

XML Security: Detecting and Preventing Disclosure in XML

Noha Nagy
Noha_Nagy_Mohy@hotmail.com

Mohamed E. El-Sharkawi
m.elsharkawi@fci-cu.edu.eg

Information Systems Department
Faculty of Computers and Information
Cairo University, Cairo, Egypt

Abstract

XML has become the prime standard for data exchange on the Web and a uniform data model for data integration. Since it solve the problem of different data representation the challenge to secure or making certain data invisible is as important as making the data available in an efficient manner. Disclosure problem still considered an open problem especially in XML..

To guarantee secure XML documents we need to develop appropriate protection techniques to control the inference process and protect sensitive data from reaching to unauthorized users.

In this paper we discuss the problem of protecting XML data at logical level specifically solve disclosure problem. The objective is to prevent unauthorized users to infer sensitive information through the data they authorized to access (result of previous queries), integrity constraints, and using inferences. In most existing access control approaches the XML elements specified by access policies are either accessible or inaccessible according to their sensitivity. However, in some cases, the original XML elements are sensitive and inaccessible, but after being processed in some appropriate ways, the results become insensitive and thus accessible [6]. We propose a security mechanism called Disclosure Prevention Algorithm (DPA) that enhances both the security and availability of data represented in XML format. The DPA is a security mechanism that works on detecting and preventing disclosure in order to prevent sensitive data from being inferred.

1 Introduction

XML is used in critical areas such as government, finance, healthcare and law. In this critical application the users must get complete and correct answers to their queries. Existing approaches focus on giving the user the information he permitted to have but don't make sure that the user can infer more information from his knowledge or from the application constraints and this information may be critical and can cause disasters.

There is an ongoing challenge in providing a balance between security requirements (confidentiality and integrity) and data availability [3]. Confidentiality is a

guarantee that data has not been disclosed to an unauthorized party. Threats to confidentiality include the direct release of sensitive data values, approximate disclosures, and leaks resulting from inferences and outside knowledge. One way to provide confidentiality is to enforce access controls, which permit or deny access to data items.

There are two types of attacks: direct and indirect attacks. In direct attacks the classified or sensitive information directly revealed to unauthorized individuals. But the indirect attacks sensitive information deduced by unauthorized individuals.

Access control models are used to detect direct attacks and prevent them from accessing data they are not authorized to access but the sensitive data remain unsafe from indirect attacks that occur through inferences. To overcome this problem we propose an algorithm which prevents both types. It extends access control (RBAC) which prevents direct attacks with the capabilities of inference engine to prevent indirect attacks.

So far disclosure problem in XML have not been studied very deeply. Most of work in XML security focused on developing access control mechanisms and authorizations such as annotation of the XML nodes to enforce access control policies [5] and solving conflict policies. Another direction which is how to compute the accessibility of XML nodes using compressed accessibility map (CAM), views [7, 9], and query rewriting approach.

In this paper, we propose a disclosure prevention algorithm DPA that detect and prevent inference channels to guarantee data security. Inference channel in a database is a means by which one can infer data classified at a high level using data classified at a low level. The disclosure problem is the problem of detecting and removing inference channels. The DPA checks whether a given query is safe and the user can access its result. A query is safe when its results can't be used by the user to infer sensitive information. The algorithm considers queries written in XPath language and uses functional dependencies between attributes in the XML document and the user's history to ensure query safety.

The paper is organized as follows: the reminder of the first section summarizes the assumptions and the problem definition, and it outlines the contribution. Section 2 presents an overview of our algorithm. Section 3 handling updates operations. Section 4 gives an overview on related work on this topic. Finally, Section 5 summarizes the paper and outlines future work.

1.1 General Assumptions and Problem Definition

We assume that some technical and some general requirements are met, because otherwise, the leak may lie outside of the system and cannot be detected by our algorithm. First, the integrity constraints we consider are functional dependency (FD) constraints. In practice functional dependencies are the most widely used integrity constraints. Moreover, other types of dependencies are not studied in XML context.

Second we assume that the users of the application system work in isolation. Third, as XPath is the core language for XML and is the base of other higher level languages (e.g. XQUERY), we concentrate on XPath. Fourth we do not consider intrusion detection.

The disclosure problem can be defined as: Given a user query Q , secret information $(S1, S2)$, and the FD constraints, our algorithm detects whether Q is safe or not. If Q is unsafe, the user can derive the secret information and then the query is rejected, otherwise Q is safe and the answer is presented to the user.

1.2 Motivating Example

Consider the XML document in Figure 1 which describes a company that has different departments. Each department is located in a specific location and each department has many employees. Employees have different salaries depending on their positions and their departments.

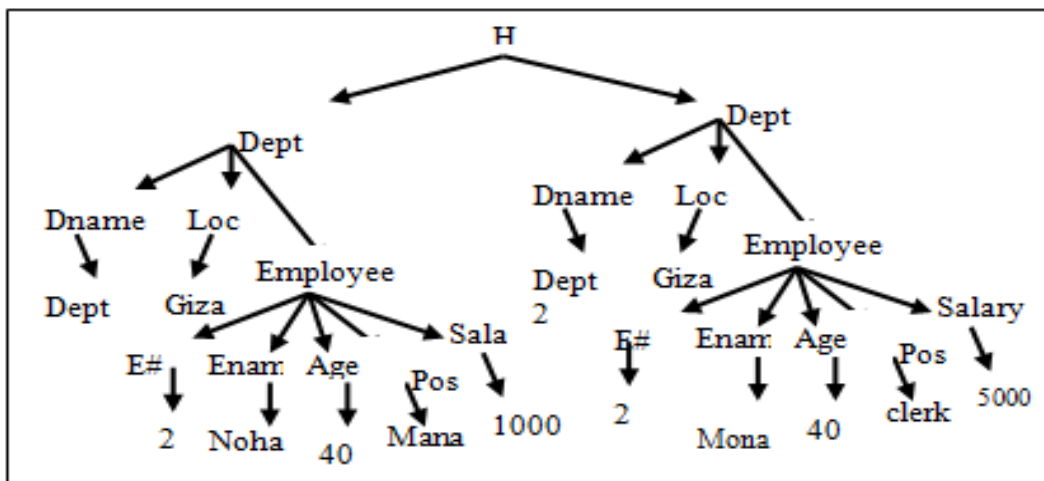


Figure 1: XML Tree of a Company System

The following constraints hold: (C1) each employee has a unique identification number, (C2) each department can be defined by its location, and (C3) employees in the same department and the same position have the same salary. Figure 2 describes these constraints.

C1: //Employee/E# → //HR/Employee/Enam
 C2: //Employee/Location → //HR/Dept/Dname
 C3: //Employee/Position, //Employee//Dept →
 //HR/Dept//Salary

Figure 2: Company System Constraints

The company authorizes an employee to access all her data except but not the salaries of other employees. But the employee name and employees' salaries can be accessed separately to increase the availability of data.

1.3 Basic Concepts and Definitions

Definition 1.1 (XML Document) we view an XML document as a tree. An interior node represents either an Element or an Attribute. A leaf node is of the string type. Some nodes are sensitive and their values should be hidden from unauthorized users.

In our model, XML constraints are playing an important role in disclosure and in finding

inference channels. The constraints we consider are Functional dependency constraints it is stored in constraints file.

Definition 1.2 (Constraints File) it is a text file that stores all the FD constraints of an XML document.

Definition 1.3 (Functional dependency) A functional dependency is represented as $p/p1 \rightarrow p/p2$, where p , $p1$, and $p2$ are finite non empty subsets of paths conforming to the document. It means that for any two subtrees $t1$ and $t2$ matching on path $p/p1$, if they have equal values in their $p1$ paths then both of them have equal subtrees that match $p/p2$. The constraints are saved in constraints file.

Definition 1.4 (Secret information) data to hide from unauthorized users is represented by regulating query. The sensitive node is marked with symbol "*" as in Figure 3. It means that the salary of an employee is sensitive. It is also possible that a node can be sensitive only if accessed with some other nodes. For instance, it may be possible for users to access salaries as values but not in combination with names of employees who earn these salaries. In such a case both nodes are considered sensitive (see Figure 3)

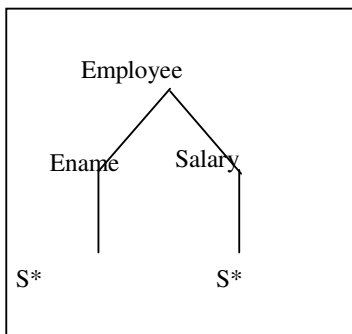


Figure 3: Regulating Query

1.4 The Disclosure Problem

Definition 1.5 (Disclosure problem) a disclosure is a result from combining metadata (e.g., database constraints) with non sensitive data to obtain sensitive information [1]. The disclosure problem also defined as the problem of detecting and removing inference channels. [20]

Disclosure problem require detection then prevention. The goal of inference detection is to determine if a user can access or get unauthorized

sensitive data from authorized data. Most of the existing approaches to disclosure detection focus on analyzing functional dependencies. The prevention requires monitoring the user's requests history to know if he can detect a sensitive data or not.

Techniques to detect inference channels can be organized into two categories. The first category includes techniques that detect inference channels during database design time. Inference channels are removed by modifying the database design or by increasing the classification levels of some of the data items [9,13]. This leads to over classification of data and may increase unavailability. The second category seeks to eliminate inference channels during query processing time. The data is more available but it increases the query processing time. The prevention in this category is to refuse or modify user query. Our algorithm is mixed it takes the advantage of both categories. It builds dictionaries such as inference channels and suspected users in design phase and at query processing it uses these dictionaries to evaluate user's queries to detect and prevent disclosure. This increases data availability and accelerates the process of query evaluation.

2 The Disclosure Prevention Algorithm DPA

The DPA is a mechanism that used to control the disclosure "blocking inference channels" in XML by integrating RBAC model with an inference engine. In RBAC, permissions are associated with role permissions as in Table 1 that define privileges (nodes) granted to each role.. The nodes defined as a path from the root to the node. Then the users are assigned as members of appropriate roles as in Table 2. The smallest protection granularity of our algorithm is the node.

DPA works in two phases the design phase and the query processing phase.

At design phase: the disclosure inference engine builds some dictionaries that contain the identified inference channels ICs, suspected users, and histories of suspected users.

This stage is repeated in the following conditions: adding a new user or deleting an existing user, revoking or grating new privileges to existing users, adding or deleting constraints.

At query processing phase: the evaluator is evaluating a user's query for direct disclosure using the RBAC(user permissions and role permissions)

then For only suspected users it makes a second evaluation for disclosure using their histories and the dictionaries built in design phase.

The user's requests are written in XPath which is a language deal with XML documents. It uses paths to navigate in XML documents. Path is based on the UNIX directory notation. XPath query language operates over XML documents and produces XML documents as a result.

2.1 Disclosure Inference Engine DIE

DIE prepares the metadata required by the query evaluator to check for query safety. This metadata is constructed at design time. It is also responsible for generating the inference channels (IC). To do this, two graphs are used. The node dependency graph is a directed graph to represent functional dependencies. The other graph is an undirected graph called, the role dependency graph. It is used to find the paths of nodes that start with sensitive node "or any node that that has a FD relation with this sensitive node" and end with the other sensitive node. Then get the inference channels in terms of roles by acquiring the roles labeled on the edges. The DIE finds the suspected users who can access one or more ICs using the user's permissions. Then it creates a history file for each suspected user to store his past and present queries to track his behavior and detect disclosure.

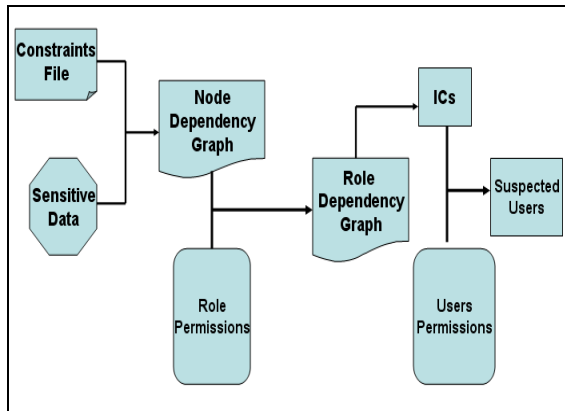


Figure 4: Inference Engine

Definition 2.1 (Node Dependency Graph) it is used to describe the FD relationship between the XML nodes. The nodes of the graph are the nodes of the FD in the constraints and there is an edge between the nodes if there is a FD relation between these nodes. An example is shown in Figure 6a.

Definition 2.2 (Role Dependency Graph) is built upon the node dependency graph and the role

permission table. It is used to define the inference channels in terms of roles. It is built as follows:

1. Draw a node for each node in the node dependency graph.
2. From the role permission table, find the roles that access two or more nodes present in the node dependency graph.
3. Connect these nodes and label the edge with role name.

An example of the role dependency graph is shown in Figure 6b.

Definition 2.3 (Inference channels) an inference channel is a method by which one can infer data classified at a high level from data classified at a lower level. [12]

Role	Nodes
R1	//Employee/E#, //Employee/Ename
R2	//Dept/Location, //Dept/Dname
R4	//Employee/Position, //Employee/salary, //Dept/Dname
R5	//Employee/Ename, //Employee/Position
R6	//Dept/Dname, //Employee/Ename
R7	//Employee/Ename, //Dept/Location
R8	//Employee/ Age, //Employee/E#

Table 1: Role Permissions

User	Role Permission
U1	R3
U2	R1
U3	R4
U4	R2
U8	R7,R2,R5,R4,R1
U11	R3,R4,R5,R6
U88	R5,R4,R2,R6,R7

Table 2: User's Permissions

The inference channels take the form of pairs of nodes. The valid inference channels in our example:

Ch1 {(Ename,Dept),(Ename,Position),(Dept,Salary, Position)},

Ch2 {(Ename,Location),(Location,Dept),(Ename,Position),(Dept,Salary,Position)}

Inference channels can be used to group roles. For example, the above two channels may group roles as follows: Ch1:{R6,R5,R4}, Ch2:{R7,R2,R5,R4}. This means if a user has access rights to R6,R5,R4 then he can disclose sensitive information.

After DIE discovers the inference channels, it uses user's permissions (Table 2) to identify the users who have opportunity to access at least one of the inference channels (suspected users). In our

example the suspected users are U8, U11, and U88. After that the DIE creates history for each suspected user to track her behavior.

Definition 2.4 (Users History) for each suspected user a profile is created. It contains the answers of submitted queries by the user and the Role/s the user has played when submitting the query. Figure 7 gives an example.

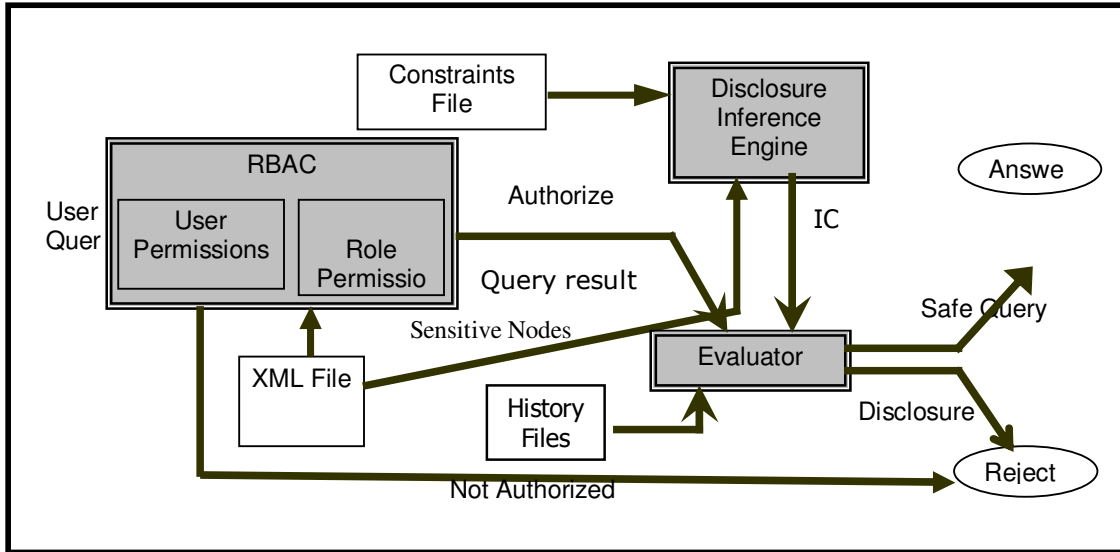


Figure 5: Architecture of Disclosure Prevention Model

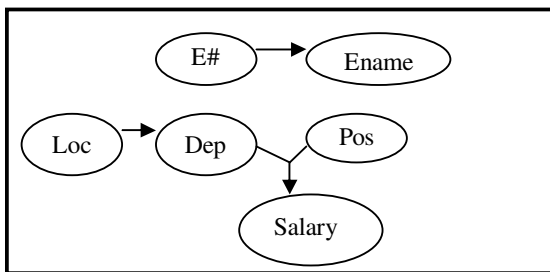
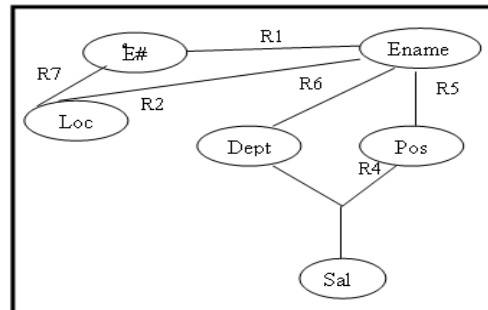


Figure 6: a. Node Dependency Graph



b. Role Dependency Graph

2.2 Evaluator

It is a module executed at query processing time to check whether processing this query will result in disclosure. If the query issuer is not a suspected user and authorized to access all the nodes used in

the query then the query is safe and the answer is presented to the user. Otherwise, if the user is one of the suspected users the evaluator checks the data stored in her history file (past answers of user's queries) combined with the answer of the current query against the ICs. If the current query answer

will let this user access all the pairs of an inference channel then a disclosure is detected and the query is rejected. Otherwise, it presents the answer to the user and updates his history. The evaluator is sketched in Figure 8.

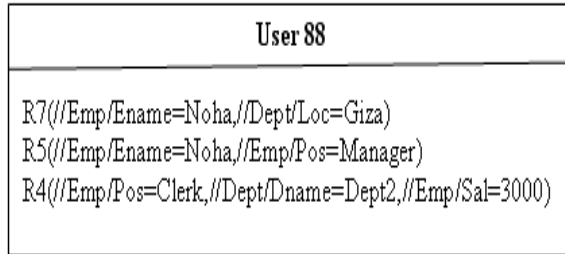


Figure 7: History

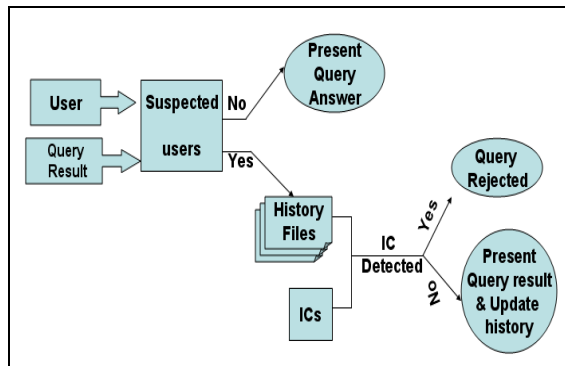


Figure 8: Evaluator

nodes deleted then we need to delete the valid inference channels that contain the deleted node from the valid inference channels. Moreover, the process of finding the suspected users is repeated. If one of the suspected users can access only the deleted channel(s) then she is no longer suspected and her history file is deleted.

Insert operation: is responsible for adding new nodes to the XML tree. If the new node(s) added to the constraints file has a FD relationship with other nodes, the process of determining inference channels is repeated.

Update operation: changes only the data not the structure of the document. If the updated node is one of the valid inference channels nodes then we need to reflect the new value in all the history files of the suspected users that contain these updated nodes. See Figure 9.

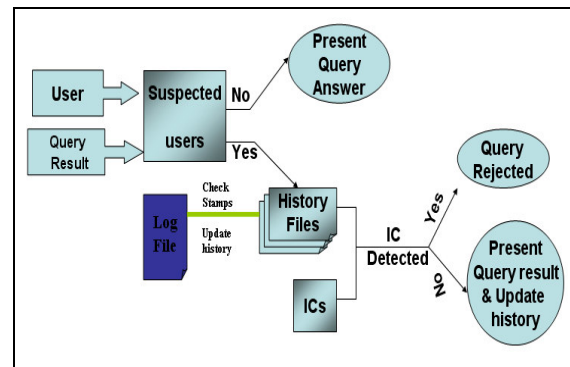


Figure 9: DPA with Update Operations

3 Handling Update Operations

We need to study the effect of changing the XML document on the disclosure. As a matter of fact XML is different in structure from relational databases. Updates can be on the structure of the XML tree or on the values of the data nodes. In this paper we consider both types of updates. We focus on rename; delete, insert, and update (modify) operations. Figure 9 show how DPA deal with update operations.

Rename operation: changes the label of an XML node. It doesn't really change the tree structure of the XML document. If the update affects one of the valid inference channels nodes the new name of the node has to replace the old name in the constraints file and the history files that contain the renamed node.

Delete operation: is responsible for removing one of the XML nodes. Hence it changes the structure of the XML tree. If one of the inference channels

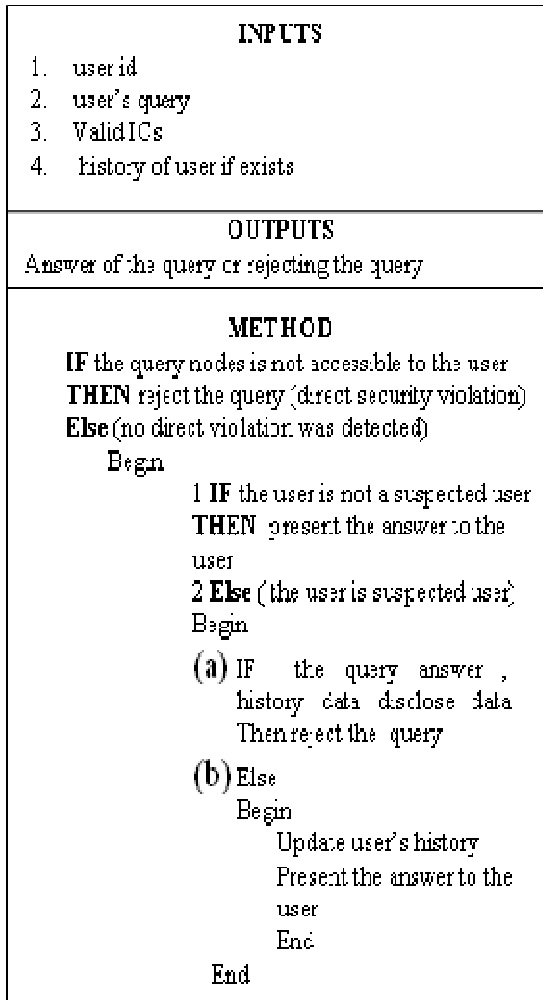


Figure 10: The DPA Algorithm

We need to check if the updates made affect the history file's data. We use two known techniques which are Log file and stamping. The update operations saved in an update log file. The update log file contains the node label, the new value, and the time of this operation. We use stamping to make the process of tracking updates easier. We stamp the update log file itself and its contents. Also, the history files have a stamp. We need to update the history files of the new updated data. This happened whenever a suspected user issue a query the algorithm checks the stamp time of the history file HFS and the stamp time of the update log file ULS. If the ULS greater than HFS then the history file need to reflect the new changes of the update log file. And if disclosure detected as a result of these updates the system should give the administrator an alarm that disclosure detected.

4 Related Works

Several systems have been proposed to support XML access control, Our access control mechanism is related to many of the previously available access control and authorization models. There are a lot of researches efforts have considered access control models for XML data [4, 5] but these models suffer from various limitations. The most important one is they don't consider the indirect attacks. DMon [1], are the most related to our model. DMon detect and prevent disclosure that may occur in relational databases. The idea is to create a history file for each user and store in it past and present queries. The DiE is the component that responsible for generating all information that can be disclosed based on a user's previous queries results(stored in history file), the current query result and database constraints. Another approach is D2Mon [12,14] which extend the capabilities of DMon to work with dynamic databases. Some work was on providing access control to semantically related XML documents and provide a general control mechanism for access made to a collection of related documents rather than individual document. Other research direction try to find a partial document to be published on the web and can't disclose sensitive data others try to use the query containment approach to easily evaluate the users queries .

5 Conclusions and Future Work

In this paper, we have presented an approach that integrate the role based access control models with the power of inference engine to prevent direct and indirect attacks to sensitive information. We benefit from using the two techniques for detecting disclosure by working in design time and query processing time. Calculating ICs, finding suspected users, and creating histories for them in the design phase make the disclosure detection process easier and faster in query processing time. Only the queries issued by suspected users need to be evaluated against disclosure but other queries issued by other users will not be evaluated as we sure that they are safe and the users cannot detect any sensitive information. We keep the histories for suspected users only which save space. Another issue we don't store all issued queries and its results only the queries that its result contains pair of the ICs pairs. This keeps the history smaller and easier in detecting disclosure.

We also extend the capabilities of the DPA by handling the update operations on XML documents

and study its effect on disclosure problem to guarantee confidentiality even if updates allowed.

For the future work we want to extend our model to handle multiple users' interactions. We also want to study the situation where multiple XML document considered. Another matter let our algorithm accept queries written by XQuery.

References

- [1] A. Brodsky, C. Farkas, and S. Jajodia, "Secure databases: Constraints, inference channels, and monitoring disclosure," *IEEE Trans. Knowledge and Data Eng.*, 900-919, November, 2000
- [2] E. Bertino and E. Ferrari. "Secure and selective dissemination of XML documents" *ACM Trans. Information System Security* 5(3):290–331, 2002.
- [3] E. Bertino, and Ra. Sandhu,, "Database security concepts, approaches and challenges," *IEEE Transactions on Dependable and Secure Computing* , 2-19, January 2005
- [4] E. Damiani, S. D. Capitani, S. Paraboschi, and P.Samarati "Securing XML Documents," In *International Conference on Extending Database Technology (EDBT)*, 121-135, 2000.
- [5] E. Damiani, S. di Vimercati, S. Paraboschi, and P. Samarati. "A fine-grained access control system for XML documents", *ACM Transactions on Information and System Security (TISSEC)*, 5(2), 196 – 202, 2002.
- [6] D. X. Liu ,"CSchema: a downgrading policy language for XML access control," *Journal of Computer Science and Technology*, 22(1): 44-53 (2007).
- [7] H. Kang, H. Sung, and C. Moon "Deferred Incremental Refresh of XML Materialized Views", *Proceedings of the 14th Australasian Database Conference*, 217-226, 2003
- [8] H. Zhang, N. Zhang, K. Salem, D. Zhuo "Compact Access Control Labeling for Efficient Secure XML Query Evaluation," *Proceedings of the 21st International Conference on Data Engineering, ICDE*, 1275, 2005
- [9] J. Wang, S. L. Osborn "Role Based Approach to Access Control for XML Databases," *Proceedings of the ninth ACM symposium on Access control models and technologies*, (20-22) 2004
- [10] G. Lee, K. Y. Whang, W. S. Han, and L. Y. Song, "The Dynamic Predicate: Integrating Access Control with Query Processing in XML Databases," *The VLDB Journal*, 16(3), 371-387, 2007
- [11] S. Cho, S. Amer-Yahia, L. Lakshmanan, and D. Srivastava. "Optimizing the secure evaluation of twig queries" *Proceedings of 28th International Conference on Very Large Databases (VLDB)*, 490-501, 2002.
- [12] S. Jajodia, C. Meadows, "The Inference Problem in Multilevel Secure Databases Management Systems", *An Integrated collection of Essays: IEEE Computer Society Press*, 570–584, 1995
- [13] T. H. Hinke, H. S. Delugach, and A. Chandrasekhar, "A Fast Algorithm for Detecting Second Paths in Database Inference Analysis," *Jour. Of Computer Security*, 3(2/3): 147-168, 1995
- [14] Toland, C, Farkas, C. M. Eastman "Dynamic Disclosure Monitor (*D²Mon*): An Improved Query Processing Solution," *Secure Data Management*, 124-142, 2005
- [15] Fan, C. Y. Chan, M. Garofalakis "Secure XML Querying with Security Views," *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (587 - 598), 2004
- [16] X.Yang, C.Li "Secure XML Publishing without Information Leakage in the Presence of Data Inference," *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, (96-107), 2004

Noha Nagy Mohy is a teacher and research assistant in the Department of Information Systems, the Faculty of Computers and Information, Cairo University, Egypt. Her research interests are in XML databases, specifically security and query processing techniques. This work is part of her Master's thesis where she studies the relationship between access control and query processing in XML.

Mohamed E. El-Sharkawi is an associate professor in the Department of Information Systems, the Faculty of Computers and Information, Cairo University, Egypt. He got his B.Sc. in Computer Engineering from Al-Azhar Univ., Egypt and his M.Sc. an Ph.D. from Kyushu Univ., Japan. His research interests include XML databases, temporal databases, and data mining.