the value of $N$ affects the performance. When $N$ is increased, the performance of the SCAN approach becomes slower (observed factor being up to 5.6 times). This is because when the number of rules is fixed and the number of objects for each rule is increased, the total number of objects needed for comparison is increased also. For the same reasons, the value

| Data set | N = 2 | | | | | | | | N = 6 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | | | 3 | | | Avg | 1 | 2 | | | 3 | | | Avg |
| | | A | B | Total | A | B | Total | | | A | B | Total | A | B | Total | |
| 1 | 1.31 | 0.02 | 0.06 | 0.08 | 0.02 | 0.04 | 0.06 | **0.48** | 4.68 | 0.04 | 0.08 | 0.12 | 0.03 | 0.07 | 0.10 | **1.63** |
| 2 | 2.05 | 0.02 | 0.06 | 0.08 | 0.02 | 0.05 | 0.06 | **0.73** | 7.59 | 0.04 | 0.09 | 0.13 | 0.04 | 0.07 | 0.10 | **2.61** |
| 3 | 3.25 | 0.03 | 0.05 | 0.07 | 0.02 | 0.05 | 0.07 | **1.13** | 12.26 | 0.04 | 0.08 | 0.13 | 0.04 | 0.07 | 0.11 | **4.16** |
| 4 | 8.39 | 0.03 | 0.06 | 0.08 | 0.03 | 0.05 | 0.08 | **2.85** | 24.91 | 0.06 | 0.08 | 0.14 | 0.05 | 0.07 | 0.12 | **8.39** |
| 5 | 12.48 | 0.05 | 0.06 | 0.11 | 0.05 | 0.05 | 0.09 | **4.23** | 37.05 | 0.06 | 0.08 | 0.15 | 0.06 | 0.07 | 0.13 | **12.44** |
| 6 | 21.52 | 0.07 | 0.05 | 0.13 | 0.07 | 0.05 | 0.12 | **7.25** | 64.04 | 0.09 | 0.09 | 0.17 | 0.09 | 0.06 | 0.15 | **21.46** |

| Data set | N = 10 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | | | 3 | | | Avg |
| | | A | B | Total | A | B | Total | |
| 1 | 7.79 | 0.05 | 0.12 | 0.17 | 0.05 | 0.10 | 0.15 | **2.70** |
| 2 | 12.53 | 0.05 | 0.12 | 0.17 | 0.05 | 0.09 | 0.14 | **4.28** |
| 3 | 20.39 | 0.06 | 0.12 | 0.17 | 0.06 | 0.10 | 0.16 | **6.91** |
| 4 | 41.36 | 0.07 | 0.12 | 0.19 | 0.07 | 0.09 | 0.17 | **13.91** |
| 5 | 61.71 | 0.08 | 0.12 | 0.20 | 0.08 | 0.09 | 0.18 | **20.69** |
| 6 | 106.72 | 0.10 | 0.12 | 0.22 | 0.10 | 0.09 | 0.20 | **35.71** |

(a) R=500

| Data set | N = 2 | | | | | | | | N = 6 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | | | 3 | | | Avg | 1 | 2 | | | 3 | | | Avg |
| | | A | B | Total | A | B | Total | | | A | B | Total | A | B | Total | |
| 1 | 15.32 | 0.08 | 0.17 | 0.25 | 0.08 | 0.13 | 0.20 | **5.26** | 45.73 | 0.21 | 0.34 | 0.56 | 0.22 | 0.21 | 0.43 | **15.57** |
| 2 | 24.99 | 0.08 | 0.17 | 0.25 | 0.08 | 0.11 | 0.20 | **8.48** | 74.55 | 0.22 | 0.34 | 0.56 | 0.22 | 0.21 | 0.44 | **25.18** |
| 3 | 40.64 | 0.08 | 0.17 | 0.25 | 0.09 | 0.12 | 0.21 | **13.70** | 121.82 | 0.23 | 0.34 | 0.57 | 0.22 | 0.22 | 0.44 | **40.94** |
| 4 | 82.63 | 0.10 | 0.16 | 0.26 | 0.10 | 0.12 | 0.23 | **27.70** | 247.58 | 0.23 | 0.35 | 0.58 | 0.24 | 0.22 | 0.45 | **82.87** |
| 5 | 123.42 | 0.12 | 0.17 | 0.29 | 0.12 | 0.12 | 0.23 | **41.31** | 370.73 | 0.24 | 0.35 | 0.59 | 0.24 | 0.22 | 0.46 | **123.93** |
| 6 | 219.92 | 0.13 | 0.17 | 0.29 | 0.14 | 0.12 | 0.26 | **73.49** | 639.68 | 0.27 | 0.34 | 0.62 | 0.27 | 0.21 | 0.48 | **213.59** |

| Data set | N = 10 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | | | 3 | | | Avg |
| | | A | B | Total | A | B | Total | |
| 1 | 75.86 | 0.30 | 0.55 | 0.86 | 0.30 | 0.33 | 0.63 | **25.78** |
| 2 | 123.75 | 0.30 | 0.56 | 0.86 | 0.31 | 0.33 | 0.64 | **41.75** |
| 3 | 202.21 | 0.31 | 0.56 | 0.87 | 0.30 | 0.34 | 0.64 | **67.90** |
| 4 | 412.95 | 0.36 | 0.56 | 0.92 | 0.37 | 0.34 | 0.70 | **138.19** |
| 5 | 618.45 | 0.38 | 0.57 | 0.94 | 0.38 | 0.33 | 0.72 | **206.70** |
| 6 | 1069.90 | 0.40 | 0.57 | 0.97 | 0.40 | 0.33 | 0.74 | **357.20** |

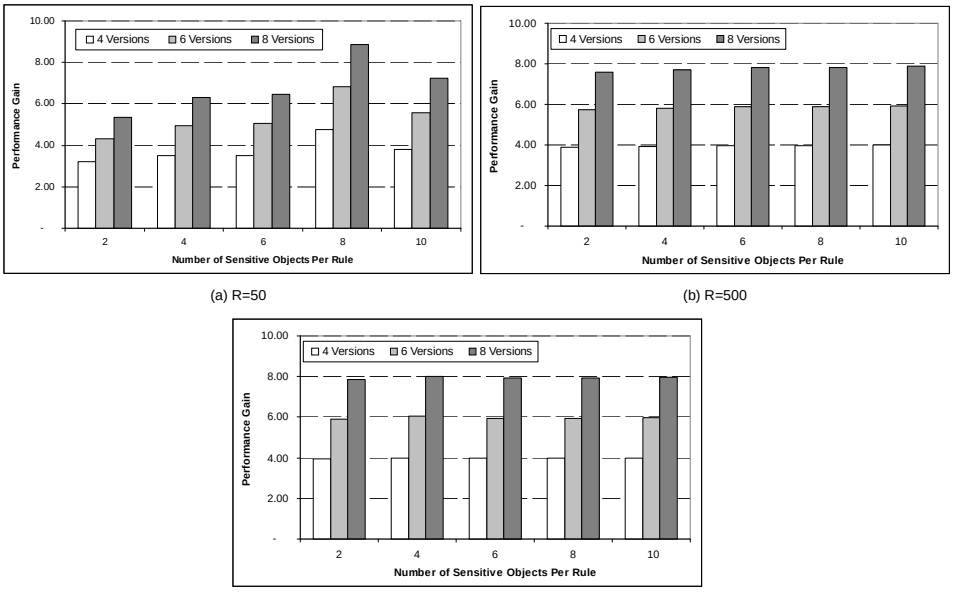
(b) R=5000

Figure 10: Experimental results: The DIFF approach (in seconds).

of $R$ also affects the performance of the SCAN approach.

The performance of the DIFF approach for $R = \{500, 5000\}$ and $N = \{2, 6, 10\}$ is depicted in Figure 10. The "$A$" and "$B$" columns denote the execution times of finding the changes and of analyzing the changes, respectively. Similar to the SCAN approach, as the values of $N$ and $R$ increase, the performance becomes slower.

Let us now compare the performance of the DIFF approach against the SCAN approach. From the results depicted in Figures 9 and 10, we observe that the performance of analyzing the first document version in the DIFF approach is almost similar to the one in the SCAN approach. This is because in both approaches the whole document is analyzed for the clearance rules. On the hand, the performance of analyzing the subsequent versions in the DIFF approach is significantly faster than that of the SCAN approach. This is because the DIFF approach evaluates much lesser number of objects. In the next subsection, we shall further examine the performance gain of the DIFF approach over the SCAN approach.




(a) R=50

(b) R=500

(c) R=5000

Figure 11: Experimental results: Performance gain.

### 6.2 Performance Gain

For this experiment, we use the fifth data set and generate 8 different versions of this document. The $R$ is set to 50, 500, and 5000 rules, and $N$ has values from 2 to 10. The *performance gain* is defined as $\frac{A}{B}$, where $A$ and $B$ are the execution times of the SCAN and the DIFF approaches, respectively. Figure 11 depicts the performance gain of the DIFF approach over the SCAN approach. Observe that when only 4 documents are evaluated, the DIFF approach is between 3.2 – 4.75 times faster than the SCAN approach. However, this gain factor increases as the number of versions of requested query results increases. For instance, when the number of versions is 8 the DIFF approach outperforms the SCAN approach significantly (highest observed factor being 8.85 times).

## 7 Related Work

CAM [26] and ICAM [17] are compact security labeling schemes for XML document instances. The dynamic predicate [19] approach integrates XPath query processing with instance-level access control, where useful information on how access control rules are defined is dynamically summarized. The ideas of these approaches are orthogonal to the DIFF approach of this paper. However, these previous work assume centralized policy enforcement and their outcomes of access control evaluation are binary: grant or deny. In a dynamic and federated access control

scenario, the outcomes are more complex.

Giving privileges based on the content of information objects is called *content-based access control* [12, 22]. In these efforts, access control rules incorporating queries over objects are discussed, but their models are centralized and not sophisticated as our model. Our clearance rules are based on co-occurrence of $n$ sensitive objects in the publishing document. Co-occurrence is important, because an occurrence of each object may not be sensitive, but association of $n$ objects can reveal sensitive information. Gowadia and Farkas [14] discussed access control over such data association on XML documents. But their modeling of data association is based on tree patterns and access control rules are defined at schema-level. On the other hand, our approach is targeted at instance-level rules. Furthermore, these efforts do not efficiently handle dynamic nature of query results by exploiting differences between them.

Several access control models have been proposed for collaborative systems [7, 21]. Cohen et al. [7] propose a family of coalition-based access control models, wherein elements required for a coalition-based access control are layered on top of a simple role-based access control model. Lin et al. [21] present a strategy to decompose a global policy and distribute it to each collaborating party ensuring the autonomy and confidentiality of each involved party and guaranteeing the consistency of decisions. Our effort is orthogonal to these approaches as it can be build on top of a federated XML access control policy. Specifically, we focus on the *Data Provider* module instead of the *Policy Enforcers*. In our approach, the clearance rules and query result set are provided as input to our approach. Note that these rules and results are generated based on the existing access control policies.

# 8 Conclusions and Future Work

In this paper, we have presented a novel and sophisticated approach for automatically evaluating sensitiveness of publishing a batch of XML documents in a federated XML access control environment, and giving security clearance based on the sensitiveness. We use the differences between requested query results for clearance evaluation in our model. Our experimental results show that the proposed diff-based approach outperforms the approach that scans the entire result set every time (the SCAN approach) to determine clearance level. The performances of the SCAN and DIFF approaches are affected by the size of the requested results, the number of versions of results, the number of clearance rules, and the number of sensitive objects for each clearance rule. As part of future work, we would like to extend our framework to support clearance of security objects that are semantically related.

# References

[1] ACM Sigmod, http://www.sigmod.org/sigmod/record/xml/index.html.

[2] Liberty Alliance Project Homepage, http://www.projectliberty.org/.

[3] OpenID Foundation, http://openid.net/.

[4] OASIS eXtensible Access Control Markup Language, http://www.oasis-open.org/committees/xacml/.

[5] M. Benedikt, A. Jeffrey, R. Ley-Wild. Stream Firewalling of XML Constraints. *In ACM SIGMOD 2008*, pp. 487–498, June 2008.

[6] E. Bertino, S. Castano, E. Ferrari. Securing XML documents with Author-X. *IEEE Internet Computing*, 5(3), 2001.

[7] E. Cohen, R. K. Thomas, W. Winsborough, D. Shands. Models for Coalition-based Access Control. *In ACM SACMAT*, 2002.

[8] E. Damiani, S. De Capitani di Vimercati, et al. A Fine-Grained Access Control System for XML Documents. *In ACM Trans. Information and System Security*, Vol. 5, No. 2, pp. 169–202, 2002.

[9] S. De Captiani di Vimercati, S. Foresti, et al. Access Control Models for XML. *In Handbook of Database Security: Applications and Trends* (M. Gertz and S. Jajodia (eds)), Springer, 2008.

[10] W. Fan, C. Y. Chan, M. Garofalakis. Secure XML Querying with Security Views. *In ACM SIGMOD*, June 2004.

[11] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, R. Chandramouli. Proposed NIST standard for role-based access control. *In ACM Trans. Inf. Syst. Secur.*, vol.4, no.3, pp. 224–274, 2001.

[12] L. Giuri, P. lglio. Role templates for content-based access control. *In 2nd ACM workshop on Role-based Access Control*, pp. 153-159, 1997.

[13] G. Gou, R. Chirkova. Efficiently Querying Large XML Data Repositories: A Survey. *In IEEE TKDE*, 19(10), 2007.

[14] V. Gowadia, C. Farkas. Tree Automata for Schema-Level Filtering of XML Associations. *In J. Research and Practice in Info. Tech. (JRPIT)*, 38(1), 2006.

[15] B. Hayes. Cloud Computing. *Communications of the ACM*, Vol. 51, No. 7, pp.9–11, Jul. 2008.

[16] H. He, R. Wong. A Role-based Access Control Model for XML Repositories. *WISE 2000*.

[17] M. Jiang, A. Wai-Chee Fu. Integration and Efficient Lookup of Compressed XML Accessibility Maps. *In IEEE TKDE*, 17(7), 2005.

[18] A. Kern, M. Kuhlmann, A. Schaad, J. Moffett. Observations on the Role Life-Cycle in the Context of Enterprise Security Management. *In ACM SACMAT*, 2002.

[19] J.-G. Lee, K.-Y. Whang, W.-S. Han, I.-Y. Song. The Dynamic Predicate: Integrating Access Control with Query Processing in XML Databases. *The VLDB Journal,* (2007) 16:371.387.

[20] B. Luo, D. Lee, W.-C. Lee, P. Liu. QFilter: Fine-Grained Run-Time XML Access Control Via NFA-Based Query Rewriting. *In ACM CIKM*, Nov. 2004.

[21] D. Lin, P. Rao, E. Bertino et al. Policy Decomposition for Collaborative Access Control. *ACM SACMAT*, 2008.

[22] M. J. Moyer, M. Ahamad. Generalized Role-Based Access Control. *In IEEE Int. Conf. Distributed Comp. Syst. (ICDCS)*, pp.391-398, 2001.

[23] M. Murata, A. Tozawa, M. Kudo, S. Hada. XML Access Control Using Static Analysis. *ACM Trans. Information and System Security*, Vol. 9, No. 3, pp. 292–324, Aug. 2006.

[24] P. Ruch, R. H. Baud, A.M. Raassinoux, P. Bouillon, G. Robert. Medical Document Anonymization with a Semantic Lexicon. *In Amer. Med. Informatics Association Symp.*, pp.729-33, 2000.

[25] S. Prakash, S. S. Bhowmick, S. K. Madria. "SUCXENT: An Efficient Path–Based Approach to Store and Query XML Documents. *In DEXA*, pp. 285-295, 2004.

[26] T.Yu, D. Srivastava, L. V. S. Lakshmanan, H. V. Jagadish. "A Compressed Accessibility Map for XML," *ACM Trans. Database Syst.* 29(2): 363-402, June 2004.