Access Control for Energy-Efficient Query Processing

Dongchan An and Seog Park

***Abstract*—An access control method has been previously performed only focused on safety, and thus not much effort has been done to consider access control in terms of energy efficiency. In this paper, we proposed a method for an energy-efficient query processing of XML data streams, such as a personal digital assistant and a portable terminal, at the client side with limited resources. Specifically, we proposed an access control processing that possesses a small overhead for attaining a secure result in a limited memory and a method to enhance the performance, finding the parts capable of optimizing each processing step for offsetting the overhead caused by the addition of access control processing. Our new method is analyzed through an experiments.**

***Index Terms*—Energy-efficient, query processing, access control, XML**

I. Introduction

The Fig. 1 shows a complete document type definition (DTD) structure of an information that a server receives from patients and an example of an access control rule of doctor A and doctor B. The access control rule on XML [1] documents can be shown as an XPath expression [2], such as queries, and this access control rule is applied to a terminal of individuals as shown in Fig. 1.

The doctor of a hospital using a portable terminal to check patient information. Each patient has one doctor in charge, whereas some may have several doctors according to their respective specialization. The doctor can check the information of patients, but they cannot look at the disease information of patients not under their care. From the patient's point of view, information about a patient's mental illness or disease that can affect one's social life should only be available to authorized individuals. However, the server cannot transmit separated data to all the clients receiving the data because managing a rule of access control to satisfy various users is difficult, and a considerable expense is needed to broadcast the results of all the various cases in a stream environment where data must be sent continuously. Therefore, even though a server broadcasts all the information, a client must read only the contents that can be accessed and answer the queries of a user [3].

The traditional method of processing query is to make sure that access control is confirmed if the query domain of the user is an accessible domain for the said user through access control technology. When the query from the user has been entered inputted and has passed, the result of the queries must be produced through the query processor. Likewise, the application of a method that independently performs access control and query processing is not feasible because an access control engine and a query processing engine are required to perform by a client with limited resources. So we need an access control for energy-efficient query processing.

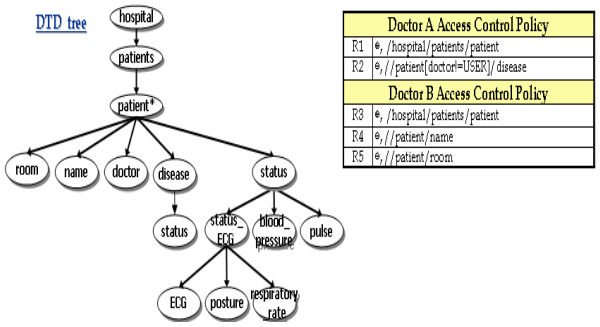

Fig. 1. DTD structure of XML data streams by a client with limited resources.

We propose a method to perform simultaneously an access control technology and a query processing technology by a client with limited resources. First, an access control processing method with fewer overheads to produce secure results within a limited memory is proposed. Second, this work aims to reduce the overheads of performance due to the addition of access control by simultaneously performing energy-efficient query processing during access control processing.

II. Related Work

A traditional XML access control enforcement mechanism [4]-[7] is a view-based enforcement mechanism. A view is created by rules of access control for each user, and queries are processed as the controlling access of users based on the created view. Many useful algorithms that can evaluate a view with tree labeling scheme have been proposed, but the problem is the high cost of creating and maintaining a view. It also develops a scalability problem when the number of users increases [8], [9].

The client-based XML document streams access control [10] and decides an access status for currently input data by avoiding the rewriting of queries and performing simultaneous access control rules with query evaluations. It is a technique that obtains a final secure XML document for final user queries by passing the XML document corresponding to user queries to the automata (access rules automata (ARA)), which is unlike Luo [11]'s method that remakes the user queries by creating ARA for access control rules. It proposes the skip index technique for skipping the

Manuscript received June 17, 2012; revised August 3, 2012. This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency (NIPA-2012-(H0301-12-3004).

D. An is with the Multimedia Contents Department, Shin Ansan University, Gyeonggi-do, Korea (e-mail: channy@ sau.ac.kr).

S. Park is with the Computer Science Engineering Department, Sogang University, Seoul, Korea (e-mail: spark@sogang.ac.kr).

unnecessary circulating process to ARA in XML documents. Therefore, the rewriting of user queries, which adjusts to the access control rule, is not required, and a secure query result is be generated, as access to restricted data is impossible. However, this method needs to have spate query processing channels for the user and the server because it handles query for previously used queries only once in a server. Hence, it has an overhead server and cannot be applied to a mobile environment that utilizes broadcasting method [12].

III. Energy-Efficient Query Processing for XML Data Streams

A. System Architecture

Fig. 2 shows the proposed system architecture. A server receives information from many data sources and gathers them into one XML document. It contains DTD and XML documents to be broadcast to the client, and it sends information to the client every time these documents are changed. The DTD document is usually unchanged, but the XML document constantly receives data from a data source and continuously updates changes.

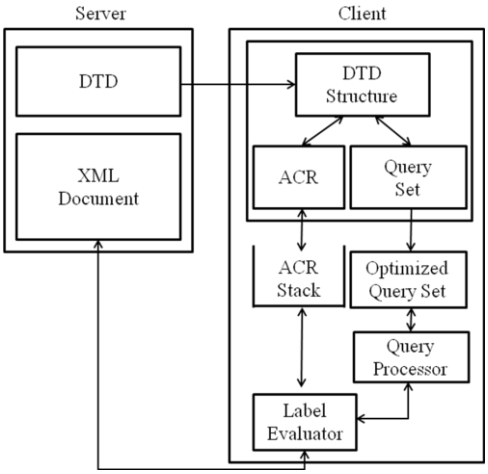

Fig. 2. Proposed system architecture.

The client stores DTD, which has been transformed from the server, and has a module to process queries entered by the access control rules and users, and to input XML files constantly. In this work, a client with limited resources, such as a portable terminal, is assumed. The Structural information of the XML document(DTD) to be received from the server should be known to process queries and access control processing. This work provides pre- and post-values to DTD and stores it in the form of a hash table. The client device has an access control rule(ACR) that suits users, and paths where the access control rules are loaded are various. Queries are entered by the users. The client outputs the result that the users want by storing a number of received queries from the users. The result is transformed into an optimized structure for fast query processing. The access control rule that satisfies the ACR stack is maintained to decide the status of the access to data being entered at the moment with the evaluation of access control rule. The label of input data evaluates for comparison with the input data while processing query and access control rules. By applying a label evaluator, faster evaluation and memory reduction effect for maintaining queries and access control rules can be achieved.

The query processor simultaneously evaluates queries and access control rules. It eventually evaluates the result that users want and decides whether to send the result to users or not after comparing it with the value of the access control stack.

B. Policy

In hierarchical data models [11], such as object-oriented data model and XML document model, the authorization that the administrator states is called explicit, and the authorization that a system derives based on explicit authorization is called implicit. It makes a "propagation policy" using the implicit authorization method to obtain the advantages of storing. The optimized propagation can vary for many different environments, but it usually uses the "most specific precedence policy." The "denial take precedence policy" is commonly used to solve "conflicts" that may occur through implicit authorization. "Open policy," which allows access, and "closed policy," which does not allow access are used as the "decision policy" on the node without explicit authorization because positive and negative authorizations are used in combination.

C. Preprocessing

1) Hash table

After the server transmits the DTD document, the client approaches the DTD accordingly and forms the DTD hash table by transferring the element information to the pre/post value to use it for evaluating queries and access control rules. The pre/post structure is used when the DTD is formed because an ancestor (or parent) and a descendent (or child) can be searched, and a path from the root can be comprehended through the pre/post value for arbitrary node when expressing XML documents.

Tag-name	Pre	Post	Parent
hospital	0	14	-
patients	1	13	hospital
patient	2	12	patients
room	3	0	patient
name	4	1	patient
doctor	5	2	patient
disease	6	4	patient
status	7	3	disease
status	8	11	patient
status_ECG	9	8	status
ECG	10	5	status_ECG
posture	11	6	status_ECG
respiratory_rate	12	7	status_ECG
blood_pressure	13	9	status
pulse	14	10	status

Fig. 3. PRE/POST hash table of the DTD tree (Fig. 1).

Fig. 3 shows the pre/post hash table for the DTD tree in Fig. 1. The pre(order) is a value obtained by approaching the DTD accordingly, whereas the post(order) is obtained from the following formula: POST = PRE + SIZE – LEVEL

Level denotes the level in the DTD tree, and SIZE is the node numbers under the tree of the arbitrary tree node.

As shown in Fig. 3, DTD has a tag name as the key and maintains the four information ({tag name, pre, post, parent}). A tag name is used in searching the DTD hash table. If the defined element repeatedly exists in DTD document, then several values (pre, post) can be found in the hash table.

2) Registration of query and access control rule

The registration of access control rule and user queries is processed using the pre/post metadata for the DTD tree during the compilation time. This section describes the process of changing the XPath expression into (pre, post) path expressing and registering it after changing it into the optimized (pre, post) path expression by analyzing the XPath expression.

The process of searching pre/post value is relevant to the XPath. The order of search goes up from the target node to the beginning node of the path expression. If there is a branch node of predicate during the process search, the value of the predicate should come first, and then the search for the previous node of the branch node should be continued. In order to find out if the searched value satisfies the XPath, verify if the newly searched nod is in ancestor relationship by comparing the (pre, post) value of the previously searched node and the (pre, post) value of the newly searched node.

As the ancestor for an arbitrary node is the only element in a tree structure, if a value has an ancestor relationship among the searched (pre, post) values, then the ancestor node is right. However, if a value having an ancestor relationship is not found, then the XPath expression does not express the DTD correctly. These inaccurate queries or access control rules are excluded from the evaluation.

D. Query Processing

In query processing, access control and query processing are performed at the same time using registered information from preprocessing.

1) Evaluation of XML data streams

If an XML document is transmitted from the server, the (pre, post) information must be known for evaluating the queries and access control rules. DTD hash table allows a suitable (pre, post) value to be searched quickly with the tag names of the input data. However, all the nodes appearing in the query path must be maintained to decide the suitable (pre, post) value because the exact position on the DTD for input data is unknown when many (pre, post) values are searched. The optimization of the query explained in preprocessing will be impossible if the (pre, post) value of input data is decided using this method because all the nodes appearing on query path must be maintained. In this work, information on the previously input data is maintained to identify the accurate information of the new data to be entered. When new data are entered, the (pre, post) value is evaluated using the input data information. This method is efficient in terms of using memory compared with maintaining all the nodes on the query path, which allows the optimization of a possible query by maintaining only the (pre, post) information on the data previously entered for evaluation of input information.

Input data information is divided into two cases. Let us call the information to be maintained for data evaluation current_data.

If a tag that notifies the start element is entered, then it is a descendent of the previous data or a root node. This allows information for the root not in the DTD hash table to be found easily. If the input data is not a root node, then search the (pre, post) value using the tag names of the data entered in the DTD hash table. If the value is relevant to the descendent of the data being maintained, then change the value of current_data to a new value.

If a tag that notifies the end element is entered, then return the value of current_data to the previous value of the data to be entered, that is, if element is the data to be entered, and if it is the same-leveled element with the same parent as the data inputted. The pre, post value relevant to the parent of current_data in the DTD hash table is searched, and the current data value is modified using the value of the parent relationship by comparing the results found and the pre, post value of current_data.

By evaluating current_data this way, a fast and accurate comparison is possible when evaluating query. The query can be optimized, making a lighter processing possible.

2) Evaluation of access control rule

Evaluating the query and access control rule is the same, but the access control stack should be maintained when evaluating the access control rule to decide the possibility to access every time the element that states access control rule is entered and the possibility to access the relevant access control rule entered in the stack. The access control stack has four values as shown in Fig. 4.

(+): Possible to access
(+?): Latently possible to access
(-): Impossible to access
(-?): Latently impossible to access

Among these four states, + and – states, which have the possibility to access relevant rule, are applied because they satisfy access control rules, including the predicate. The +? and -? state can satisfy all the other rules except predicate and it is a condition that possibility to access can be applied after figuring out if it satisfies the condition of predicate.

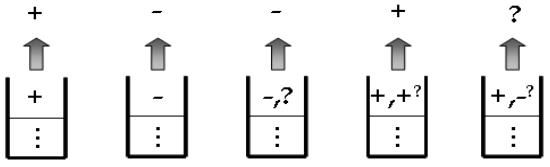

Fig. 4. Stack of access control.

The access control calculation method, similar to the access control used in this work, is defined as follows:

- This work cannot access the data when the stack is empty. Even though the query is satisfied, the result is not produced until the value + reaches the top of the stack.

- When the + value reaches the top of the stack, the relevant rule is entered if another access control rule is satisfied. In this case, the query is evaluated, and a result is sent if the query is satisfied because access to the input data is allowed.

- When the – value reaches the top of the stack, the used access control model cannot access the sub node of nodes that cannot be accessed. Therefore, all values entered later will change to –. In this case, query is not evaluated because access to the input data is not allowed. It is processed as if

data were not entered.

- When the +? value reaches the top of the stack, and the query is satisfied, then the result is temporarily stored because it can be latently accessed. If it later satisfies the predicate, then it changes to value + and sends the temporarily stored query results to users. If it does not satisfy the predicate, the result of the temporarily stored query is deleted.

- When the –? value reaches the top of the stack, then it may not be latently allowed to access or does not satisfy access control rule at all. The result is not stored because a result should not have to be produced even if it satisfies the query. If it satisfies the predicate later on, it is changed to–, and then all values are changed to–. If there is +? value, the query result, which is temporarily stored as the value changes to–, will be deleted because it becomes useless information that cannot be accessed by users.

3) Evaluation of query with added access control

Query evaluation is similar to access control evaluation. It is processed as follows:

- First, if the value of the access control stack is –, the query is not evaluated for input data. A node that cannot be accessed according to the access control rule evaluation is considered not entered because the users do not know if it is entered.

- Second, if all values of the access control stack are –?, state if they have the potential to become –; however, they may not satisfy the query. These two cases do not have to produce the result of the query, but the query is evaluated unlike in the case of –.

- Third, if the value of access control is +, then evaluate the query because it is possible to access the input data. If the data satisfying the query is entered, it is produced as the result.

- Fourth, if the value of the access control stack is in +? state then there's a potential possibility of being +. Since the result is unknown, evaluate queries and then temporarily store it when the data that satisfies query is inputted.

The detailed evaluation method will be used. Let us now look into the example of how query is evaluated with access control rules

IV. Evaluation

In this section, the performance of the XML access control method [13] is compared with the proposed method and related work. The performances of the proposed method before and after access control is applied are also compared and analyzed by dividing them into memory usage and processing time.

A. Experiment Environment

This experiment used the Microsoft Windows XP operating system with a Pentium IV 3.0 GHz processor and 1GB DDR2 memory. To input the document, the SigmodRecord.xml (467KB) file was used. The number of access control rules and queries was limited to the maximum of 10 due to the client environment. Table 1 gives the detailed information for SigmodRecord.xml.

TABLE I: SIGMODRECORD.XML

Size	467kb	# distinct tags	11
Depth	5	# text nodes	8,383
		# elements	10,022

B. Experiment Results and Analysis

1) Memory usage

The number of access control rules and queries was limited to 5. The rules that include the predicate had one predicate, but the target node did not have a predicate.

The proposed method passes the optimization process to reduce memory usage during the registration of query and rule as shown in Fig. 5 and 6. The rule, which does not include the predicate, maintains only one target node regardless of the depth of the rule, and the rule that includes the predicate maintains three nodes, namely, target node, branch node, and leaf node, of the predicate regardless of the depth of the rule. Hence, there is no additional cost of memory usage even if the depth of query or rule becomes longer.

As shown in the Fig. 5, our method used memory in processing did not exceed 255kb, but Bouganim [10]'s method used memory in processing was over 380kb.

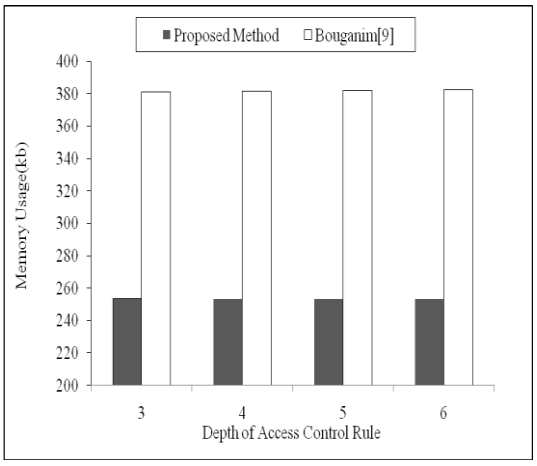

Fig. 5. Memory usage for the depth of access control rules.

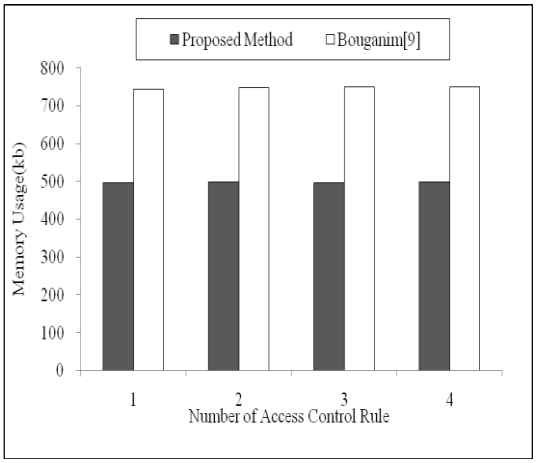

Fig. 6. Memory usage for the number of access control rules.

2) Processing time

The Fig. 7 shows the result of comparing the query processing time and the access control rule processing of the two methods by changing the number of access control rule, which increases the difference in performance.

In the client environment, where the number of access control rules that only one user can register, sharing the access control rule can be meaningless or may entail additional expenses.

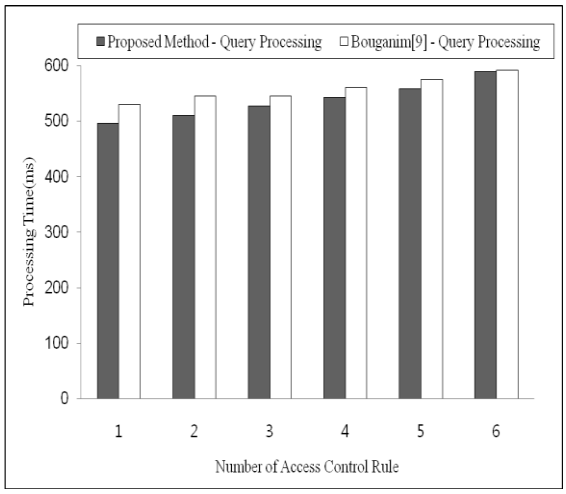

Fig. 7. Processing time for the number of access control rules.

3) Analysis of the overhead of applying access control

To process XML data streams in a client with limited memory, the difference in performance must be very small compared with the previous fast query processing even after applying the access control. Space should not be wasted only for the access control processing due to the limited resources, and receiving the result of the query slowly may be worse than previous security application. Fig. 8 shows the result of the performance comparisons before and after access control is applied. This work does not require additional cost for processing access control. The numbers of input query and access control rule are fixed data of 5 in the average memory usage test (Fig. 8) and processing time for file size (Fig. 9). The input XML data when estimating the query time for file size were measured by changing the file size of SigmodRecord.xml.

Fig. 8 shows the average memory usage when the access control rule of the proposed method is applied and when access control rule of the proposed method is not applied. The difference in memory usage between the minimum and maximum is 2kb.

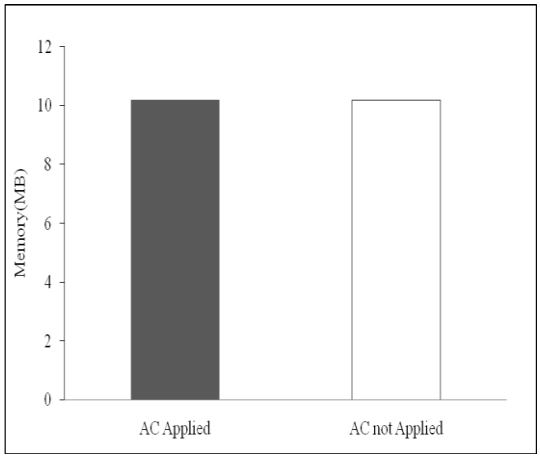

Fig. 8. Average memory usage.

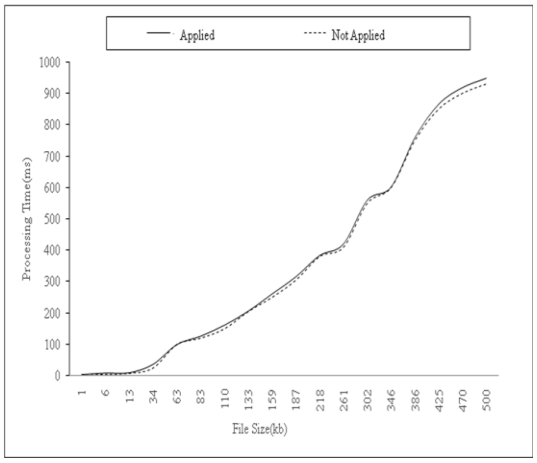

Fig. 9. Processing time for file size.

For the result of processing time for file size in Fig. 9, it does not overhead too much on access control application

V. Conclusions

We propose a method that can perform access control and energy-efficient query processing at the same time using schema information because it assumes an efficient XML data environment. Generally, two methods are used to express DTD in an efficient XML data environment. The one is expressed in the form of a tree, whereas the other one is expressed in the form of a direct acyclic graph (DAG). This is because there are cases involving several parents for a single element DTD due to the rule that an element should be expressed element only once. The DAG-based work can solve the wasting of storage space caused by the repeated node expression, which is a problem of the tree expression, and the wasting of evaluation time added to understand the parent node of self.

The characteristics of the proposed method are as follows: First, additional costs are reduced by finding a part (DTD hash table) for sharing the processing of query and property information during the access control process. Second, it optimizes the quality by determining the exact location in the DTD using input data evaluated and by reducing the node required for query evaluation. Third, the preprocessing cost is reduced through the optimization of XPath, improving the performance of the query process.

These characteristics can make the energy-efficient query processing in a client environment with limited resources more secure than the existing query processing

References

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, eXtensible Markup Language (XML) 1.0, World Wide Web Consortium (W3C), 2004.

[2] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon, XPath 2.0, World Wide Web Consortium (W3C), 2007. [Online]. Available: http://www.w3.org/TR/xpath20/

[3] W. Lindner and J. Meier, "Towards a secure data stream management system," *VLDB Workshop TEAA* 2005.

[4] E. Damiani, S. Vimercati, S. Paraboachk, and P.Samarati, "Design and implementation of access control processor for XML documents," Computer Network, pp. 59-75, 2000.

[5] E. Damiani, S. Vimercati, S. Paraboachk, and P.Samarati, "A fine-grained access control system for XML documents," *ACM Trans. Information and System Sec*., vol. 5, no. 2, pp. 169-202, May 2002.

[6] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti, "Specifying and enforcing access control policies for XML document sources," *WWW Journal*, vol. 3, no. 3, pp. 139-151, 2000.
[7] E. Bertino, S. Castano, and E. Ferrai, "Securing XML documents with Author-x," *IEEE Internet Computing*, May. June, pp. 21-31, 2001.
[8] J. Cai and C. K. Poon, "OrdPathX: supporting two dimensions of node insertion in XML data," in *Proceeding of DEXA*, 2009, pp. 332-339.
[9] W. Liang, A. Takahashi, and H. Yokota, "A low-storage-consumption XML labeling method for efficient structural information extraction," in *Proceeding of DEXA*, 2009, pp.7-22.
[10] L. Bouganim, F. D. Ngoc, and P. Pucheral, "Client-based access control management for XML documents," *VLDB*, pp. 84-95, 2004.
[11] B. Luo, D. W. Lee, W. C. Lee, and P. Liu, "Qfilter: Fine-grained run-time XML access control via NFA-based query rewriting," *CIKM'04*, pp. 543-552, 2004.
[12] D. C. An and S. Park, "Access control labeling scheme for efficient secure XML query processing," in *Proceeding of the Knowledge-Based Intelligent Information and Engineering Systems*, LNAI 5178, 2008, pp.346-353.
[13] F. Rabitti, E. Bertino, W. Kim, and D. Woelk, "A model of authorization for next-generation database systems," *ACM Transaction on Database Systems*, vol. 126, no. 1, PP. 88-131, March 1991.

Dongchan An is an Associate Professor of Department of Multimedia Contents at Shin Ansan University, Ansan City, Gyeonggi-do, Korea from 2002. He received the M.S. and Ph.D. degree in Computer Science and Engineering from Sogang University, Seoul, Korea in 1998 and 2011, respectively. His major research areas are role-based access control model, access control for distributed systems, access control for XML data, XML transaction management, and ubiquitous environment security.

Seog Park is a Professor of Computer Science and Engineering at Sogang University. He received the B.S. degree in Computer Science from Seoul National University in 1978, the M.S. and the Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 1980 and 1983, respectively. Since 1983, he has been working in the Department of Computer Science and Engineering, Sogang University. His major research areas are database security, real-time systems, data warehouse, digital library, multimedia database systems, role-based access control and web database. He is a member of the IEEE Computer Society, ACM and the Korean Institute of Information Scientists and Engineers (KIISE). Also, he has been a member of Database Systems for Advanced Application (DASFAA) steering committee since 1999.