Computation and Search over Encrypted XML Documents

Hoi Ting Poon and Ali Miri
Department of Computer Science
Ryerson University
Toronto, Ontario, Canada
hoiting.poon@ryerson.ca, samiri@scs.ryerson.ca

***Abstract*—The need for privacy-protected searching has garnered increasing interest as industries continue to adopt cloud technologies. Much of the recent efforts have been towards incorporating more advanced searching techniques. Although many have proposed solutions for search and computations in unencrypted data, developing efficient solutions over encrypted documents remains difficult. In this paper, we investigate the problem of processing a large amount of encrypted documents in XML-like formats where a user may wish to search or compute based on certain elements in the XML tree. Our solution makes use of index tables to allow for fast keyword and location queries. To allow computations to be performed on an untrusted server, homomorphic encryption is proposed and used in conjunction with symmetric encryption to reduce computational and storage cost.**

***Keywords*-Search, XML, Privacy, Security, Encryption, Big Data**

I. Introduction

Cloud technologies promise great scalability and accessibility, but its popularity also highlights the need for greater privacy and security. While it is generally agreed that encryption is necessary to protect data security, working with encrypted data proves to be challenging. We consider the problem where a large amount of documents in XML/JSON-like formats is used by a web service hosted on a cloud server. To ensure privacy, the documents are encrypted. However, the web service provider would like to extract aggregated data from these documents. A trivial solution would be to download and decrypt all the documents to extract the relevant information, which could be expensive in bandwidth and computations, and potentially infeasible for a popular service with a significant amount of traffic. Instead, we propose a solution that allows the extraction of useful information from the encrypted documents themselves.

To the best of our knowledge, the problem of obtaining aggregated data from encrypted XML documents has not been addressed though there have been efforts towards querying of such documents. In [1], the authors described a technique for encrypting and querying XML documents in a tree structure. While the scheme is reasonably efficient, it is limited to *select* type queries and the use of trees exposes the structure of the XML documents. Sung [2] proposed using Elliptic curve cryptosystem to encrypt XML documents and Juan [3] investigated techniques to reduce the overhead in decrypting XML data. Both of these solutions considered only partially encrypted XML documents and are also limited to *select* type queries. The problem of search over encrypted data is actively being investigated by many researchers. In [4], Boneh proposed a solution involving the use of keyword searching using public key encryption which allows a set of keywords to be searchable without revealing the content of emails. Waters [5] addressed the problem of searching over encrypted audit logs. While early works focused on single keyword searches, recent works have considered conjunctive keyword searches involving multiple keywords [6], [7]. More advanced searching techniques such as phrase search [8], [9] and fuzzy keyword search [10] have also been investigated. In terms of data-mining techniques over encrypted data, Du [11] proposed a technique for secure approximate matching while Goethals [12] investigated the computation of scalar products and Damgård [13] provided solutions for secure comparisons.

In this paper, we present a scheme which combines search and computations over encrypted documents to extract aggregated data from XML documents. Unlike existing schemes, our approach can be used for partial or fully encrypted XML documents and allows computations to be performed server-side, expanding to queries that include functions such as *sum* and *avg*. We begin by presenting the communication model of the proposed scheme in section II and provide a simple description of the XML format in section III. Then, we will introduce the basic protocols required for keyword search in section IV and V. Finally, in Section VI, a brief description of homomorphic encryption and the full protocol for search and computations are presented.

II. Model for keyword search and computation over encrypted data

The communication model for our protocol involves up to three parties: The data owner, the cloud server and the user. In a private cloud, the user is simply the data owner. Figure 1 illustrates a standard protocol where the user initiates the request by sending the keywords, kw_i, and the function, $F(x)$, to the data owner. The data owner then generates a trapdoor and sends it to the cloud. A protocol to search over the requested keywords and function arguments follows.

978-1-4673-7278-7/15 $31.00 © 2015 IEEE
DOI 10.1109/BigDataCongress.2015.96

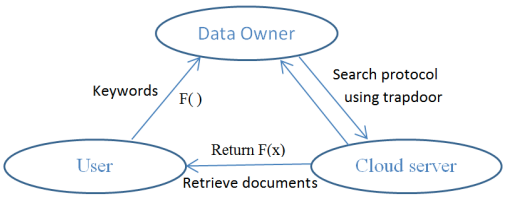

Figure 1. Communication model for search and computation over encrypted data

Finally, the cloud responds to the user with the indexes to the requested documents and the result of $F(x)$.

III. XML FORMAT

XML is a markup language that specifies how documents encoded in a format that is simple and easy to use [14]. Since its inception, it has seen wide-spread use across the Internet and many variants, such as JSON, have since been specified.

A typical XML file includes tags, attributes and contents. A tag is a markup that begins with a '<' and ends with '>'. Attributes are sometimes included in a tag to better describe the content. The following is an example of a XML file describing music in an album, where various tags such as title and artist are used and a number attribute describes ordering of the songs.

```
<ALBUM>
  <SONG number="1">
    <TITLE>Blowing in the Wind</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <PRICE>5.99</PRICE>
    <YEAR>1963</YEAR>
  </SONG>
  <SONG number="2">
    <TITLE>Lost in France</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <PRICE>6.99</PRICE>
    <YEAR>1976</YEAR>
  </SONG>
...
</ALBUM>
```

It is important to note that the location of a tag within a document is not fixed. The ordering of the tags may differ without affecting the content of the document. Certain elements may be present in some, but not all documents. The size of the content may also be inconsistent. Therefore, a searching algorithm is required for processing XML files.

IV. BASIC CONJUNCTIVE KEYWORD SEARCH PROTOCOL

We describe here a simple index based keyword search scheme.

Given a document collection, $D = \{D_1, D_2, \ldots, D_n\}$, each document, D_i, is parsed for a list of keywords, kw_j. To generate an index entry, $I(kw_j) = \{d_a, d_b, \ldots, d_n\}$, mapping keywords to documents, we set each bit, d_i, to 1 if the keyword, kw_j, is linked to the document, D_i. We then encrypt and upload the documents to the server. The keyword-to-document index is also encrypted and stored on the cloud server:

$$I(E_K(kw_j)) = \{E_K(d_a, d_b, \ldots, d_n)\}. \quad (1)$$

To perform a search, the user sends a set of keywords $kw' = \{kw_1, kw_2, \ldots, kw_q\}$ to the data owner. The data owner encrypts the search terms, $E_K(kw')$, and sends them to the cloud server. The cloud server then locates and returns the encrypted index entries to the data owner. Finally, the data owner decrypts and finds the matching documents from the intersection of the index entries:

$$I(E_K(kw_1)) \,\&\, I(E_K(kw_2)) \cdots \,\&\, I(E_K(kw_q)), \quad (2)$$

where $\&$ denotes a bitwise *and* operation.

V. SEARCH OVER ENCRYPTED XML DOCUMENTS

The basic conjunctive keyword scheme would allow a user to identify documents matching a set of keywords. However, it would not be able to access specific elements within the XML documents. For example, in order to answer the question, "What is the average price of a song by Bob Dylan?", one would first require identification of albums that contain Bob Dylan in the artist tag and retrieve the corresponding price. Note that to determine the price, one must first determine the documents listing the artist's songs, and that the artist's name contains two consecutive keywords. Matching of consecutive terms, or phrase search, in encrypted data is only recently explored by researchers. While the recent work by Tang [9] and Zittrower [8] can be used here, we present instead a light weight alternative. Interested readers can refer to [15] for a more detailed discussion on the phrase search algorithm.

The key difference between conjunctive keyword search and phrase search is that, in addition to containing the requested keywords, they must also appear contiguously in the specified order in the document.

To provide phrase search capability, a keyword location index is used. To generate the keyword location index, we compute

$$I_L(H(D_i|kw_j)) = E_K(l_1, l_2, \ldots, l_n), \quad (3)$$

where $H()$ is a cryptographically secure hash function and l_x are the locations of kw_j within D_i. Given a user phrase search request $kw' = \{kw_1, kw_2, \ldots, kw_q\}$, the data owner proceeds as in section IV to identify n documents containing all keywords. It then queries the location of the keywords by sending $H(D_i|kw_j)$ for $i = 1$ to n and $j = 1$ to q to the cloud. Given the locations, the data owner can verify that a phrase exists if, for any $l_x \in I_L(H(D_i|kw_1))$, $(l_x + j) \in I_L(H(D_i|kw_{j+1}))$, where $j = 1$ to q.

It should be noted that, even when querying single keywords, it is possible that a tag or attribute contains the queried keyword. It is rarely the intent to return results where keywords belong to both markup and content, therefore leading to false matches. When applying the basic phrase search protocol to XML documents, it is important then to also index certain symbols as keywords, in particular $<$, $>$ and /, to allow differentiation between tags and contents, and facilitate identification of sibling, child and parent elements from the XML tree.

VI. Computations over encrypted XML documents

The previous phrase search scheme would be sufficient to perform most common searches. However, despite the computational and storage efficiency of symmetric encryption, such as AES, computations cannot be performed on symmetrically encrypted data without decryption, leading to a potentially expensive process of retrieving thousands of values, if, for example, one were to compute the average price of all albums in a catalog.

A. Homomorphic encryption

Homomorphic encryption allows computations to be carried out on ciphertexts, where the results would decrypt to the corresponding computation on plaintexts. For example, an additively homomorphic scheme would have the following property:

$$E(A) + E(B) = E(A + B) \tag{4}$$

This feature allows for third parties to perform computations without exposing confidential information.

1) Paillier cryptosystem: Paillier cryptosystem [16] is one of the most popular probabilistic homomorphic encryption algorithms in the literature. The scheme involves three algorithms:

1) Key Generation: Generate two large primes, p and q, of equal length. Set $n = pq$, $g = n + 1$, $\lambda = \phi(n)$ and $\mu = \phi(n)^{-1} \bmod n$, where $\phi(n) = (p-1)(q-1)$

2) Encryption: For a message, $m \in \mathbb{Z}_n$, compute ciphertext, $c = g^m r^n \bmod n^2$, where $r \in \mathbb{Z}^*_{n^2}$

3) Decryption: For a ciphertext, $c \in \mathbb{Z}_{n^2}$, compute the message, $m = L(c^\lambda \bmod n^2)\mu \bmod n$, where $L = \left\lfloor \frac{\mu - 1}{n} \right\rfloor$

The scheme is additively homomorphic:

$$E(m_1, r_1) + E(m_2, r_2) \bmod n^2 = E(m_1 + m_2, r_3) \bmod n^2, \tag{5}$$

Although there's no known method to compute the multiplication of two ciphertexts, multiplication with plaintexts can be performed by

$$E(m_1, r_1)^{m_2} \bmod n^2 = E(m_1 m_2, r_3) \bmod n^2. \tag{6}$$

The latter is used in protocols for more complex functions such as secure scalar products [12].

B. Combining Symmetric and Homomorphic Encryption

To enable computations over encrypted data, homomorphic encryption is a natural choice. However, it is significantly more expensive than symmetric encryption in both storage and computations. To benefit from its feature while minimizing its cost, we propose adapting the encryption scheme based on the content of the document. That is, only content which may be used for computations, assumed here to be numeric data, are homomorphically encrypted, while remaining content are symmetrically encrypted. Using the example in section III, we may have $E_{AES}(</\text{ARTIST}>)$, $E_{AES}(<\text{PRICE}>)$, $E_{Paillier}(5.99)$, $E_{AES}(</\text{PRICE}>)$. Without loss of generalization, we will describe our scheme using Paillier's cryptosystem.

To enable computations over encrypted XML documents,

1) Document Encryption: Each document, D_i, is parsed and encrypted such that numeric data are encrypted using $E_{Paillier}(m)$, as described in section VI-A and the remaining content is symmetrically encrypted using AES, $E_{AES}(m)$.

2) Document Indexing: Each document, D_i, is parsed and indexed as described in section IV and V.

3) Keyword Search: To compute $F(x)$ where $tag_m = kw_1, kw_2 \ldots kw_q$ and $x \in tag_x$, we must first resolve the clause by searching for keyword, kw_i using the keyword-to-document index and the keyword location index if phrases are queried. The location of tag_x is then queried for the matched documents. Markup symbol locations are also queried to resolve potential conflicts between keywords and tags.

4) Function computation: Once the location of the function arguments are determined, the cloud server computes $F(x)$ using $E_{Paillier}(x)$ where $x \in tag_x$. For example, if the average price is desired, then $tag_x =$ '$price$' and $F(x) = \sum E_{Paillier}(x)$. Upon receiving $F(x)$, it is decrypted to obtain the sum and the average is obtained by dividing the number of matched elements.

Note that the function, $F(x)$, is not limited to summations. With minor modifications, other more complex functions can also be used such as scalar product [12] and comparison [13]. Our solution requires the exchange of two rounds of queries and the encrypted result of the function. In terms of computations, the queries consist of binary searches, hash computations and equality comparisons, each of which can be done efficiently. The efficiency of the computation of $F(x)$ depends on the function and the encryption algorithm used. In terms of summation, Paillier's cryptosystem consists of a simple integer addition modulo n^2. Since most XML documents consists of few numeric value relative to text, the computational cost of the document encryption is not believed to be significantly worse than pure symmetric encryption.

It is interesting to note that it is infeasible to determine whether a ciphertext resulted from AES encryption or Paillier by examining the ciphertext alone. Therefore, a cloud server would yield no additional information from the encrypted documents. If further security is desired, it is possible to hide the location of the desired numeric values by requesting the computation of $F(x)$ over other symmetrically encrypted data, at the expense of computational and communication cost. Other security measures can also be included in the design of the keyword-to-document index and keyword location index to defend against statistical analysis at the expense of efficiency [9], [8].

VII. Conclusion

We described a scheme for searching and computing over encrypted documents in XML-like formats with a low computational and communication cost. Due to the non-consistent placements in XML documents, a searching algorithm was required. In particular, a phrase search algorithm was required for many common search terms. Indexes provided an efficient way to access documents and keyword locations. To enable computations, homomorphic encryption was used. However, its computational and storage cost can be prohibitive depending on the application. To minimize its effect on the scheme, we restricted its use to numeric values. Despite its efficiency, the scheme is vulnerable to statistical analysis. In particular, a document set with commonly used tags in a known plaintext model could reveal meaningful information to a curious cloud server. Although the methods described in [9], [8] can provide some measures of security, they are expensive and would drastically reduce the practicality of the algorithms. As future work, we intend to investigate techniques to increase the level of security against statistical analysis while maintaining efficiency.

References

[1] Ravi Chandra Jammalamadaka and Sharad Mehrotra, "Querying encrypted xml documents," in *International Database Engineering and Applications Symposium*, 2006, pp. 129–136.

[2] Kyung-Sang Sung, Hoon Ko, and Hae-Seok Oh, "Xml document encrypt implementation using elliptic curve cryptosystem," in *International Conference on Convergence Information Technology*, 2007, pp. 2473–2478.

[3] Li Juan and Ming De-ting, "Research and application on the query processing for encrypted xml data," in *IEEE International Conference on Advanced Management Science*, 2010, pp. 707–711.

[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *In proceedings of Eurocrypt*, 2004, pp. 506–522.

[5] Brent Waters, Dirk Balfanz, Glenn Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log," in *Network and Distributed System Security Symposium*, 2004.

[6] Maozhen Ding, Fei Gao, Zhengping Jin, and Hua Zhang, "An efficient public key encryption with conjunctive keyword search scheme based on pairings," in *IEEE International Conference onNetwork Infrastructure and Digital Content*, 2012, pp. 526–530.

[7] F. Kerschbaum, "Secure conjunctive keyword searches for unstructured text," in *International Conference on Network and System Security*, 2011, pp. 285–289.

[8] S. Zittrower and C. C. Zou, "Encrypted phrase searching in the cloud," in *IEEE Global Communications Conference*, 2012, pp. 764–770.

[9] Yinqi Tang, Dawu Gu, Ning Ding, and Haining Lu, "Phrase search over encrypted data with symmetric encryption scheme," in *International Conference on Distributed Computing Systems Workshops*, 2012, pp. 471–480.

[10] He Tuo and Ma Wenping, "An effective fuzzy keyword search scheme in cloud computing," in *International Conference on Intelligent Networking and Collaborative Systems*, 2013, pp. 786–789.

[11] Wenliang Du and Mikhail J. Atallah, *Protocols For Secure Remote Database Access With Approximate Matching*, pp. 87–111, 2001.

[12] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikinen, "On private scalar product computation for privacy-preserving data mining," in *International Conference in Information Security and Cryptology*. 2004, pp. 104–120, Springer-Verlag.

[13] Ivan Damgård, Martin Geisler, and Mikkel Kroigard, "Homomorphic encryption and secure comparison," *International Journal of Applied Cryptology*, vol. 1, no. 1, pp. 22–31, 2008.

[14] "XML 1.0 Specification," http://www.w3.org/TR/REC-xml/, Accessed: March 2015.

[15] H.T. Poon and A. Miri, "An efficient conjunctive keyword and phase search scheme for encrypted cloud storage systems," to appear in *IEEE International Conference on Cloud Computing*, 2015.

[16] Pascal Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *Lecture Notes in Computer Science*, vol. 1592, pp. 223–238, 1999.