

XGuard: A System for Publishing XML Documents without Information Leakage in the Presence of Data Inference

Xiaochun Yang
Northeastern Univ., China
yangxc@mail.neu.edu.cn

Chen Li
UC Irvine, USA
chenli@ics.uci.edu

Ge Yu
Northeastern Univ., China
yuge@mail.neu.edu.cn

Lei Shi
Northeastern Univ., China
neushi@126.com

1. Introduction

With the fast development of the Internet, there is an increasing amount of data published on the Web. Often the owner of a database needs to publish the data to others such as public Web users or collaborative peers [1, 3, 4]. For instance, consider a hospital that shares its patient information with a collaborative department [6]. The information is stored in an XML document, part of which is shown in Fig. 1. Suppose the hospital wants to hide the disease value “leukemia” of patient Alice. It could hide the information by just removing this node from the XML tree. However, it could be well known that “patients in the same ward have the same disease.” Since there are other patients Betty and Cathy staying in the same ward “W305” as Alice, the collaborator could use this common knowledge to infer the disease of Alice using that of Betty and Cathy. In this case, the data owner has to hide additional information (e.g., the ward of Alice) to hide the sensitive information.

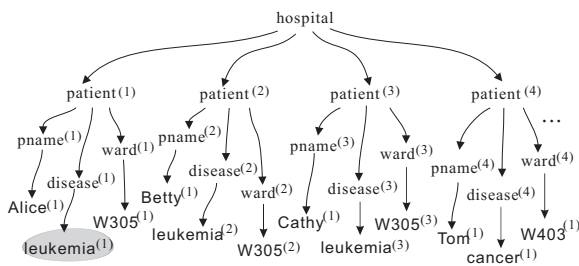


Figure 1. An XML document about patients.

In general, in data publishing, if the data is published carelessly, public users could use common knowledge to infer more information from the published data, causing leakage of sensitive information. To address related research challenges, we develop a system called XGuard, which can help data owners publish a partial XML document without leaking sensitive information, even if public users can do

inference. Specifically, the system has the following functionalities.

- It allows the data owner to define sensitive information and specify common knowledge as XML constraints.
- Given a partial document, the system can validate if the document can cause information leakage due to common knowledge and how much data can be leaked.
- The system can help the data owner interactively analyze the data inference and produce a secure valid partial document using the algorithms presented in [6].

Fig. 2 shows the architecture of the XGuard system. A data owner provides an original XML document to the system, and defines sensitive data by either using XPath queries or marking the nodes on the XML tree. Such definitions are stored in the module called *Sensitive-Data Definitions*. In addition, the owner specifies common knowledge that is known to public users. Such knowledge is defined as XML constraints, stored in the *XML Constraints* module. We consider three types of common constraints: child constraints, descendant constraints, and functional dependencies. The owner can provide a partial document by removing the sensitive nodes from the original nodes. Then the *Validator* module can check if the current partial document can leak any sensitive information when public users can use the common knowledge to do inference. In addition, the module can also compute a valid partial document by providing a data structure called AND/OR graph [5]. See [6] for details of our algorithms for constructing an AND/OR graph. Such a graph can help the data owner analyze the data inference and decide what additional nodes need to be removed.

2. Demonstration

In this demonstration, we will show the main functionalities of the system using a visualization toolkit. Fig. 3 shows a screen shot of the interface (implemented in Java based on the Shunsaku XMLDB system [2]). It contains

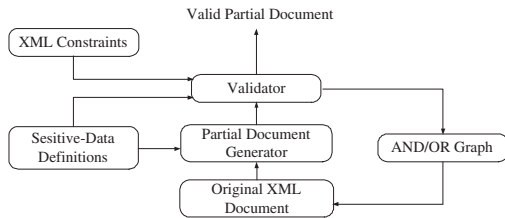


Figure 2. Architecture of XGuard.

five main frames (from left to right) to define a regulating query that specifies sensitive data, display an original document, define XML constraints, display an AND/OR graph, and display a valid document. We will use the hospital example above to illustrate these functionalities.

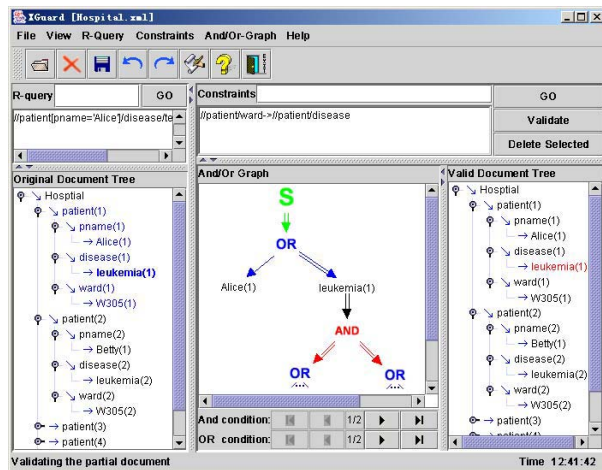


Figure 3. XGuard interface.

Defining Sensitive Data and Constraints: A data owner first specifies an XML document to be loaded into the system in order to compute a valid partial document. The system displays the XML tree of the given document. The owner defines sensitive data in two possible ways. (1) *Marking Nodes:* The owner defines sensitive data by marking nodes in the tree. For example, the owner can define the node “leukemia” (of Alice) as sensitive data by right clicking on the node and choosing the “Mark” option. If a nonleaf node is defined as sensitive, then all its subtree is also considered to be sensitive. (2) *XPath Queries:* Such a query is defined as an XQuery, the answer to which is considered to be sensitive. For instance, query `//patient[pname = ‘Alice’]/disease/text()` defines that the disease value of patient Alice is sensitive. Such queries are especially needed for defining sensitive

data with complicated conditions. The owner also specifies common knowledge using XML constraints.

Validating a Partial Document: Given a partial document, the system can validate whether the partial document can cause information leakage. Since public users can apply constraints in arbitrary sequences to infer different documents, the system needs to consider the worst case by computing a maximal, unique inferred document [6], and matching the regulating queries with the inferred document. If such a match does not exist, then the partial document is safe. Otherwise, the leaked sensitive data will be shown to the data owner.

Calculating a Valid Partial Document: In addition to supporting validation of a partial document, XGuard also helps the data owner compute a valid partial document interactively using the interface. There are many possible partial documents that do not cause information leakage (a trivial one is the empty document). Our goal is to publish as much data as possible without leaking sensitive information. One key challenge is deciding how to break mappings from regulating queries to the inferred document that cause the leakage, and how to chase back the inference steps. In [6] we presented an algorithm for finding a valid partial document by constructing an AND/OR graph. For instance, the AND/OR graph in the figure shows that, in order to hide the disease value of Alice, we can either hide the *Alice*⁽¹⁾ node or the *leukemia*⁽¹⁾ node. This relationship is represented as an OR connector. To hide the latter, we may need to hide more information about Betty and Cathy who live in the same ward as Alice, and this relationship will be represented as an AND connector. In the demo, we will show the procedure of backtracking multiple inference steps using constraints, and use a greedy search algorithm to provide a suggestion for calculating a valid partial document with as many nodes as possible.

References

- [1] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. A Fine-Grained Access Control System for XML Documents. *ACM Transaction on Information and System Security*, 5(2):169–202, 2001.
- [2] Fujitsu. Interstage Shunsaku Data Manager. <http://interstage.fujitsu.com/jp/v6/shunsaku>.
- [3] C. Li, J. Li, and Q. Zhong. RACCOON: A Peer-Based System for Data Integration and Sharing. In *ICDE*, 2004.
- [4] G. Miklau and D. Suciu. A Formal Analysis of Information Disclosure in Data Exchange. In *SIGMOD*, 2004.
- [5] N. J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, 1994.
- [6] X. Yang and C. Li. Secure XML Publishing without Information Leakage in the Presence of Data Inference. In *VLDB*, 2004.