STRUCTURE ENCRYPTION IN XML*

Taflan İ. Gündem and Mustafa F. Çelikel

Computer Engineering Dept., Boğaziçi University, 34342 Bebek, İstanbul, Turkey

Abstract

In the literature in XML encryption, just the text in the document is encrypted using an encryption algorithm. In this paper we introduce the concept and technique of structure encryption for XML. Structure encryption is basically encrypting the structure of XML (i.e. how elements are nested in XML). Fundamentally, it is hiding the structure of an XML document in addition to hiding the element values. Structure encryption may be used for two different purposes. One purpose is for increasing the difficulty of the decryption of very sensitive documents. One needs to decrypt the structure also. The other purpose of structure encryption is facilitating light encryption (i.e. just encrypting the structure and not the whole text) of not so sensitive documents.

Keywords: database modeling, XML, relational databases, information hiding

1. INTRODUCTION

XML is becoming a standard in semi-structured document representation and data exchange. For the security of sensitive XML documents encryption is a commonly utilized method. Thus we see some work done on encrypting XML documents [1] in the literature. However the work done in the literature on XML encryption is based on encrypting just the

* This work is supported by Boğaziçi University Research Fund under grant number 04A108.

text in the document using an encryption algorithm. To make decryption more difficult we propose to encrypt the structure of the nested elements in an XML document as well as the elements themselves so that even if the elements are decrypted, the placement of the elements with respect to each other will not be known. To be able to obtain the original XML document one needs to further decrypt the structure with an additional key to find the correct placement of the elements within the document. Another use of structure encryption is for obtaining a light encryption of a not so very sensitive document. We can encrypt just the structure and leave the element values unencrypted. In this paper we give the description of an algorithm to encrypt the structure of XML documents.

The basic motivation behind the methodology that we present is that structure is so important that it deserves to be further hidden and encrypted with an additional key. The basic idea behind the methodology that we present is that two keys are used for encrypting an XML document: one for encrypting the elements of the document and the other for encrypting the structure. We think this is the best way of making use of an additional key (i.e. hiding the most important part of a data model, the structure). If two keys were used for encrypting two different parts of a document (let us say one key for a half and the other key is for the other half) then with one key it is possible to obtain a part (half) of the document. But in our methodology, one does not get any useful information with just one key. Just elements without any structure (i.e. ordering and hierarchical relationships among elements) are useless just as only structure without any elements is.

Usually there is a hierarchical relationship among the elements of an XML document. This relationship may be shown by a DTD. To store an XML document one of the basic approaches is using relational databases. In converting an XML document into a relational database one of the techniques that may be used is the relational DTD [2] approach. In this paper we assume that the XML document is stored using relational DDT approach. If the

original XML document is not stored using the relational DDT approach, we do it. In relational DDT approach the structure of the XML document as well as the values of the elements are stored in relations.

The organization of this paper is as follows. In the next section the encryption algorithm and the ideas behind it are given. In section three we illustrate the encryption algorithm by means of an example. The last section is for conclusions.

2. STRUCTURE ENCRYPTION ALGORITHM

The general outline of the algorithm is as follows:

1. Store the XML document in relations using shared inlining technique, if it is not already stored so.. This is a standard technique used in representing XML documents in relations. A methodology for converting XML documents to relations using shared inlining method is explained in [3] and may also be found in [4]. In section 3.5 an XML document and the corresponding relational schema obtained using shared inlining method are given. The explanation of the details of shared inlining method is beyond the scope of this paper and is not necessary for understanding the concepts explained in this paper. What is important is to notice that in the relations or tables, obtained as a result of applying the shared inlining method, each tuple or row represents an element, say x, in the XML document and a pointer (identifier) to the parent element of the element x, as can be seen from the example in section 3.5.

2. After obtaining the relational schema, we encrypt all the structural components of the tuples in the relational representation with an algorithm that we call the *field-key algorithm* and that is explained in section 2.1 in detail.

2.1. THE FIELD-KEY ALGORITHM

In the whole algorithm, this may be the most crucial part. In this part we explain the encryption of the fields that hold the structural information about the XML document. After the application of the Relational DTD approach, all the resulting tables will have ParentID and ID fields. The ID field holds the record's ID and ParentID field holds the ID of this record's parent. Using the information in the ParentID and ID fields, general structure of an XML document can be configured (In a tree, if we know the parent of each node, we can construct the tree). By encrypting these two fields, we can encrypt the structure of an XML document that is stored in the relations. In the following we will explain how we do the encryption.

The field-key algorithm uses the CBC mode of DES [5]. In the field-key algorithm, we encrypt the ParentID and ID field values in such a way that the encryption of two ParentID (or ID) field values that are exactly the same are different. The reason for this is to make the construction of the document structure from the encrypted ParentID and ID field values impossible. This is explained in detail in the following.

The XML document can be reconstructed by matching the proper ParentID and ID field values in the tables. Since our objective is not allowing an unauthorized user construct the XML document, we should make this match difficult. We illustrate the importance of this match in example 1.

ParentID	ID	Dept_id
1	2	"dept1"

DEPT.

ParentID	ID	Student_id	Name
2	3	"123"	"St1"
2	4	"124"	"St2"

STUDENT

Figure 1. Dept. and Student relations

Example 1: Let us assume we have the Student table and the Dept. table shown in figure 1. In order to construct the element graph, ParentID fields of the Student table is equi-joined with the ID field of the Dept. table (i.e. For each row, x, in Student table if the ParentID field of row x is equal to the ID field of a row, y, in Dept. table, x and y are concatenated.). In this example, we see that the parent of "St1" and "St2" is "dept1" because the ParentID fields of students "St1" and "St2" match the ID field of "dept1" in Dept. relation. By this match we can construct the document. Thus we should make this match difficult and not obvious.

When two identical plain texts are encrypted with the same algorithm, the resulting cipher texts are also the same. For example if we encrypt the ID field value '2' in the Dept table and the ParentID field value '2' in the Student table using the same algorithm, we get the same encryption. Consequently by concatenating rows with the same encryption for proper fields from different tables, it may be possible to get the structure and reconstruct the document without decrypting it. Thus we need an encryption that will make this match difficult.

The encryption we have chosen for this purpose is the CBC mode of DES. In the CBC mode of DES, the identical cipher text blocks result when the same plain text is enciphered under the same key and IV (Initial Vector). However, changing the key results in different cipher text. Thus we can easily get different cipher texts in CBC mode when identical plain texts are encrypted.

In our algorithm we use different keys for every plain text in the structure fields. The important thing here is how we choose or change the key for every plain text. One way of changing the key is explained in Method A.

Method A: Since we are dealing with tables, we can choose one of the field values of a table to be the key for the encryption / decryption processes. In the relational schemas of the stored XML document, we have the following obligatory fields: a ParentID field, an ID field

and a field for the value part. In our algorithm, we choose the value field of each tuple (row) as the key for the encryption / decryption processes of the CBC mode of DES.

In Dept. relation in Figure 1, we encrypt ParentID and ID fields using “dept1” as their cryptographic key. In Student relation in Figure 1, we use “St1” as the cryptographic key for ParentID and ID fields. As a result the cipher text for the value “2” in a row in relation Dept. is different from that in a row in relation Student. When those fields are compared in the encrypted document, they will not match and it would not be possible to construct the document. In the example in section 3, we use this method to encrypt ParentID and ID fields.

Method B: The other method that we propose for encrypting the structure fields (i.e. ParentID and ID fields) is as follows. Include an extra column(field) called Strc-ID in each table. The value corresponding to Strc-ID in a tuple is used as the cryptographic key to encrypt the structure fields, ParentID and ID fields, in that tuple. The Strc-ID field values of the whole database are encrypted using a different key than the one used to encrypt the value fields. That is we have two keys. One key is used to encrypt all the fields except Strc-ID and the structure fields, ParentID and ID. The other key is used to encrypt all the Strc-ID fields in the database. Strc-ID field value in each row is used to encrypt the structure fields (ParentID and ID).

The encryption of the fields for the value parts is achieved also by using DES algorithm. We encrypt a value field the using a common key for all the value fields. The decryption process is the reverse of the encryption. That is we get the relational tables using proper the keys.

3. AN EXAMPLE ON STRUCTURE ENCRYPTION OF XML

In the following, we apply the whole algorithm to the XML document, dept.xml shown in Figure 2. In this example we use the Method A for encrypting the structure fields. The structure graph (DDT) of dept.xml is shown in Figure 3.

```
<?xml version="1.0"?>
<!DOCTYPE Dept SYSTEM "Dept.dtd">
<Dept dept_id="dept1">
<Student student_id="123">
<Name>St1</Name><Enrollment> CS10</Enrollment><Enrollment>CS20</Enrollment></Student>
<Student student_id="124"><Name>St2</Name></Student>
</Dept>
```

Figure 2. The XML document dept.xml.

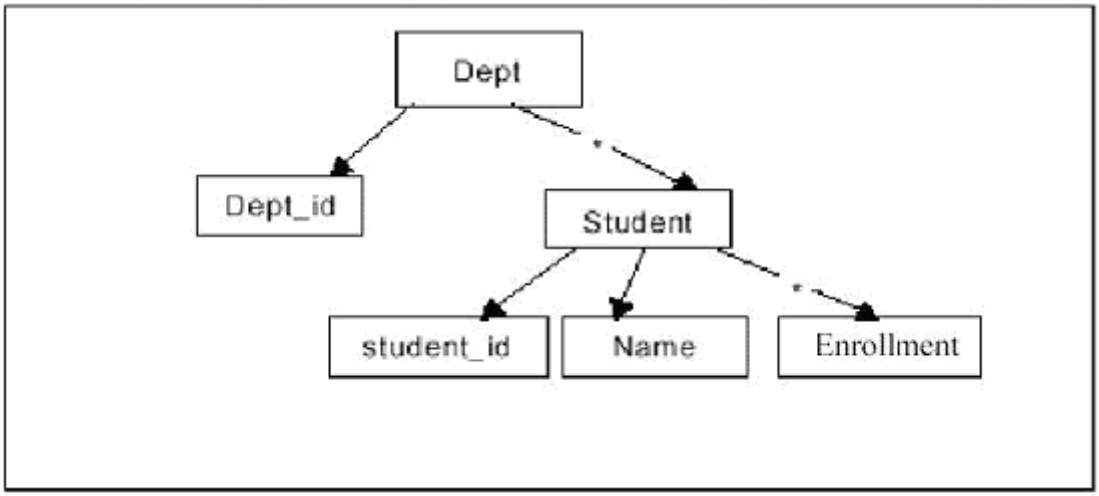

Figure 3. Structure (DDT) graph of dept.xml

By applying the Shared Inlining technique to `dept.xml`, we get the relations shown in Figure 4.

ParentID	ID	Dept_id
1	2	“dept1”

The Dept. Table

ParentID	ID	TEXT
3	5	“CS10”
3	6	“CS20”

The Enrollment Table

ParentID	ID	Student_id	Name
2	3	“123”	“St1”
2	4	“124”	“St2”

The Student Table

Figure 4. The relations obtained by applying Shared Inlining technique to `dept.xml`

As can be seen from Figure 4, there are three tables that hold the information in the XML document dept.xml. In these tables both the structural information (i.e. the hierarchical relationship among the elements) and the values that are present in the XML document are stored. The structural information is stored in ParentID and ID columns. When we look at, for example, the Enrollment table, the enrollment tuples (rows) with Text fields “CS10” and “CS20” have ID values 5 and 6, respectively. Their parent’s have the ID value 3 as can be seen from the first record in the Student table. Likewise all parent-child relationships can be easily seen from these tables. The document can be reconstructed from these tables by matching the values in the ID and the ParentID fields of proper tables..

After encrypting the field values of the tables given in Figure 4, we get the tables shown in Figure 5, Figure 6 and Figure 7.

ParentID	ID	Dept_id
Üœ‘Çv‘}ĞĞrHm□zL6@□ Ä:.ó□□¯‡öEMDv[	¿Ë‹4Fèªî'O°;C□Æ¾‹ ûm□F+ tÅMÖÌt□µ	…ğšîõkŞ…7í □ÿ□ êøp!!ŞBA§H“‘‘ñ?□Å □

Figure 5. Encrypted Dept. table

ParentID	ID	TEXT
ê×fy7ğŞ/ëòª†□xê?Ò÷ Œåé□c¸"ZN6._p	ª□f<voù &□L&€•t¡Æz»õ□ÄĞ□B;äiú □]6	#E+Z!?ÿ□□2□1İİdé□¦ZÖ□³z ,□şbÅĞ□
Ç□□²Ÿ·xˆ- □¶¨D'éæ¿r§à¥S¿uB ÿÚ□□†6	÷—k³…ä!uè□ìÆ5/ Pj?²ª□É”ó kw{Yå	èu[¦ÄR□óÓIİ«~ñå□aœ}ÛÇ□ê ´¸ˆ0È‘±Å‰

Figure 6. Encrypted Enrollment table

ParentID	ID	Student_id	Name
=Ê„§a□»€ièÌ:□´— ÀÄÔ□Yü›+ğßç0˜^u □¹	ñ§sm□DZÄ9›Ïc˜MkjÙG¸ô÷h P□ƒ1'QSv□	÷□Év,£¹□Ãohç"b1è £P□°UÂ□Û9J¤Ë□Jg Õ	h$™@©Ş □— 'iœ□€•ò*š~l£E?M YN(□*□
ö ÆÌ□à¼□7Ä-ÄÈÿ ö‡'Ö®ÎÎG.H²^Ñ¿5† □?	ÊU‰Ä!€ r*–jÃ<,é □³V ÑŒ* –-£GGÇÕ	ìôÒÄ~ÿ□©åÿñy□ˆ»- □□f□□µ□w³õ□ëË£ Øå	œWÈßd41´□_¼|□ŒÖ =9²,¿#'9¨`–¹ãÑ□\T

Figure 7. Encrypted Student table

When we look at the encrypted tables, in Figures 5, 6 and 7, we do not see any matching fields. For example The value 3 that appears in the ParentID fields of the tuples in the Enrollment table and in the ID field of the first tuple in the Student table and that is critical in establishing the parent child relationship among the Student and Enrollment tuples are encrypted differently. There are three different encryptions of the value 3. These are "ê×fy7ğŞ/ëòª†□xê?Ò÷Œå é□c¸"ZN6._p", "Ç□□²Ÿ·xˆ□¶¨D'éæ¿r§à¥S¿uBÿÚ□□†6" and "ñ§sm□DZÄ9›Ïc˜MkjÙG¸ô÷hP□ƒ1'QSv□" that appear in the first two tuples of the Enrollment table and in the first tuple of the Student table, respectively. Thus the structure of the dept.xml document can not be seen from the encrypted tables. When a user wants to construct the XML document, he/she first has to decrypt the structure aswell as the element values.

4. DISCUSSION AND CONCLUSIONS

We have chosen DES to encrypt values in our algorithm. DES has favorable features when we evaluate it using the matrix given in [6]. However we could just as well use another encryption in our algorithm.

There are various implementation of DES. In implementing our algorithm, we used an implementation of DES obtained from [7]. To evaluate the structural XML encryption algorithm that is presented in this paper, documents of various sizes have been used on a Pentium IV (2G. Hz.) PC with 128 megabyte RAM and the results shown in table 1 are obtained. The results obtained are due to the implementation of DES [7]. That is the results are for XML documents already stored in shared inlining method. Shared inlining method is one of several methods commonly used for storing XML documents even when no encryption is necessary [8].

Table 1: Evaluation of Structural XML Encryption Algorithm

EVALUATION OF THE STRUCTURAL XML ENCRYPTION ALGORITHM	
Size of the Document	**Time to encrypt/decrypt (seconds)**
1 KB	**0,4**
2 KB	**1,2**
4 KB	**2,1**
8 KB	**4,9**
16 KB	**8,3**
64 KB	**32**
256 KB	**136,5**

1 MB	**543**

The basic purpose of this paper is to introduce concept of structure encryption in XML. The basic idea behind the methodology presented is that two keys are used for encrypting an XML document. One key for encrypting the structure and the other for encrypting the elements and their values. If two keys were used to encrypt different parts of the document (let us say one key for a half and the other key for another half), then it is possible to obtain a half of the document with one key. In our methodology one does not get any useful information with one key. One key may give you just the structure information which is not useful without elements and their values or it may give you the elements in a document which again is useless without any structure (i.e. ordering and sub super element relationships).

The methodology presented is for XML documents stored in the Relational DDT approach. The structure encryption may also be applied to XML documents stored in relational databases using other approaches [2] and also for native XML storage approaches. However the details of such applications have to be discovered. The basic contribution of this paper is the introduction of the concept of structure encryption for XML.

REFERENCES

[1] http://www.w3.org/Encryption

[2] http://www.rpbourret.com/xml/XMLAndDatabases.htm

[3] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt and J. F. Naughton, Relational Databases for Querying XML Documents: Limitations and Opportunities, VLDB Proceedings, 1999, p. 302-314.

[4] M. F. Çelikel, "Structural Encryption in XML-Enabled Databases", Boğaziçi University, Computer Engineering Dept., Technical Report, July 2003.

[5] Data Encryption Standard (DES); Federal Information Processing Standards Publication; Reaffirmed 1999 October 25; U.S. Department Of Commerce/National Institute Of Standards And Technology, csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[6] N. D. Jorstad and L. T. Smith, Jr., Cryptographic Algorithm Metrics, Technology Identification and Analyses Center, January 1997, csrc.nist.gov/nissc/1997/proceedings/128.pdf

[7] DES.EXE / DEDES.EXE Version 4.00, home.planet.nl/~napel/des.htm

[8] Tian, F., DeWitt, D. J., Chen, J. and Zhang, C., "The Design and Performance Evaluation of Alternative XML Storage Strategies", *ACM SIGMOD Record*, (1): 5-10 (2002).