Efficient XQuery over Encrypted XML Documents

Azhar Rauf*, Waqas Ali†, Maher Ahmed‡, Shah Khusro* and Shaukat Ali§
*†§Department of Computer Science, University of Peshawar, Pakistan
‡Department of Physics & Computer Science, Wilfrid Laurier University, Canada
*{azhar.rauf, khusro}@upesh.edu.pk, †waqasali0702@gmail.com, ‡mahmed@wlu.ca, §shaukat191@yahoo.com

***Abstract*—Data plays an important role in today's business activities. In order to share information, data is outsourced occasionally, usually in the form of Extensible Markup Language (XML) files which may contain sensitive information only intended for specific audiences. Securing the data is, as a result, of the utmost importance. Several techniques are used for this purpose, with encryption being the most powerful method. The major disadvantage of using encryption is that it reduces the performance of those queries which perform operations on encrypted data. In this paper we propose a technique that significantly improves the performance of range XQuery on encrypted XML data. Results show that the proposed technique is more efficient than the state-of-the-art technique.**

Keywords—Database; encryption; XML; XQuery; Performance

I. Introduction

Data needs to be secured when it is being transferred over unreliable networks [1], making data security a major concern in the digital world. Several layers of security can be deployed in order to prevent access of intruders to the sensitive data. Applications and database-level security mechanisms are the most popular ways of controlling access. Using these mechanisms alone, however, may not be sufficient to protect the data from unauthorized access. Sensitive data can still be accessed if attackers break into the system [2]. Encryption is a strong layer of security [3], even if the database is breached and the intruder gets access to the data. Encryption prevents intruders from interpreting the information, as it is not in readable form. A major disadvantage of encryption is that it greatly reduces the performance of the query [4].

Data is outsourced in order to share information. Since Extensible Markup Language (XML) is platform-independent [5], it is used to store the data being outsourced. XQuery [6] is the standard for querying XML documents. Security of the XML documents is important, and encryption is a solution to this problem [7]. There are several levels of encryption. Column-level encryption is the most commonly used technique for relational databases, and element-wise encryption is used for XML documents containing sensitive data.

An XQuery whose WHERE clause contains the sensitive element requires that data in each instance of the element be decrypted in order to find the desired records. The overhead of decrypting every instance of the sensitive element is quite noticeable when the target XML document contains a large amount of data. This technique is not practical in such cases. Performance of the XQuery can be improved by eliminating the need for the extra decryption [5].

This paper proposes a technique that increases the efficiency of the XQuery on encrypted XML data. The technique removes extra decryptions in order to improve the search process, and is flexible and applicable on any type of encryption algorithm.

The rest of the paper is arranged as follows: Section II reviews the related work. Section III presents the security technique proposed in this paper. Section IV describes the architecture of the proposed security technique and its algorithmic steps. Section V discusses the experiment conducted. In section VI, the proposed technique and the state-of-the-art technique are compared, and the results of the experiment are presented. Section VII discusses the conclusion of this research.

II. Preliminaries

Encryption in XML documents is a well-known research problem. Liu and Gai [8] have proposed a column-wise encryption technique, in which data is classified into sensitive and public data. According to this technique, only those columns which contain sensitive information are supposed to be encrypted. This selectivity leads to the decryption of only those columns that contain the sensitive information, instead of decrypting the entire table which improves the performance of the query.

Chang et al. [5] have proposed a technique for querying encrypted XML documents using the Document Security Language (DSL), and referring to the structural information of the XML document by using the XML schema during the query process. The XQuery is rewritten using DSL and the XML schema. This new XQuery has the capability to locate the encrypted elements that are to be decrypted.

Nan et al. [9] have proposed a retrieval model for encrypted XML databases. They have suggested the creation of structure and value indices in the encrypted XML database, and recording of the entry address using the Bucket management algorithm. This technique supports range queries in order to improve the retrieval of encrypted data and data security.

Wang and Lakshmanan [10] have proposed a technique for storing the data and its metadata on the server side. According to this technique, the metadata should contain a Discontinuous Structural Index (DSI) and a Value Index. Structure of the XML document is hidden using the DSI and B-trees are used to support range queries. The XQuery is sent to the server from the client side, the server answers the XQuery using the stored metadata, and the results are returned to the client. At the client side, the encrypted blocks are then decrypted. This process ensures security of the data as well as efficient query evaluation.

978-1-4799-6600-4/15/$31.00 ©2015 IEEE

Shanmugasundaram et al. [11] have proposed a technique using the traditional relational database systems to evaluate queries over XML documents. The process is carried out using Document Type Descriptors (DTDs). In this technique, the XML documents are stored in the relational database tables, and semi-structured queries are transformed into SQL queries. These queries are then used to retrieve data stored in the relational tables. The resulting records are later converted to XML format. Their technique has been able to handle most queries.

III. Proposed Security Technique

Searching records in encrypted data is a well-known area of research, both for relational and XML databases. Typically, whenever an XQuery involves an encrypted XML element, the search process is carried out by first decrypting all instances of the element in the document. If even a single record is required from the XML document, all instances are decrypted in order to retrieve the results. Decrypting every instance of the encrypted element(s) significantly reduces the performance of the XQuery.

This paper proposes a novel technique that does not require the decryption of every instance of the encrypted element(s) for an XQuery. In the proposed approach, a specific number of decryptions are performed, as per the number of records fetched by an XQuery, significantly increasing the performance of the XQuery.

This method generates an additional XML document along with the original document. The original XML document contains the sensitive element(s) in encrypted form. Consider Figure 1, in which social security number is a sensitive element and is stored in encrypted form. The second XML document referred to as the Helper Document, contains the sensitive element(s) in unencrypted form, along with the primary key element in encrypted form, as shown in Figure 2. Since the sensitive element in the Helper Document is in decrypted form it is kept in a secure schema in order to prevent unauthorized access. The number of Helper Documents produced is always one for each original document regardless of the number of sensitive elements present in the original document. Data in both XML documents is encrypted, using any strong encryption algorithm.

```
<Employees>
  <employee>
    <id>1</id>
    <emp_name>John Doe</emp_name>
    <social_security_no>Encrypted Data</social_security_no>
  </employee>
  <employee>
    <id>2</id>
    <emp_name>Mark Taylor</emp_name>
    <social_security_no>Encrypted Data</social_security_no>
  </employee>
  <employee>
    <id>3</id>
    <emp_name>Jason Roberts</emp_name>
    <social_security_no>Encrypted Data</social_security_no>
  </employee>
</Employees>
```

Fig. 1. Original Document

The Helper Document is used to search whenever a sensitive element is referred to in the XQuery. If the WHERE clause in the XQuery contains a sensitive element, then the XQuery is updated, and records are searched in the Helper Document first.

```
<Employees>
  <employee>
      <id>Encrypted Data</id>
      <social_security_no>1021</social_security_no>
  </employee>
  <employee>
    <id>Encrypted Data</id>
    <social_security_no>1022</social_security_no>
  </employee>
  <employee>
      <id>Encrypted Data</id>
      <social_security_no>1023</social_security_no>
  </employee>
</Employees>
```

Fig. 2. Helper Document

Since the sensitive element is stored in unencrypted form in the Helper Document, this search process is performed quickly. If a match is found, then the primary key(s) of the respective data element(s), which are stored in encrypted form in the Helper Document, are retrieved. These encrypted primary key elements are decrypted, and a search is performed on the primary key column in the Original Document. Again as the primary key element(s) are stored in unencrypted form in the Original Document, the search process is efficient. When the search is successful, the respective sensitive data element(s), which are stored in encrypted form in the Original Document, are decrypted and fetched. The number of decryptions in the proposed technique is two times the total number of records to be retrieved.

For example, if an XQuery is searching for two encrypted records, the proposed technique will perform only four decryptions, as compared to the decryption of all instances. On the other hand, if the XQuery does not refer to the sensitive element(s) present in the document, the XQuery will execute on the Original Document in a normal fashion, and will not involve the Helper Document.

In the state-of-the-art technique, the data in each and every instance of the encrypted element must be decrypted when that element is referred to in the XQuery. The number of decryptions does not depend upon the number of records to be retrieved. Whether a single record or all records of the sensitive element are to be retrieved, the state-of-the-art technique decrypts every instance of the encrypted element. Hence, data that does not need to be retrieved by the XQuery is unnecessarily decrypted, reducing the performance of the XQuery.

IV. Architecture

Figure 3 shows the architecture of the proposed searching technique. The user inputs an XQuery, and the query is then checked. If the XQuery does not refer to the sensitive element in the Original Document, then the query is evaluated using the Original Document. Otherwise, the user is authenticated to the secure schema and the XQuery is updated to retrieve data first from the Helper Document, and then from the Original Document.

In the technique proposed in this paper, only those records are decrypted which satisfy the condition in the WHERE clause of the XQuery, once for the Helper Document, and then a second time for the Original Document. Two decryptions are performed for each retrieved record, considerably less

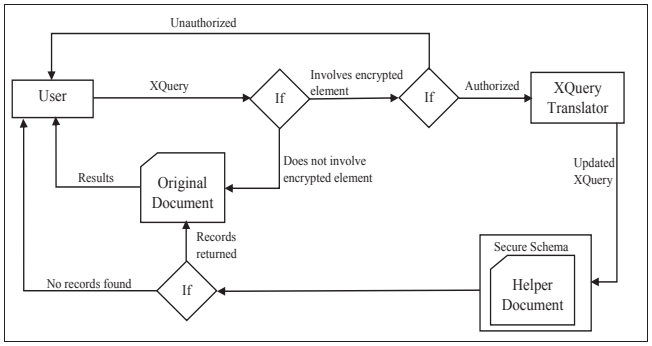

Fig. 3. Architecture of the Proposed model

than the number of decryptions involved in the state-of-the-art technique. Consider the XQuery shown in Figure 4, in which the identification and social security number of those employees whose social security number is greater than 1020 and less than 1042 is returned. The XQuery mentioned in Figure 4 targets 20 records out of the total 10,000 in the original XML document. Since the XQuery refers to social security number, which is the sensitive, encrypted element, the query uses the Helper Document to first retrieve the records that match the conditions in the WHERE clause. The XQuery is rewritten to retrieve these records from the Helper Document.

As mentioned earlier, there are 20 records in the document that match the condition given in the XQuery, so 20 records are retrieved using the XQuery listed in Fig. 5. The updated XQuery is the same as the original XQuery; the only difference is that it retrieves records from the Helper Document. According to the proposed technique, the primary key element(s) in the Helper Document in this case, the identification will be in encrypted form. Hence, the identifications of the 20 retrieved records will be decrypted. The next step is to match the identifications of the retrieved records with the Original Document, and to return the resulting records, which in this case also totals 20. Since the social security number is encrypted in the Original Document and will require decryption, another decryption for 20 records is performed, and results are shown to the user. In other words, a total of 40 decryptions are performed. The decryptions required in this technique are two times the number of records retrieved by the XQuery. Algorithm 1 shows the algorithmic steps involved in the process.

```
for $a in doc("employee.xml")/Employees/employee
where $a/social_security_no > 1020 and
      $a/social_security_no < 1042
return <employee>{($a/id,$a/social_security_no)}
       </employee>
```

Fig. 4. Original XQuery

```
for $a in doc("hlp_employee.xml")/Employees/employee
where $a/social_security_no > 1020 and
      $a/social_security_no < 1042
return <employee>{($a/id,$a/social_security_no)}
       </employee>
```

Fig. 5. Helper XQuery

Algorithm 1 Algorithm of the Proposed technique

Input: An XQuery
Output: Records returned by the XQuery
1. [XQuery]
User initiates an XQuery
2. [Checking whether the Query involves the encrypted element(s)]
if XQuery contains encrypted element(s) **then**
  if User is Authorized **then**
    goto step 3
  else
    print "Unauthorized User"
    goto step 5
  end if
else
  goto step 4
end if
3. [Check for Query match with the Helper Document]
if No match found **then**
  print "No records that match the XQuery were found"
  goto step 5
else if a match is found **then**
  a. Decrypt the data in the primary key element for the matched records.
  b. Match the decrypted data of the primary key element from the Helper Document with the data of the primary key element in the Original Document to get the corresponding records from the Original Document.
  c. Decrypt the data in the sensitive element of the matched records.
  d. goto step 5
end if
4. [Retrieval of records from Original Document]
Run the XQuery against the original XML document to get the results
5. Exit

V. Experiment Design

For the experiment outlined here, an XML document containing the data of employees of an organization was used. Social security number was identified as the sensitive element and was encrypted. The document had a total of 10,000 records. The condition in the XQuery was frequently changed in order to retrieve a different amount of data from the document. An open-source XML database, Sedna [12], was used to store the XML documents. Sedna's Java APIs were used to implement the proposed technique. These APIs provide the functionality of connecting to XML databases, loading documents into the database, and querying those documents. Encryption and decryption were performed using the classes and interfaces provided by the Apache Santuario project [13]. The experiment was carried out on a core i5 system with a 1.7GHz processor and 4Gb of RAM, running on a Windows 8 64-bit operating system.

VI. Results and Discussions

Several XQueries were used to retrieve a different number of records from the XML documents each time, using both the state-of-the-art technique and the proposed technique. Each

XQuery was run 3 times, then, the average times, shown in Table I were calculated. It is clear from Table I that whenever the number of records retrieved was less than 10% of the total records, the proposed technique performed better than the state-of-the-art technique. When the number of records retrieved was 1% of the total, the proposed technique was 6 times faster than the state-of-the-art technique. Similarly, for 2% of the retrieved records, the proposed technique worked 3.6 times faster; for 5%, times faster; and for 9%, it took almost the same amount of time as the state-of-the-art technique. The results given in Table I include the disk access times as well as some other overheads, such as the creation and deletion of XML documents. These overheads had no major effect on the time taken by the entire query process.

TABLE I. PROPOSED VS STATE-OF-THE-ART-TECHNIQUE

No. of records	State of the art	Proposed
100	25.29s	4.14s
200	24.34s	6.70s
300	25.13s	9.93s
400	24.63s	11.94s
500	24.76s	14.58s
600	25.73s	17.18s
700	25.09s	19.61s
800	25.30s	22.72s
900	26.00s	25.56s
1000	25.38s	28.27s
1100	25.77s	32.16s
1200	25.21s	35.39s
1300	25.27s	37.60s
1400	25.97s	41.08s
1500	25.52s	44.20s

The graph in Figure 6 shows the comparison between the proposed and state-of-the-art techniques. The execution of the XQuery using the state-of-the-art technique took almost the same time, no matter how many records were retrieved, as each time, all of the data present in the encrypted element was be decrypted. In the proposed technique, the number of decryptions depended upon the number of records retrieved by the XQuery. As the number of decryptions was two times the total number of records retrieved, the time taken by the XQuery also depended on the number of decryptions. The greater the number of records to be retrieved, the more time it took.

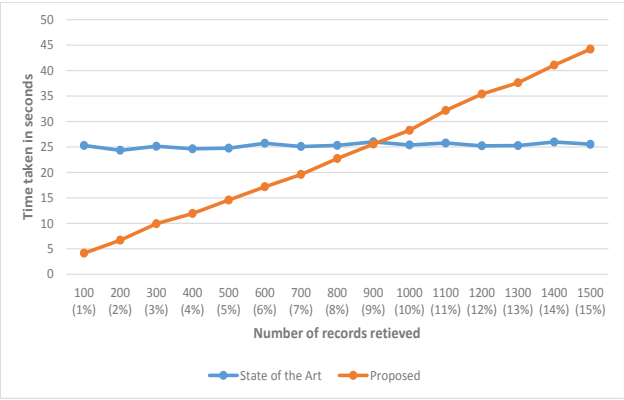

Fig. 6. Proposed VS State-of-the-Art

As shown in Figure 6, the proposed technique approached the speed of the state-of-the-art technique as the number of records retrieved increased. This equalizing occurred because of the fact that the number of decryptions increased in the proposed technique as the number of records to be retrieved increased. The lines cross each other at the 900 record mark that is, 9% of the total data. Beyond this point, the state-of-the-art technique performed better than the proposed technique. However, the typical search queries in XML documents retrieved a smaller number of records [5], showing that the proposed technique is suitable for practical applications.

VII. CONCLUSION

This paper proposes a technique to efficiently query encrypted XML documents by reducing the number of decryptions. This reduction is accomplished by splitting the target XML document into two documents. The results show that the proposed technique significantly increased the efficiency of the query whenever the number of records retrieved by an XQuery was less than 10% of the total number of records in the XML document.

REFERENCES

[1] H. Kayarkar, "Classification of various security techniques in databases and their comparative analysis," *ACTA Technica Corviniensis*, vol. 5, no. 2, pp. 135–138, Jun. 2012.

[2] O. Ünay and T. İ. Gündem, "A survey on querying encrypted xml documents for databases as a service," *ACM SIGMOD Record*, vol. 37, no. 1, pp. 12–20, Mar. 2008.

[3] U. T. Mattsson, "A practical implementation of transparent encryption and separation of duties in enterprise databases: protection against external and internal attacks on databases," in *7th IEEE Int. Conf. on E-Commerce Technology*, Munich, Germany, 2005, pp. 559–565.

[4] S. Ali, A. Rauf, and H. Javed, "A technique for handling range and fuzzy match queries on encrypted data." *Int. Arab J. Information Technology*, vol. 10, no. 3, pp. 239–244, May 2013.

[5] T.-K. Chang and G.-H. Hwang, "Developing an efficient query system for encrypted xml documents," *J. of Systems and Software*, vol. 84, no. 8, pp. 1292–1305, Aug. 2011.

[6] D. Chamberlin, "Xquery: An xml query language," *IBM systems journal*, vol. 41, no. 4, pp. 597–615, Apr. 2002.

[7] N. Liu, Y. Zhou, X. Niu, and Y. Yang, "A novel model for query over encrypted xml databases," in *IEEE Int. Conf. on Network Infrastructure and Digital Content*, Beijing, China, 2009, pp. 986–990.

[8] L. Liu and J. Gai, "A new lightweight database encryption scheme transparent to applications," in *6th IEEE Int. Conf. on Industrial Informatics*, Daejeon, South Korea, 2008, pp. 135–140.

[9] S. Nan, L. Xuanmin, and W. Heng, "A retrieve model of wesu for encrypted xml database," in *2nd Int. Workshop on Database Technology and Applications*, Wuhan, China, 2010, pp. 1–3.

[10] H. Wang and L. V. Lakshmanan, "Efficient secure query evaluation over encrypted xml databases," in *Proc. of the 32nd Int. Conf. on Very large databases*, Seoul, South Korea, 2006, pp. 127–138.

[11] J. Shanmugasundaram, E. Shekita, J. Kiernan, R. Krishnamurthy, E. Viglas, J. Naughton, and I. Tatarinov, "A general technique for querying xml documents using a relational database system," *ACM SIGMOD Record*, vol. 30, no. 3, pp. 20–26, Sep. 2001.

[12] A. Fomichev, M. Grinev, and S. Kuznetsov, "Sedna: A native xml dbms," in *SOFSEM 2006: Theory and Practice of Computer Science*, J. Wiedermann, G. Tel, J. Pokorn, M. Bielikov, and J. tuller, Eds. Germany: Springer, 2006, pp. 272–281.

[13] T. A. S. Foundation. (2013, Jun.) Apache santuario project. [Online]. Available: http://santuario.apache.org/