Access Control of XML Documents Considering Update Operations

Chung-Hwan Lim, Seog Park
Department of Computer Science, Sogang University,
1-1 Shinsu-Dong Mapo-Gu Seoul Korea 121-742
+82-02-715-2234
{tuner1004, spark}@dblab.sogang.ac.kr

Sang H. Son
Dept of Computer Science, University of Virginia,
Virginia 22904, U.S.A
+1-434-982-2205
son@cs.virginia.edu

ABSTRACT
As a large quantity of information is presented in XML format on the Web, there are increasing demands for XML security. Until now, research on XML security has been focused on the security of data communication using digital signatures or encryption technologies. As XML is also used for a data representation of data storage, XML security comes to involve not only communication security but also managerial security. Managerial security is guaranteed through access control, but existing XML access control models consider only read queries. These models may make some problems when unauthorized users try to change XML documents or their structure. Therefore the access control of update queries must be executed correctly and efficiently as well as read queries. In this paper, we discuss an XML access control model and propose a technique that supports not only read operations but also update operations. We define new action types to systematically manage complex information of access right and to process various update queries in an efficient manner. Using these action types, the system can save memory and other system resources that are used in DOM-based DTD verification process, and shortens the overall steps of access control by filtering unnecessary queries out at the early stage. Although for read queries the proposed access control model introduces a minor overhead in determining action types, for update queries it shows better performance compared to existing access control models.

Categories and Subject Descriptors
H.2.7 [**Information Systems**]: Database Administration– *Security, integrity, and protection*

General Terms
Security, Performance

Keywords
Access control, XML document, XML update

ACM Workshop on XML Security, October 31,2003, Fairfax VA, USA

1. INTRODUCTION
XML is a markup language for presenting data in Web environments. XML removes the complexity of SGML and the user can define document structures, overcoming the limit of fixed tags in HTML. Due to such advantages, W3C designated XML as the standard for Web data [2, 15]. Since the standardization of XML, many data have been presented in XML, and now various types of XML data exist on the Web. XQuery is a standard query language for querying XML data. Because the language contains only read operations, the user is only allowed in reading XML data. However, the user needs not only read operations but also update operations such as inserting, deleting and modifying XML data. In response to such demands, active research has been being carried out to include operators for updating XML data in XQuery. If update operators are included in XQuery in the near future [13], users can change XML documents and their structures as well as read using XQuery. Like other data, XML data must be protected from exposure to or modification by inadequate queries [11]. However, it is not simple to guarantee the security and integrity of XML data from various queries in Web environments where there are a myriad of users.

In this paper, we propose an access control model to support update operations. Because existing XML access control models provide XML read operations they guarantee data security (confidentiality) only, but if XQuery include update operations, it must guarantee data integrity as well as data confidentiality. First, we define new operators for XML update. Once new update operators are added to an access control model, it becomes complicated to manage information of access rights, and the access control process requires repetitive labeling process and DTD verification process. The repetitive labeling process and DTD verification process consume a lot of memory for XML parsing and DOM tree search, which may degrade system performance. To solve these problems, we define new action types. New action types make it easy to manage information of users and access rights, and supported prompt access control by removing unnecessary parsing and DOM tree search..

This paper is composed as follows. Section 2 reviews previous researches. Section 3 defines XML update operators and action types, and introduces a new DTD verification method using the action types. Then, the algorithm of the proposed access control technique is examined and is applied to sample problems. Section 4 evaluates the proposed access control model and analyzes its performance. Section 5 describes conclusions and future work.

2. RELEVANT WORK

There have been researches on access control [9, 12, 16], and with the rapid development of Web environments XML data access control have been intensively studied [3, 4, 5, 6, 18]. This section reviews previous work on access control and especially on access control considering the characteristics of XML.

2.1 General Access Control

Since computer systems provide multiple applications to multiple users, data security has been a key issue from the beginning. Access control is a means to allow or deny subjects (users or processes) to do operations (read, write, execute, etc.) on objects (data or programs) in the computer system [12, 16]. When a user requests to access information or resources in the computer, the system decides whether to allow the access or not according to access control rules, and if necessary it demands the user to modify the request. Access control rules are determined based on information of access rights, security policy, authorization rules, and other factors depending on the system environment. Through access control, the system can restrict unauthorized users' access to resources in the system and guarantees the confidentiality and integrity of the resources.

2.2 XML Document Access Control

Because an ordinary file has a complex structure, it is difficult to restrict access to only part of the file or to assign access rights to part of the file. Thus access control has to be made against the entire file. However, XML has a tree-shaped hierarchical structure and an XML document is composed of several elements [4, 9]. Each element corresponds to a node in the tree. Because each node represents a part of XML data, access control to a part of data is possible in XML by assigning access rights to elements.

The access control model proposed in [3,4] sets access rights to elements of XML documents and DTD using DOM trees, and control users' access to XML data according to the information of the access rights. Figure 1 shows access control process using the access control model.

The access right information is specified in XAS (XML Access Sheets). If a user requests access to the XML document, the system executes a task as shown in the lower part of Figure 1. First, it parses the XML document to get the DOM tree. Then it sets nodes in the DOM tree with a sign of '+' (allow) or '-' (deny) based on XAS of the XML document and its DTD. Such a task of setting access rights on the nodes of a DOM tree is called labeling. From the labeled DOM tree, finally, nodes with '-' sign are removed and those with '+' sign are shown to the user in XML format. The document that the user sees is the same as a view in that it is part of the entire document, but is different from a view in that it is not preset or stored in a separate storage. Part of a XML document, whose nodes have been removed from the DOM tree, may be invalid in DTD. To solve this problem, a loosening process is necessary as shown in the upper part of Figure 1. The process sets all elements and attributes in DTD to be optional so that existing DTD be preserved even if nodes are removed from the DOM tree. The user who has been authenticated through such a process views only part of the XML document, for which he/she is given a right to access, and the user who has not been authenticated cannot access the other part. This is the way how data security (confidentiality) is guaranteed. However, this model supports only read operator without considering integrity problems caused by XML update operators.

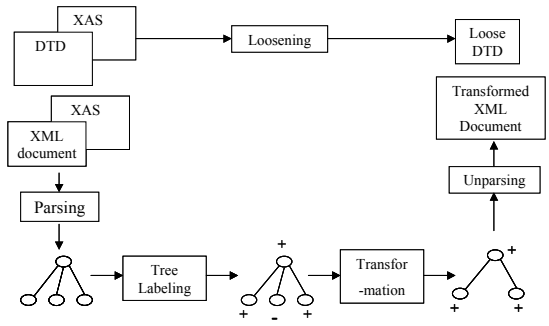


Figure 1. DOM-based Access Control Process

When update operators are added into this model to support update queries, DTD verification process is required to determine whether the structure of the XML document has been changed or not by update queries. Existing DTD verification method parses the XML document and the DOM tree is changed by update queries. The changed DOM tree is compared with the existing DTD by traversing each node in the tree. In an environment that does not allow the change of XML structure, if a XML document is not valid for DTD after the execution of an update query, the update query must be rejected and the subsequent access control must be executed by using the DOM tree before the execution of the update query. However, if it is valid, remaining subsequent access control must be processed using the changed DOM tree. Such a DTD verification process needs to maintain two DOM trees in the memory, so consumes a lot of memory space. Moreover, parsing and searching of DOM tree slow down the speed of access control. In addition, an XML access control system has to execute repetitive labeling process. Because Security Officer set access rights to the unit of operators, the results of labeling vary according to the type of operators included in queries. Therefore, even for the same user, the labeling process is repeated whenever operation to be executed on the XML document changes. Due to the repetitive labeling process, the cost of tree search will continue to rise as new operators are added to the access control model. Therefore, a new access control model, which can support update operators are required in Web environment.

3. XML ACCESS CONTROL

3.1 Assumptions

We will consider several assumptions for update operations:

- XQuery including update operators is being standardized [7].
- There is no semantic dependency among XML elements.
- The contents and structure of XML documents may be changed by users' update queries.
- Update queries are more frequent than read queries.

As an example, consider bank account information in XML format that includes balance elements, each of which also has deposit elements and withdrawal elements as its children elements. The balance is computed by summing deposits and withdrawals. Therefore changes in deposit or withdrawal elements cause changes in the balance element. In addition, if a deposit or withdrawal element is deleted or inserted, the previous balance is

no longer valid. In this way, if there is semantic dependency among elements, update queries causes semantic discrepancy among elements. To maintain semantic dependency, additional works are necessary. However, they are beyond the scope of this paper, and it will not be considered here.

3.2 Access Control Model

We refer types of update operators from [7] and redefine their meanings here as follows.

- Insert (content)
- InsertBefore | After (ref, content)
- Delete (child)
- Replace (child, content)
- Rename (child, name)

Insert is an insert operator, in which content can be a PCDATA, an element or an_attribute. If the XML document contains sequence information, the operator uses InsertBefore or InsertAfter. InsertBefore inserts before the element denoted by ref, and InsertAfter does after the element denoted by ref. Delete is a delete operator, in which child can be PCDATA, an element or an attribute. Replace is a replace operator, in which child can be an element including attributes or PCDATA. Rename is a rename operator, in which child can be an element or an attribute, and name can be a name specified using |(or) in DTD or a new name. To apply the defined XML operators to access control model, they must be distinguished semantically. It is because the same update operator may have different scopes of right verification and execution of operation. Figure 2 shows a DTD graph and a query in which the same update operator has different meanings. The DTD graph below is composed of three elements. There can be only one division and about_div element, but multiple seminar elements. The query on the right is a request to delete about_div and seminar elements [7]. Because the deletion of an about_div violates DTD, it involves a change in the XML structure. However, the deletion of seminar elements is valid with DTD, it changes the DOM tree but not the XML structure. Therefore, the access control process for the first delete operation must include a test for the right to structure change. If the user has a right to change the XML structure, the process must include a task of changing DTD. On the contrary, the access control process for the second delete operation has only to include a test for the right to XML document change. It is because the operation does not involve a change in DTD, so a test for the right to DTD change is not necessary. As illustrated in this example, even operations using the same operator must be distinguished because they may have different access control process

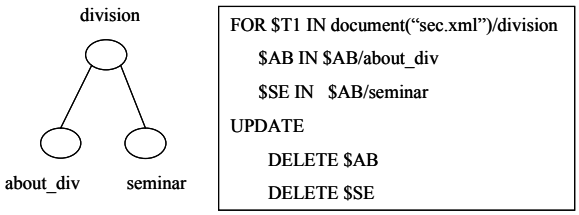


Figure 2. DTD graph and an example of update query

3.2.1 Action Types

If new update operators are added to an access control model, information related to corresponding operators must also include data about access rights. In addition, repetitive DOM tree search and parsing in labeling and DTD verification process degrade system performance. To solve these problems in respects of management and performance and also to distinguish update operators semantically, action types are defined. Action types are sets of operators classified based on to what degree the operators change XML. There are four action types.

- R Type: A set of operators that do not cause any change in XML. Therefore no additional work is necessary. Read operator belongs to this type.
- U Type: A set of operators that cause changes only in XML documents. Parsing is necessary for obtaining a new DOM tree. Replace, rename, insert and delete operators belong to this type.
- D Type: A set of operators that cause changes not only in XML documents but also their structure. It requires DTD change and parsing. Rename, insert and delete operators belong to this type.
- E Type: This is an exceptional type, which is necessary to guarantee access control to the unit of operators. U type and D type belong to this type.

These action types form a hierarchical structure as in Figure 3. The right of high level types (authorization) includes that of low level types. For example, if a user called LIM has a right to execute D type operators about XML elements, LIM also has a right to execute U type and R type operators against the same elements.

Rename, insert and delete operators belong to U type as well as D type. It is because the action type of these operators may be different according to the number of the XML document and DTD. According to whether these operators are U type or D type, the scope of test for access right is different and the procedure to process queries also changes. Therefore, the action type of these operators must be clearly distinguished. The distinction of action types can be done through existing DOM-based DTD verification process. If rename, insert or delete operator is executed, the XML document is changed. If the change of the XML document is valid, the operator is U type, and if invalid, it is D type. As mentioned in section 2.2, however, this method has a problem of consuming a lot of memory. To improve the performance, we need to distinguish action types in an efficient manner.

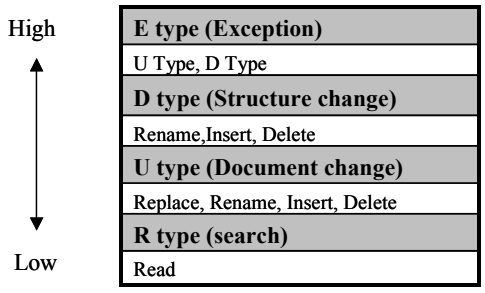


Figure 3. ATH (Action Type Hierarchy)

■ Explicit Distinction of Action Types

The action type of insert, delete and rename operators is determined according to the number of XML elements in XML documents and the sign in their DTD. Table 1 shows the signs that elements can have in DTD and action types corresponding to the signs. If an element in DTD does not have any sign, that is, if the node of DTD and that of XML document are in 1:1 relationship, all the update operators are determined to be D type.

Table 1. Action Type of operators by DTD sign

DTD	None (1:1)	?	+	*	|
Insert	D	U or D	U	U	D
Delete	D	U	U or D	U	D
Rename	D	D	D	D	U or D

Insert of ? sign and delete of + sign can be U type or D type depending on the number of elements in the XML document. For example, if the name element in DTD has ? sign and there is no name element in the XML document, a query to insert a name element is U type. Although it is an insert query, it satisfies the condition of ? sign of DTD, namely 0 or 1, and hence it is valid for DTD. However, if the name element already exists in the XML document, the insert query makes it redundant, which violates DTD. As a result, the insert query becomes D type. | sign affects the determination of the type of rename operator. If the changed name is a name that is already specified in DTD the query is U type, and if it is a new name, the query is D type. Consequently, the type of queries except insert operator of ? sign, delete operator of + sign and rename operator of | sign is explicitly determined by DTD. To determine the action type of the tree operators mentioned above, Both of DTD and the XML document must be referred to. If there is additional information about the XML document, it is not necessary to do DOM tree search every time to refer to the XML document. For insert and delete operators, if the number of elements of ? sign and + sign among elements in the XML document is known, the action type of the operators can be distinguished through a simple algorithm. In addition, for rename operator, if the name specified in | operator is known, the action type of the operator can also be distinguished through a simple algorithm. The number of elements in the XML document can be obtained and kept from initial parsing, and the name specified in | operator can be obtained by referring to DTD and kept in access right information in the form of < | (or) XPath of Element, NameSet >. The following is a method of distinguishing the action type of queries when the queries can be U type or D type:

· Type of insert operator in ? sign
  IF Opt + 1 > 1 THEN D Type
  ELSE U Type (Opt is ? (optional) the number of elements)
· Type of delete operator in + sign
  IF Man - 1 < 1 THEN D Type
  ELSE U Type(Man is + (mandatory) the number of elements)
· Type of rename operator in | sign
  IF Name NameSet THEN U Type
  ELSE D Type(Name is changed name)

In this way, if the action type is determined based on the number of elements in the XML document, update query operators can be distinguished without creating DOM trees. For example, if both the number of speaker nodes and that of member nodes are 0 in the XML document, action types of SEC.DTD in Appendix 2 are as shown in Table 2.

Table 2 shows that a query to insert a new member element is U type, and one to delete a member element is D type. If the environment allows the change of XML structure, the access control process must include a test for the user's right to change the structure and DTD update for queries to delete a member element. Without such an action type table, parsing and tree search have to be performed for every query to refer to DTD and the XML document for determining the access right and the access procedure.

Table 2. Action Type Table

DTD	Insert	Delete	Rename
division	D	D	D
about_div	D	D	D
seminar (*)	U	U	D
address	D	D	D
member (+)	U	D	D
contact	D	D	D
title	D	D	D
speaker (?)	U	U	D

3.2.2 Access Right Information

The XML document access control model including update operators is composed of six components as follows.

· Subject : User name, IP address or symbolic name
· Object : XPath 1.0
· Action : Read, Insert, Delete, Replace, Rename
· Action Type : R, D, U, E
· Sign : + / -
· Type : LDH, RDH, L, R, LD, RD, LS, RS

Subject is subjects who access XML documents. They support the group and pattern of users. Object is elements in XML documents, which are represented as XPath. Action is operation that subjects can execute, and Action Type is a set of operators. Sign represents the acceptance or denial of rights, and Type means the attribute value of rights. In this model, access right information is expressed in the following form.

< Subject, Object, Action, Action Mode, Propagation, Option >

Action mode is the combination of Action Type and Sign like 'U+'. Propagation has a value of R (recursive) or L (local). Its value depends on whether the right is propagated or not. In addition, Option represents the priority of access rights. Access right information like this must be created only by Security Officer. Security Officer must specify the action type of each operator in access right information on initial access right setting. For this work, the action type table is referred to.

3.2.3 Labeling

■ Propagation Rules

Because XML is of hierarchical tree structure, rights to higher nodes may influence those to lower nodes, and even the right of the same user may be different according to the group the user belongs to, IP address and symbolic name. In addition, rights are set simultaneously for XML documents and DTD. For this reason, access rights to the same node access, which are defined by the Security Officer in access right information, may conflict with each other [14]. Because a node is ultimately given a right, the conflict must be resolved. Propagation rules determine the priority of rights when right conflicts occur. Cases in which right conflicts may occur are classified into five types as follows according to the information of subjects, the sign of action type, the position of ATH and the object of right information. The following are propagation rules to determine the priority of rights for the five types of cases where right conflicts may occur.

· *Rule 1:* When the right information of a subject is in conflict with that of another subject, the right information with more detailed information about the subject has priority.
· *Rule 2:* When the right information of a XML document is in conflict with that of DTD, the former has priority. However, if the type in the DTD right information is set for 'hard' the DTD right information has priority.
· *Rule 3:* When different signs are in conflict with each other, if they are of the same action type, '-' has priority, and if not, the action type with lower ATH has priority.
· *Rule 4:* When two + signs with different action types are in conflict with each other, the action type with higher ATH has priority. At that time, if the action type becomes higher than the degree of the user's right, the action type of lower ATH has priority. If two - signs are in conflict with each other, the action type with lower ATH has priority.
· *Rule 5:* When E type and another action type are in conflict with each other, if they have the same sign, the action type other than E type has priority, and if they have different signs E type has priority.

■ Labeling Algorithm

Labeling is a process to set access rights on nodes in a DOM tree, which queries request to access, using access right information defined by the Security Officer [3,4]. The labeled right information is used in determining whether a user's query is approved or rejected. Because labeling is done by operator in existing labeling process, the labeling process has to be repeated as many times as the number of kinds of operators included in a query. To remove repetitive labeling, we propose a labeling algorithm by action type. Because labeling by action type produces the same result regardless of the kind of operators, labeling process is required just once. The Security Officer can label using operator as in previous models using E action type. Because, in the propagation rules, the access right information of XML documents has priority over the access right information of DTD, the two must be distinguished from each other.
To achieve that, access rights corresponding to XML documents are kept in A_xml, and those corresponding to A_dtd. If the Security Officer did not explicitly set a right on a specific element, a default access right is assigned to the element. Therefore, all elements have their action mode. The upper part of Figure 4 explains the variables and functions used in the algorithm, and the lower part is the labeling algorithm using the action types.

```
· A_xml := {a ϵ Auth | rq <= subject(a), uri(object(a)) = URI}
· A_dtd := {a ϵ Auth | rq <= subject(a), uri(object(a)) = dtd(URI)}
· r is the root node of URI document, n is a node other than r,
  p is the parent node of n
· AM( ): returns the action mode of a node specified in the
  authorization rule,
· default( ): returns the default action mode of a node,
· type ( ): returns the type specified in the authorization rule,
· propagation_rule( ): returns the action mode determined by
  propagation rules

1. label (r)
2. For each c children (r) do label (n, p)

   Procedure label (r)
   L1r = AM (r) in A_dtd
   L2r = AM (r) in A_xml
   if L1r U L2r = { }, Lr = default (r)
   else Lr = propagation_rule ([L1r , L2r])

   Procedure label (n,p)
   L1n = AM (n) in A_dtd
   L2n = AM (n) in A_xml
   if type (p) in {R, RD, RDH, RS}
      if L1n & L2n = { }, Ln = Lp
      else Ln = propagation_rule([Lp, L1n, L2n])
   else
        if L1n & L2n = { }, Ln = default (n)
        else Ln = propagation_rule([L1n, L2n])
```

Figure 4. Labeling Algorithm

3.2.4 Access Control Technique

Existing XML access control techniques determine whether to allow a query to access or not after labeling the DOM tree. Thus the system has to keep all information necessary for right tests to the end and unnecessary right tests were repeated [3, 4]. Such extra tasks slow down the speed of access control. To address this problem, the proposed access control technique divides the access control process into two steps. When two steps are used in access control, queries violating the right are removed in the first step and, as a result, no additional work is necessary for the denied queries.

Step 1: Access control step 1 first removes the operators of queries that violate the user's right. A user's right is assigned based on access right information when the user requests access to XML documents. If the assigned right is U type, the user cannot make a query of D type, since it is located higher than that of U type in ATH. That is, if a user without a right to change XML structure requests a query that causes a change in XML structure, the query is excluded in advance, which removes unnecessary right tests and relevant tasks.

Step 2: A query approved in access control step 1 goes through a final right test to decide whether the query should be processed or not. While right tests are performed to the entire XML document in access control step 1, they are performed to elements in the XML document in access control step 2. To test a user's access

right to an element requested by a query, the labeled DOM tree must be traversed. Here, since queries without a right are excluded in advance in access control step 1, tree searches for them are not necessary.

■ Example of Access Control Technique

Consider an XML document, DTD and access right information as shown in Appendix 2.

The access right information shows that ADMIN has a right to insert a division node, and that the insert is D type, so it changes the structure of the XML document. In addition, it shows that KANG cannot read sub-elements including seminar element. The query in Figure 5 is to insert an element having the value 'database', as a sub-node of about_div, to replace the content of address with 'Pusan', and to delete a contact element.

Name : LIM
IP : 163.239.131.116
Group : DBLAB
Level : U

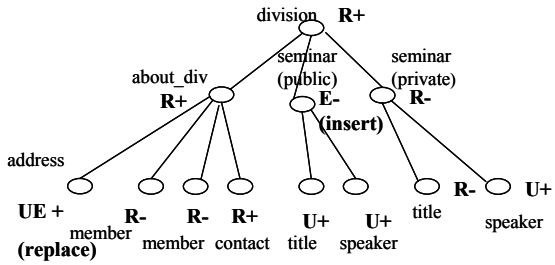
```
FOR $DIV IN document("SEC.XML)/division
UPDATE $DIV
  DELETE $DIV                          → D type : reject
  FOR $ABV IN document("SEC.XML")/division/about_div
    $ADD IN $ABV/address
    $CON IN $ABV/contact
  UPDATE $ABV
    INSERT <first> database </first>   → D type : reject
    REPLACE $ ADD WITH <address> pusan </address>
    DELETE $CON                        → U type : 1st approval
```

DTD	Insert	Delete	Rename	Replace
division	D	D	D	U
about_div	D	D	D	U
seminar (*)	U	U	D	U
address | tel	D	D	U	U
member (+)	U	U	D	U
contact (?)	D	U	D	U
title	D	D	D	U
speaker	D	D	D	U

< Action Type Decision >

Figure 5. The process of access control step 1

If a user called LIM makes such a query, requests for access by operators without a right are rejected in access control step 1. The user's right is referred to access right information, and the type of operators included in the query can be seen in the action type table. Because user LIM's right is U type, he/she does not have a right to change the structure of the XML document. However, the first delete and insert operators in the query are D type, which causes a change in the structure. Accordingly, these operators are rejected and only U type operators get the 1st approval. Then, in access control step 2, the rights of the approved operators are checked against rights set on the DOM tree and only those with access right are executed. Operators approved in access control step 1 requested access to an address element and a contact element. In Figure 6, the address element is set to UE+ (replace) and the contact element to R+. Therefore, in a nested query, statements including replace are accepted and executed, but those including delete are rejected. It is because the user does not have a right to read a contact element, and consequently, to delete it.



REPLACE $ ADD WITH <address> Pusan </> ➔ executed

DELETE $CON ➔ reject

Figure 6. The process of access control step 2

The access control process composed of two steps reduces unnecessary tasks. Table 3 is the comparison between existing access control technique and the technique proposed in this paper when they are used in processing the query of Figure 7 in an environment that allows the change of XML structure.

```
FOR $SPRIV $DIV/seminar[@cat="private"]
    $TI $SPRIV/titile
    $SPK $SPRIV/speaker
UPDATE
  DELETE $SPK   → U
  INSERT $TI    → D
```

1) Parsing (initial DOM tree creation)
2) Labeling
3) DTD Validation
4) Check for right to change DTD structure
5) Change DTD
6) Change DOCUMENT
7) Parsing (to reflect the changed value)

User's Query **Query Execution procedure**

Figure 7. General Access Control Process

When the user's right is U type, his/her request for insert operation is rejected in access control step 1, so no extra work is necessary for the insert operation.

Table 3. Comparison of access control processes

User information	Securing XML Documents [4]	Proposed technique
Read permitted (R)	1-<2>-(2)	1-2
Structure change inhibited (U)	1-<2><3><6><7>-(2)-(3)-(4)	1-2 -<6>-7
Structure change permitted (D)	1-<2><3><6><7>-(2)-(3)-(4)-(5)-(6)-(7)	1-2 -<6>-(5)-(6)-7

Numbers within < > in Table 3 denote procedure numbers in delete, and those within ( ) are procedure numbers for executing insert operation. In the table above, if a user' right is U type and the access control technique employed in [4] is used, the following procedure is required. First, parse the XML document to test the right of delete operator, and label rights on the DOM tree using access right information. Then, test through DTD verification if a change in the structure occurs. Execute the delete operation if the result of DTD verification shows that the operation does not cause a change in the structure. After executing delete operation, parse the XML document to get a new DOM tree. Then test the right of insert operator, by labeling rights on the

DOM tree and verifying DTD. Insert operator is rejected because the result of DTD verification shows that insert operator causes a change in DTD. However, the proposed access control technique rejects insert operator in access control step 1, it has only to execute access control with regard to delete operation. Accordingly, the proposed technique needs to parse the XML document and label rights only for testing rights to elements requested by delete operator.

3.3 System Structure

The structure of the proposed access control system is shown in Figure 8.

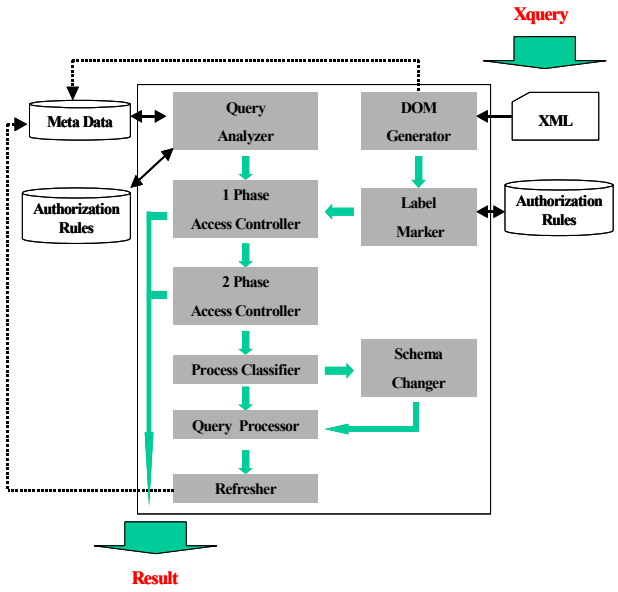


Figure 8. Access control system structure

A user requests access to an element in a XML document using XQuery. The access control system that received the query parses the XML document using DOM generator and creates a DOM tree. In the parsing process, the number of elements with ? sign and + sign are counted and kept in DTD as meta data to determine action types in the action type table. Referring to metadata and access right information abstracted through the parsing, Query Analyzer assigns action types to operators in the query and the user. Then Label Maker labels the DOM tree with rights corresponding to the user, the IP address and the symbolic name. A test for the right to the entire XML document is performed in Phase 1 Access Controller, and a test for the right to the XML elements requested in Phase 2 Access controller. Process Classifier checks whether a change has occurred in XML structure or not. If a change has occurred in the structure of the XML document, DTD is updated in Schema Changer and then the query is executed by Query Processor. If the XML document or its structure has been changed by the query, Refresher parses the DOM tree to update the change. If the user's query is entirely rejected in the step of access control, the query is processed no longer and a result message is returned to the user.

4. EVALUATION

Because existing models [3, 4, 17] include only read operations, they cannot control access to update operations that always happen in the real world. However, the proposed access control model can control not only access to read operations but also access to update operations. Moreover, because the proposed model supports an environment that allows changes in a XML document and its structure, it can exercise access control against valid XML documents and well-formed XML documents. Therefore, with the proposed model, it is possible to manage efficiently various types of complex information of access right and it is unnecessary to assign/revoke repetitively a right to/from each operator whenever the Security Officer assigns to/revoke from a user a right to a XML document. Moreover, the proposed model is advantageous in that its two step sequence of access control process remove unnecessary works.

We analyze the XML access control technique proposed in [3, 4] and our model. The evaluation criterion is overall access control time for executing a given query. Although the model proposed in [3, 4] does not support update operators, it is said to conduct access control on update operators in the same way as it does on read operators. Thus, this evaluation assumes that update operators go through the same access control process as that for read operators in the model. This evaluation uses XML documents and DTD provided in XMark [1] for XML benchmarking. Queries for performance evaluation are Q1, Q2 and Q3 as follows.

· Read query Q1 :

```
FOR $O IN document(auction.xml)/site/people/person
RETURN
<homepage> $O/homepage </homepage>
```

· Update query Q2 :

```
FOR $C
IN document(auction.xml)/site/people/person/creditcard
UPDATE
  DELETE $C
```

· Compound query Q3 :

```
FOR $P IN document(auction.xml)/site/people/person
$N IN $P/name
$CT IN $P/city
$PH IN $P/phone
UPDATE
  DELETE $N
  REPLACE $CT WITH <city> California </city>
  INSERT <person> AFTER $P
     <name> SHAN </name>
     <email address>tuner@hotmail.com </email address>
          </person>
```

In addition, the following parameters are used in measuring access control time for these queries.

· FSD (Full Search DTD): Time to search DTD
· ARP (Authorization Rules Parsing): Time to parse access right information
· FSAR (Full Search Authorization Rules): Time to search access right information
· XP (XML Parsing) : Time to parse XML documents
· FSX (Full Search XML) : Time to search the entire XML documents
· SN (Search Node) : Time to search elements included in the query

The measuring unit of these parameters is millisecond (msec). Table 4 shows parameter values corresponding to authorization.xas (273KB), access right information of XML documents of different sizes, which are auction3.xml (3M), auction5.xml (5M) and auction10.xml (10M), and DTD, which is auction.dtd (5.16KB), resulting from a test using a computer with 1.7 GHz Pentium IV CPU and 512MB memory. DTD of XML data in the test is in Appendix 1.

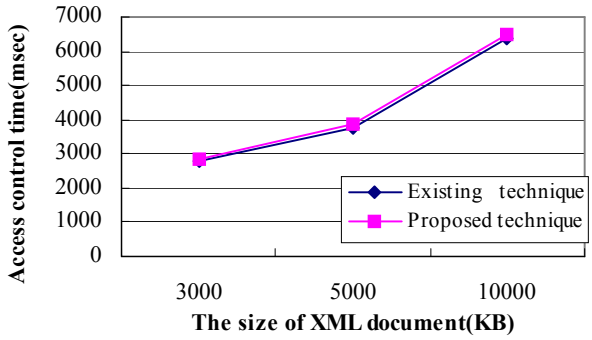
Table 4. Values of parameters

	XP	FSX	SN (homepage)	SN (credit)	SN (name)	SN (phone)	SN (city)
auction3.xml (3MB)	1672	437	16	15	16	16	16
auction5.xml (5MB)	2406	703	20	16	17	16	16
aution10.xml (10MB)	4766	953	31	32	32	31	31
authorization.xas (273KB) : **ARP** (534), **FSAR** (109) **auction.dtd (5.16KB)** : **FSD** (15) (msec)							

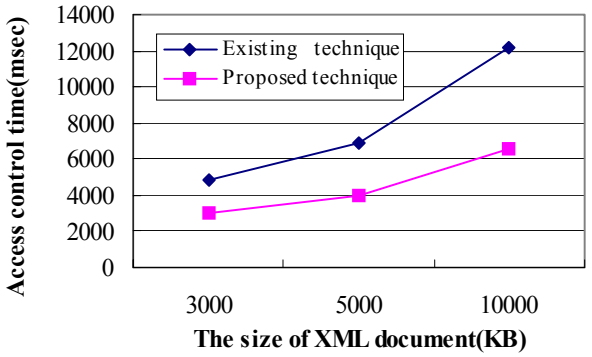
As document size increases, parsing time (XP) and tree search time (FSX, SN) also increase. In addition, of the parameters, parsing time (XP) is the most influential variable on system performance. The entire access control time for query Q1, Q2 and Q3 can be obtained from the formulas below.

· Existing technique [4] :
Parsing (XP)
+ Labeling (ARP + (Operator Type * (FSAR + FSX + SN)))
[+ DTD Verification (Update Operator Type * (XP + FSX + FSD))]
· Proposed technique :
Action Type Assignment (ARP + FSAR)
+ Parsing (XP)
+ Labeling (FSAR + FSX + (Operator Type * SN))
[+ DTD Verification (FSAR)]

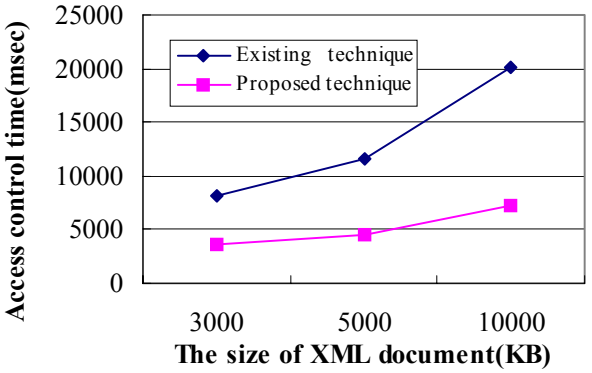
Figure 9 is the result of comparing the access control time of query Q1, Q2 and Q3 using the proposed technique with that of the existing technique. For read query (a), the existing technique takes access control time of XP + ARP + FSAR + FSX + SN, while the proposed technique does access control time of ARP + FSAR + XP + FSAR + FSX+ SN. The proposed technique must search access right information authorization.xas to assign the action type of operators. Because authorization.xas is in XML format, the system must parse authorization.xas (ARP) and search the tree (FSAR) to refer to access control information. Because the parsing of authorization.xas is necessary in labeling process for referring to access right information, read query (a) has overhead of FSAR (109 msec). For update query (b), the existing technique takes access control time of XP + ARP + FSAR + FSX + SN + XP + FSX + FSD, while the proposed technique does access control time of ARP + FSAR + XP + FSAR + FSX + SN + FSAR. A query including update operators needs DTD verification process to check if a change occurs in the structure by the update operators. For this process, the existing technique executes update operation, parses updated XML documents (XP), and visits all nodes in the new DOM tree and compare them with DTD (FSX + FSD). However, using action types, the proposed technique can determine whether a change occurs in the structure or not. Accordingly, the DTD verification process takes only the time to search access right information (FSAR). The compound query (c) includes three types of operators, which are delete, replace and insert. In this case, the existing technique takes access control time of XP + ARP + 3 * (FSX + FSAR + SN) + 2 * (XP + FSX + FSD) and the proposed technique does access control time of ARP + FSAR + XP + FSX + FSAR + (3 * SN) + (2 * FSAR).



(a) Access control time for a read query (Q1)



(b) Access control time for an update query (Q2)



(c) Access control time for a compound query (Q3)

Figure 9. Comparison of Access control time

Because the existing technique performs labeling to the unit of operators it has to do labeling as many times as the number of types of operators (3 * (FSX + FSAR + SN), but because the proposed technique performs labeling to the unit of action types it needs only a time of labeling (FSX + FSAR) and three times of search to view the labeling result of the node requested by each operator (3 * SN). Because replace operator changes only elements or attribute values, it does not cause a change in the structure. Accordingly, replace operator does not need DTD verification. However, insert and delete operators may cause a change in the structure, they need DTD verification. Therefore, the existing technique takes time for two times of parsing and tree search (2 * (XP + FSX + FSD), but the proposed technique takes time for only two times of search of access control information (2 * FSAR).

Although the proposed access control technique has overhead for assigning action types to operators, it save a lot of time in access control over update queries and compound queries including update operators. Accordingly, the proposed access control technique has better performance in an environment where update queries occur frequently.

5. CONCLUSION

As Web environment develops and XML becomes the standard of Web data presentation, users demand various types of queries other than read queries including update queries such and insert and delete. In addition, as XML becomes popular, the security of XML data becomes an essential requirement. Because existing XML access control models do not include update operation, they cannot support efficient access control over update queries.

In this paper, we proposed an XML access control model and technique for the real world environment where update queries are used for storing and processing XML data. To develop a XML model that supports update operation, we defined XML update operators and included them in the access control model. In addition, to address problems in performance that are caused by the addition of update operators to the access control model, we defined new action types. Using the action types defined, complex information of access right can be managed efficiently. Moreover, the proposed model improved existing DOM-based DTD verification process, and addressed problems in performance by removing repetitive tree search. Our model divides access control process into two steps. The division removes unnecessary tasks related to operators rejected in the first step, and consequently, shortens the procedure of access control. Although our model has minor overhead for read queries, it shows improved performance for update queries. Accordingly, the proposed model is suitable for environments such as ordering system, in which users' rights are clearly distinguished and most queries involve update operation, or mirroring system, which invariably executes update queries to guarantee the consistency of data. Because our access control model is also based on DOM, however, it has to search trees, which limits the performance improvement.

6. ACKNOWLEDGMENTS

This work was supported by grant No.R01-2003-000-10395-0 from the Basic Research Program of the Korea Science & Engineering Foundation, and, in part, by grant EIA-9900895 and IIS-0208758 from NSF.

7. REFERENCES

[1] Albrecht Schmidt, Florian Waas, Martin Kersten, Michael J.Carey, Ioana Manolescu and Ralph Busse, "Xmark: A Benchmark for XML Data Management," Proc. VLDB, Hong Kong, China, 2002.

[2] David Hunter (Editor), et al. Beginning XML. wrox, 2001.

[3] E.Damiani, S.Vimercati, S.Paraboaschi and P.Samarati, "Design and implementation of an access processor for xml documents," In Proceedings of the 9th international WWW conference, Amsterdam, May 2000.

[4] E.Damiani, S.Vimercati, S.Paraboschi, and P.Samarati, Securing xml document," In Proceedings of the 2000 International Conference on Extending Database Technology(EDBT2000), pp 121-135, Konstan, Germany, March, 2000.

[5] E. Damiani, S. De Capitani di Vimercati, E. Fernndez-Medina, P. Samarati, "An Access Control System for SVG Documents," In Proceedings of the Sixteenth Annual IFIP WG 11.3 Working Conference on Data and Application Security, King's College, University of Cambridge, UK, July 29-31, 2002.

[6] Elisa Bertino, Elena Ferrari, " Secure and Selective Dissemination of XML Documents", ACM Transaction on Information and System, Vol 5,No. 3, August 2002.

[7] Igor Tatarinov, Zachary G. Ives, Alon Y.Halevy, Daniel S.Weld, "Updating XML," ACM SIGMOD, pp 413-424, Santa Barbara, California, USA, May, 2001.

[8] Kevin Williams (Editor), et al. Professional XML Databases. wrox, 2001.

[9] Oracle, Database Security in Oracle 8i, February 1999.

[10] P.Samarati, E.Bertino, and S.Jajodia, "An Authorization Model for a Distritbuted Hypertext System," IEEE TKDE, 8(4):555-562, August 1996.

[11] Rutgers Security Team. WWW Security: A survey, 1999. http://www-ns.rutgers.edu/www- security/.

[12] S.Castano, M.Fugini, G.. Martella and P.Samarati, Database Security, Addision-Wesley, 1995.

[13] S.Boag, D.Chamberlin, M.F. Fernandez, D.Florescu, J.Robie, J.simeon, and M.Stefanescu, "XQuery 1.0:An XML query language," http://www.w3.org/TR/xquery/, 30 April 2002. W3C working draft.

[14] S.Jajodia, P.samarati, and V.S Subrahmanian, "A Logical Language for Expressing Authorization," In Proceeding of the IEEE Symposium on Security and Privacy, pages 31 - 42, Oakland, CA, May 1997.

[15] T.Bray et.al. (ed.). Extensible Markup Language (XML) 1.0, World Wide Web Consortium (W3C), February 1998. http://www.w3.org/TR/REC-xml.

[16] T.F. Lunt, "Access Control Policies for Database Systems," In C.E. Landwehr (editor), Database Security, II:status and Prospects, North-Holland, Amsterdam, 1989.

[17] Yue Wang, Kian-Lee Tan, "A Scalable XML Access Control System," In Proceedings of the 10th International WWW Conference(Poster), 2001.

[18] M. Kudo, S. Hada, "XML Document Security Based on Provisional Authorizations", In Proc. of the 7th ACM Conference on Computer and Communications Security, Athens, Greece, Nov. 2000.

APPENDIX

[APPENDIX 1] AUCTION.DTD

```
<!ELEMENT site regions, categories, catgraph, people, open_auctions, closed_auctions)>
<!ELEMENT categories (category+)>
<!ELEMENT category (name, description)>
<!ATTLIST category id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (text | parlist)>
<!ELEMENT text (#PCDATA | bold | keyword | emph)*>
<!ELEMENT bold (#PCDATA | bold | keyword | emph)*>
<!ELEMENT keyword (#PCDATA | bold | keyword | emph)*>
<!ELEMENT emph (#PCDATA | bold | keyword | emph)*>
<!ELEMENT parlist (listitem)*>
<!ELEMENT listitem (text | parlist)*>
<!ELEMENT catgraph (edge*)>
<!ELEMENT edge EMPTY>
<!ATTLIST dge rom IDREF #REQUIRED to IDREF #REQUIRED>
<!ELEMENT regions africa, asia, australia, europe, namerica, samerica)>
<!ELEMENT africa (item*)>
<!ELEMENT asia (item*)>
<!ELEMENT australia (item*)>
<!ELEMENT namerica (item*)>
<!ELEMENT samerica (item*)>
<!ELEMENT europe (item*)>
<!ELEMENT item ,location, quantity, name, payment, description, shipping, incategory+, mailbox)>
<!ATTLIST item id ID #REQUIRED, featured CDATA #IMPLIED>
<!ELEMENT location (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>
<!ELEMENT payment (#PCDATA)>
<!ELEMENT shipping (#PCDATA)>
<!ELEMENT reserve (#PCDATA)>
<!ELEMENT incategory EMPTY>
<!ATTLIST incategory category IDREF #REQUIRED>
<!ELEMENT mailbox (mail*)>
<!ELEMENT mail (from, to, date, text)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT itemref EMPTY>
<!ATTLIST itemref item IDREF #REQUIRED>
<!ELEMENT personref EMPTY>
<!ATTLIST personref person IDREF #REQUIRED>
<!ELEMENT people (person*)>
<!ELEMENT person (name, emailaddress, phone?, address?, homepage?, creditcard?, profile?, watches?)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT emailaddress (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT address (street, city, country, province?, zipcode)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT province (#PCDATA)>
<!ELEMENT zipcode (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT homepage (#PCDATA)>
<!ELEMENT creditcard (#PCDATA)>
<!ELEMENT profile (interest*, education?, gender?, business, age?)>
<!ATTLIST profile income CDATA #IMPLIED>
<!ELEMENT interest EMPTY>
<!ATTLIST interest category IDREF #REQUIRED>
<!ELEMENT education (#PCDATA)>
<!ELEMENT income (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT business (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT watches (watch*)>
<!ELEMENT watch EMPTY>
<!ATTLIST watch open_auction IDREF #REQUIRED>
<!ELEMENT open_auctions (open_auction*)>
<!ELEMENT open_auction (initial, reserve?, bidder*, current, privacy?, itemref, seller, annotation, quantity, type, interval)>
<!ATTLIST open_auction id ID #REQUIRED>
<!ELEMENT privacy (#PCDATA)>
<!ELEMENT initial (#PCDATA)>
<!ELEMENT bidder (date, time, personref, increase)>
<!ELEMENT seller EMPTY>
<!ATTLIST seller person IDREF #REQUIRED>
<!ELEMENT current (#PCDATA)>
<!ELEMENT increase (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT interval (start, end)>
<!ELEMENT start (#PCDATA)>
<!ELEMENT end (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT amount (#PCDATA)>
<!ELEMENT closed_auctions (closed_auction*)>
<!ELEMENT closed_auction (seller, buyer, itemref, price, date, quantity, type, annotation?)>
<!ELEMENT buyer EMPTY>
<!ATTLIST buyer person IDREF #REQUIRED>
<!ELEMENT price (#PCDATA)>
<!ELEMENT annotation (author, description?, happiness)>
<!ELEMENT author EMPTY>
<!ATTLIST author person IDREF #REQUIRED>
<!ELEMENT happiness (#PCDATA)>
```

[APPENDIX 2] Example for Access Control of XML

SEC.XML

```
<division name = “DBLAB”>
    <about_div>
     <address> SEOUL </address>
     <member> SONG </member>
     <member>LIM</member>
     <contact> tuner1004@dblab.sogang.ac.kr </contact>
  </about_div>
  <seminar category = “public”>
    <title> access control of XML documents  </title>
    <speaker> LIM </speaker>
  </seminar>
  <seminar category = “private”>
    <title> repetitive color exposure effect </title>
    <speaker>  SONG </speaker>
  </seminar>
</division>
```

SEC.DTD

<! ELEMENT division (about_div, seminar*>
<! ELEMENT about_div (address, member +, contact)>
<! ELEMENT address (#PCDATA)>
<! ELEMENT member (#PCDATA)>
<! ELEMENT contact (#PCDATA)>
<! ELEMENT seminar (title, speaker ?)>
<! ELEMENT title <#PCDATA>
<! ELEMENT speaker <#PCDATA>
<!ATTLIST division name CDATA #REQUIRE>
<!ATTLIST seminar category (public | private) #REQUIRE>

ACESS RIGHT INFORMATION

<<ADMIN, *, *>, sec.xml:/divisoin, insert, D+, R, -- >
<<PUBLIC, *,*>, sec.dtd:/division/about_div, read, R+ , R -- >
<<PUBLIC, *,*>, sec. xml:/division/seminar[@cat=private], read, R-, R, -- >
<<LIM, *.*>, sec. xml:/division/seminar[@cat=private]/speaker, read, R+, L , -- >
<<LIM.163.239.*,*>, sec.dtd:/division/seminar, delete, U+, R, -->
<<LIM, *.*>, sec. xml:/division/seminar[@cat=public]/speaker, insert, U+, L, -- >
<<LIM, 163.239.*.*>, sec.dtd:/division/about_div/member, read, R-, L, Hard >
<< LIM, 163.239.*.*>, sec.xml:/division/about_div/member, read, R+, L, -- >
<<DBLAB, *,*>, sec.dtd:/division/seminar[@cat=public]/title, rename, DE+, L , -- >
<<SOGANG, *,*>, sec.dtd:/division/about_div/contact, insert, U+, L , -- >
<<PUBLIC, 163.239.*>, sec.xml:/division/about_div/address, delete, U-, L, -- >
<<LIM, 163.*>, sec.xml:/division/about_div/address, replace, UE+, L, -- >
<<*.163.239.131.116,*>, sec.xml:/division/about_div/contact, insert, U-, L, -- >
<<KANG, *,*>, sec://xml:/division/seminar, read, R-, R, -- >