The Journal of Supercomputing (2019) 75:7147–7173
https://doi.org/10.1007/s11227-019-02920-8

An efficient stream structure for broadcasting the encrypted XML data in mobile wireless broadcast channels

Madeh Shokri[1] · **Meghdad Mirabi**[2]

Published online: 17 June 2019

Abstract
In mobile wireless broadcast networks, XML data is encrypted before it is sent over the broadcast channel in order to ensure the confidentiality of XML data. In these networks, mobile clients must not have access to all the XML data; rather they should have access to some parts of the XML data that are relevant to them and to which they are authorized to have access. In this paper, a new encrypted XML data stream structure is proposed which supports the confidentiality of XML data over the broadcast channel. In our proposed stream structure, the size of encrypted XML data stream is reduced by grouping the paths, XML nodes, texts, and attributes together. The proposed structure includes several indexes to skip from irrelevant data over the broadcast channel. The experimental results demonstrate that the use of our proposed stream structure efficiently disseminates XML data in mobile wireless broadcast networks in a secure manner and the indexes in our proposed stream structure improve the performance of XML query processing over the encrypted XML data stream.

Keywords Broadcast channel · Encrypted XML data · Indexing · XML query processing

✉ Meghdad Mirabi
meghdad.mirabi@gmail.com

Madeh Shokri
madeh.shokri@gmail.com

[1] Institute of Applied Science and Technology, Jahad Daneshgahi of Kurdistan, Sanandaj, Iran

[2] Department of Computer Engineering, Faculty of Engineering, Islamic Azad University, South Tehran Branch, Tehran, Iran

1 Introduction

XML [1] is a de facto standard language for data dissemination over the Internet. Recently, many applications are using XML for data broadcasting in wireless broadcast environments such as traffic and travel information systems and weather information systems [2–5].

Data broadcasting in mobile wireless broadcast networks is an effective way to disseminate the data [6, 7]. Two issues, namely efficient access to the data and energy consumption of mobile devices in retrieving the data over the broadcast channel, are very important in this type of networks, because mobile devices have low battery power and limited processing resources [8, 9].

Generally, mobile devices work in two modes: the active mode and the doze mode [10, 11]. The active mode is when mobile devices are consuming the maximum amount of energy or battery power, while the doze mode is when they consume the least amount of energy [12, 13].

In mobile wireless broadcast networks, XML data is disseminated in two modes: the on-demand mode and the push-based mode [2–5]. In the on-demand mode [14–19], mobile clients send their XML queries to a broadcast server through an uplink channel; then, the broadcast server receives XML queries and processes them based on their priorities. After the XML queries are processed and the XML query results are prepared, the broadcast server sends the XML query results in a downlink channel. In the push-based mode [20–24], mobile clients do not send XML queries to the broadcast server; rather the server broadcasts the XML data through the broadcast channel and mobile clients listen to the broadcast channel and receive the required XML data. As no request is sent to the broadcast server for retrieving the XML data, this mode of data dissemination is thus more cost-effective than the on-demand mode in terms of energy consumption in mobile devices. It should be noted that this paper proposes an efficient XML data stream structure to disseminate the XML data in a secure manner over a wireless broadcast channel in the push-based mode.

Actually, two different performance metrics are used for assessing the access performance to the data in a wireless broadcast stream, namely access time and tuning time. Access time refers to the period of time that mobile clients listen to the broadcast channel (i.e., send their requests) until they completely receive the required data from the broadcast channel. Therefore, it is used to measure the efficient access to mobile wireless broadcast channels. Tuning time refers to the period of time that mobile clients are in the active mode in order to retrieve the desired data from the broadcast channel and it is thus used to measure the energy consumption of mobile devices for downloading the data over the broadcast channel [25, 26].

Figure 1 shows the XML data dissemination over a mobile wireless broadcast network in the push-based mode. As shown in Fig. 1, the broadcast server first retrieves the XML data from the XML data repository and then parses it using an XML parser. The broadcast server then converts the parsed XML document into an XML data stream via the stream generation algorithm. Next, the server

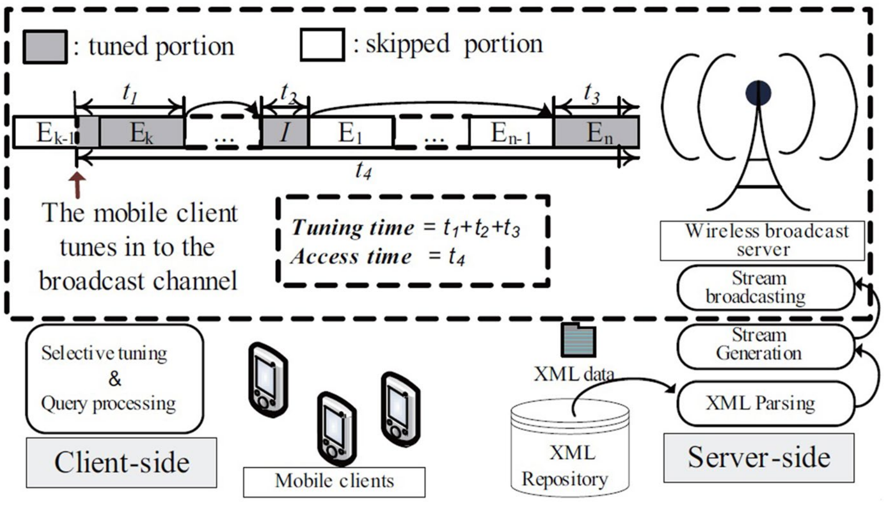

Fig. 1 Mobile wireless broadcast network architecture for XML data broadcast [22]

broadcasts the XML data stream over a wireless broadcast channel. Suppose that the mobile client c wants to receive the XML data E_n. The period of time that the mobile client c listens to the broadcast channel until the XML data is received completely is referred to as the access time, which is equal to the time t_4. The period of time that the mobile client c is in the active mode (i.e., when the XML data is being received) determines the tuning time, which is the sum of times $t1 + t2 + t3$. In fact, the times which the mobile client c does not retrieve the data from the broadcast channel and it is waiting for the desired data, it is in the doze mode and its energy consumption reduces significantly.

In mobile wireless broadcast networks, the XML data is encrypted before it is sent over the broadcast channel in order to ensure the confidentiality of XML data [27]. In addition, in these networks, mobile clients must not have access to all the XML data; rather they should have access to some parts of the XML data that are relevant to them and to which they are authorized to have access [28–36]. A variety of indexing methods have recently been proposed for XML data broadcast in mobile wireless broadcast networks [20–24]. Although these indexing methods can process XML queries efficiently over a mobile wireless broadcast channel in terms of access time and tuning time, but they do not guarantee the confidentiality of XML data over a broadcast channel.

In the literature, the only stream structure that can broadcast the encrypted XML data over a mobile wireless broadcast channel is SecNode proposed by Fathi et al. [37, 38]. The SecNode structure uses the KeyID field (to encrypt/decrypt the XML nodes), the Min(NCS) field (to jump to the desired XML nodes over the encrypted XML data stream), the Min(NIS) field (to skip from the irrelevant data over the encrypted XML data stream), and the preorder field and the postorder field (to label the XML nodes in order to process the twig pattern XML queries over the encrypted XML stream).

The goal of this paper is to propose a stream structure to disseminate the encrypted XML data over a mobile wireless broadcast channel which contains several indexes to skip the irrelevant data during the process of XML queries over the broadcast channel. The proposed stream structure efficiently disseminates XML data over a mobile wireless broadcast channel in a secure manner and the indexes in our proposed stream structure improve the performance of XML query processing over the encrypted XML data stream in terms of access time and tuning time. Hence, the main contributions of this paper are summarized as follows:

- We define a new encrypted XML data stream structure to selectively access the XML data over a mobile wireless broadcast channel. This proposed stream structure improves the performance of XML querying in terms of access time and tuning time by applying several indexes to skip from the irrelevant data over the encrypted XML data stream. The proposed stream structure also reduces the length of the encrypted XML data stream by grouping the paths, XML nodes, texts, and attributes together.
- We explain how to process the different types of XML queries over the encrypted XML data stream efficiently in terms of access time and tuning time. The proposed stream structure contains three segments, namely "Index Segment," "Data Segment," and "Text & Attribute Segment." These segments contain the information about the equivalent XML nodes and their child and sibling nodes, the structural information of equivalent XML nodes, the text and attributes values for each equivalent XML node, respectively. The mobile clients can efficiently process the simple path XML queries as well as the twig pattern XML queries using the information of these three segments.
- We evaluate the performance of our proposed stream structure in processing the different types of XML queries by performing several experiments using the different XML data sets. Our proposed structure is compared with the SecNode structure proposed by [37, 38] since the SecNode structure is the only structure that broadcasts an encrypted XML data stream over a mobile wireless broadcast channel.

The rest of this paper is organized as follows: In Sect. 2, the existing research works on the XML data dissemination and secure XML data broadcast in mobile wireless broadcast channels are reviewed. In Sect. 3, the process of generating an encrypted XML data stream based on our proposed structure is explained. In Sect. 4, the process of XML querying over the encrypted XML data stream is described. In Sect. 5, the experimental results in processing the different types of XML queries over the encrypted XML data stream are presented. Finally, in Sect. 6, the paper is concluded with a conclusion and discussion of future works.

2 Related works

In this section, we first review different ways of XML data broadcast in mobile wireless broadcast networks and then explain different methods for secure XML data broadcasting.

2.1 XML data broadcast in mobile wireless broadcast networks

The first indexing method for XML data broadcast in mobile wireless broadcast networks has been proposed in [24]. A structure called S-Node has been introduced in [24] for XML data broadcast and process the simple path XML queries. Three indexing methods called OSA,[1] TSA,[2] and SPA[3] have been introduced for the proposed structure. In the OSA indexing method, each S-Node contains the address of the next sibling node in the XML data stream, which indicates the time interval between the current S-Node and the next sibling S-Node over the broadcast channel which has been delivered. This address is suitable for mobile clients in order to jump from the current S-Node into the next sibling S-Node when the current S-Node has no child node, while the next query node does not exist or the S-Node is not the query result. The OSA indexing method can reduce the energy consumed by mobile devices to retrieve the XML data from the broadcast channel. The TSA indexing method uses two different addresses in addition to having the fields of the OSA method. In the TSA indexing method, each S-Node may contain the address of the nearest sibling node with the same tag name for that which is called same-tag-address, as well as the address of the nearest sibling node with a different tag name which is called different-tag-address. The TSA indexing method uses two additional fields (i.e., same-tag-address and different-tag-address) to process the simple path XML queries. Therefore, it is more efficient than the OSA indexing method. The SPA indexing method contains another additional field called same-path-address, which refers to the nearest cousin/sibling node with the same path name. Using the SPA indexing method, a chain of S-Node containing the same-path-address for the XML data stream is constructed, which can contribute to more effective search on the XML data stream. Although the three OSA, TSA, and SPA indexing methods can significantly process the simple XML queries, they contain replicated XML node names in the XML data stream, which increase the XML data stream size and access time; moreover, these indexing methods cannot process the twig pattern XML queries.

An indexing method based on the path summarization technique [39–41] called PS[4] has been proposed in [20]. This method eliminates redundant node names in the XML data stream and thus reduces the XML data stream size and access time, but this indexing method cannot process the twig pattern XML queries.

A new structure called DIX has been proposed in [21] for XML data broadcast in mobile wireless broadcast networks. The DIX structure contains CL[5] field (the address of the nearest DIX node with the same depth and root-to-node path), FL[6] field (the address of the nearest DIX node with the same depth but

[1] One-Sibling-Address.

[2] Two-Sibling-Addresses.

[3] Same-Path-Address.

[4] Path Summary.

[5] Clone node Link.

[6] Foreign node Link.

different root-to-node path), and LPI[7] field (the root-to-node path). Using the LPI field, mobile clients can start process the XML querying over the XML data stream at any time without the need to wait for the next broadcast cycle starts. The CL index is used to jump forward to the next candidate node which may be the result of the XML query, while the FL index is used to ignore the irrelevant parts of the XML data stream. A cluster-based technique called C-DIX has also been proposed in [21]. If the nodes with the same features in the XML data stream are clustered, mobile clients can retrieve the XML data via access to a more limited parts in the XML data stream, meaning that the access time and the tuning time are reduced. Based on these observations, a clustering indexing method based on the depth of the XML nodes was proposed. In the C-DIX indexing method, similar XML nodes are clustered in the parts called CR,[8] which can have access from the current depth to the next depth via a field called Next Depth Pointer.

In [23], an indexing method called PS + Pre/Post has been proposed. This indexing method is actually a combination of the path summary technique [39–41] and the pre/post labeling scheme [42–45]. As this indexing method makes the use of the path summary technique, it can improve the access time and tuning time by eliminating the similar node names. It also uses the pre/post labeling scheme to label the XML nodes in order to process the twig pattern XML queries over the XML data stream.

In [22], a new technique called lineage encoding has been proposed to process the twig pattern XML queries over a broadcast channel. The proposed technique uses the two types of lineage encoding called lineage vertical (V) code and horizontal (H) lineage code. The lineage coding presents the parent–child relationships between XML nodes as a sequence of bit strings called lineage code (V, H). This lineage code is an effective and lightweight technique that supports the process of twig pattern XML queries over the XML data stream.

2.2 Secure XML data broadcast

In [46], a standard has been proposed for XML data encryption by World Wide Web Consortium [47]. According to this standard, the encrypted data retains the meaning and syntax of XML and adds only an element with an encrypted string called EncryptedData. This element has four sub-elements called EncryptionMethod, KeyInfo, CipherData, and EncryptionProperties. The EncryptionMethod represents an algorithm and the parameters used to encrypt/decrypt the XML data. The KeyInfo represents a key used for encryption/decryption (not including the key value). The CipherData has a CipherValue as the sub-element, and the CipherValue represents an encrypted string generated by encrypting the element name and body. The EncryptionProperties represents the additional information related to the EncryptedData element.

[7] Location Path Information.

[8] Clustering Region.

In [48], a new method called XFlat has been proposed to broadcast an encrypted XML view over the Internet. In the XFlat method, the XML tree is first decomposed into a set of XML sub-trees based on the accessibility of XML nodes. The sub-trees include a set of XML nodes with the same accessibility. The XFlat method then encrypts each tree individually and stores it in the last view of transmitted XML. In order to process the XML query over the encrypted XML views, the XFlat method uses the flat structure under the XML view to accelerate the processing of XML query.

In [49], a query-aware decryption method has been proposed for processing the XML queries against the encrypted XML data. To decrypt a part of the encrypted XML data which may contain the XML query results, an encrypted XML index is transmitted along with the encrypted XML data on the Internet. The index information determines the location of the XML query results in the XML data, and therefore, this prevents the unnecessary decryption during the process of XML querying.

Although these methods broadcast XML data securely over the Internet, they cannot be used for XML data broadcast in mobile wireless broadcast networks, because these methods consider efficiency and scalability, while the main concern in mobile wireless broadcast networks is the effective access to data stream over a broadcast channel and the reduction of energy consumption in mobile devices to retrieve data over a broadcast channel.

In [37, 38], an indexing method has been proposed to broadcast the encrypted XML data over a mobile wireless broadcast channel. In [37, 38], a structure called SecNode, which is actually an extension of the DIX structure proposed in [21], has been provided. The SecNode structure uses the KeyID field (to encrypt/decrypt the XML nodes), the Min(NCS) field (to jump to the desired XML nodes over the encrypted XML data stream), the Min(NIS) field (to skip from the irrelevant data over the encrypted XML data stream), and the preorder field and the postorder field (to label the XML nodes in order to process the twig pattern XML queries over the encrypted XML stream). In [37, 38], an authorization code (AC) has also been defined which authorizes different users in order to have access to the XML nodes. The proposed indexing method not only processes the simple path XML query, it can also process the twig pattern XML queries.

3 The proposed encrypted XML data stream structure

The XML document shown in Fig. 2 is used to explain our proposed stream structure. Figure 3 shows the tree structure of the XML document shown in Fig. 2. As shown Fig. 3, each XML node is assigned with two rectangles where the left and right rectangles represent the accessibility of an XML node to mobile clients of the groups g1 and g2, respectively. In Fig. 3, the gray rectangles represent accessibility of the XML nodes to mobile clients of the different groups, while the white rectangles represent non-accessibility of the XML nodes to mobile clients of the different groups. For example, the root node "SigmodRecord" is accessible to mobile clients of the group g1 and g2.

```
<SigmodRecord>
  <issue>
    <volume>11</volume>
    <number Position="1010">1</number>
    <articles>
      <article>
        <title>Network</title >
        <authors>
          <author Position="00" Id="A1" >Rowe</author>
          <author Position="01" Id="A2">Michael</author>
        </authors>
        <initPage>30</initPage>
        <endPage>44</endPage>
      </article>
    </articles>
  </issue>
  <issue>
    <volume>12</volume>
    <number Position ="100">2 </number>
    <articles>
      <article>
        <title>Security</title>
        <authors>
          <author Position="00" Id="A3">Robert</author>
          <author Id="A4">  </author>
        </authors>
        <initPage>45</initPage>
        <endPage>63</endPage>
      </article>
      <article>
        <title>Computing<title>
        <authors>
          <author Position="000" Id="A500" Key="A500">Gary<author>
       </authors>
       <initPage>55</initPage>
       <endPage>68</endPage>
      </article>
    </articles>
  </issue>
</SigmodRecord>
```

Fig. 2 An example of the XML document

In order to generate an encrypted XML data stream based on our proposed structure, the paths, XML nodes, texts, and attributes are grouped together. In the following, we explain how to group the paths, XML nodes, texts, and attributes.

Definition 1 The root-to-node path of XML node *e* refers to a sequence of node names from the root node to the node *e*.

For example, in Fig. 3, the root-to-node path of the XML node "title" is "/SigmodRecord/issue/articles/article/title."

In our proposed stream structure, the paths are summarized and separated based on the Definition 1. Figure 4 shows a tree generated based on the path

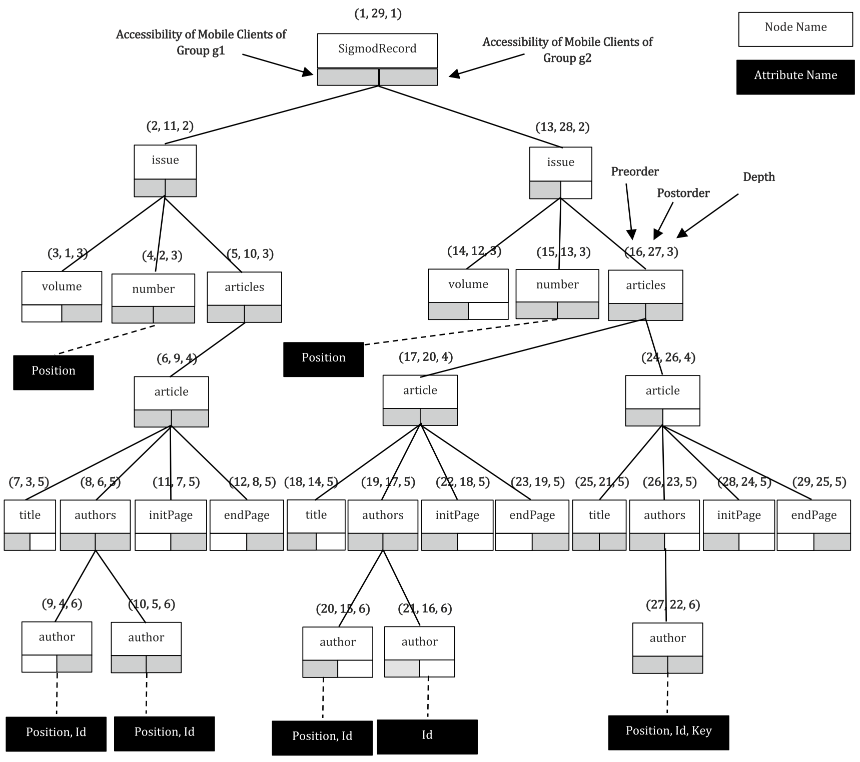

Fig. 3 The tree structure of an XML document with the accessibility and labeling of the XML nodes

summarization and separation concept. By summarizing and separating the paths, the size of encrypted XML data stream can be reduced.

In our proposed stream structure, all the XML nodes in the XML tree are labeled based on the pre/post labeling scheme [42–45]. In this XML labeling scheme, each XML node is labeled by the three values which are the preorder, postorder, and depth. These values are required for processing the twig pattern XML queries over the broadcast channel. This process will be explained in Sect. 4. Figure 3 shows the tree structure of the XML document shown in Fig. 2 when it is labeled by the pre/post labeling scheme.

Definition 2 Assume that $U=\{u_1, u_2,\ldots, u_n\}$ is a set of users (or mobile clients) and $\mathrm{UG}=\{g_1, g_2,\ldots, g_m\}$ is a set of user groups. We assume that M: $U\rightarrow\mathrm{UG}$ is a function that maps a user in U into a group in UG. The accessibility code (AC) of XML node e in the XML tree is defined by $\mathrm{AC}_e=a_1a_2\ldots a_m$ where:

$$\forall 1 \le i \le m \quad a_i = \begin{cases} 1, & \text{if the } i\text{th user group in UG, } g_i\text{, has access to the XML node } e \\ 0, & \text{otherwise} \end{cases}$$

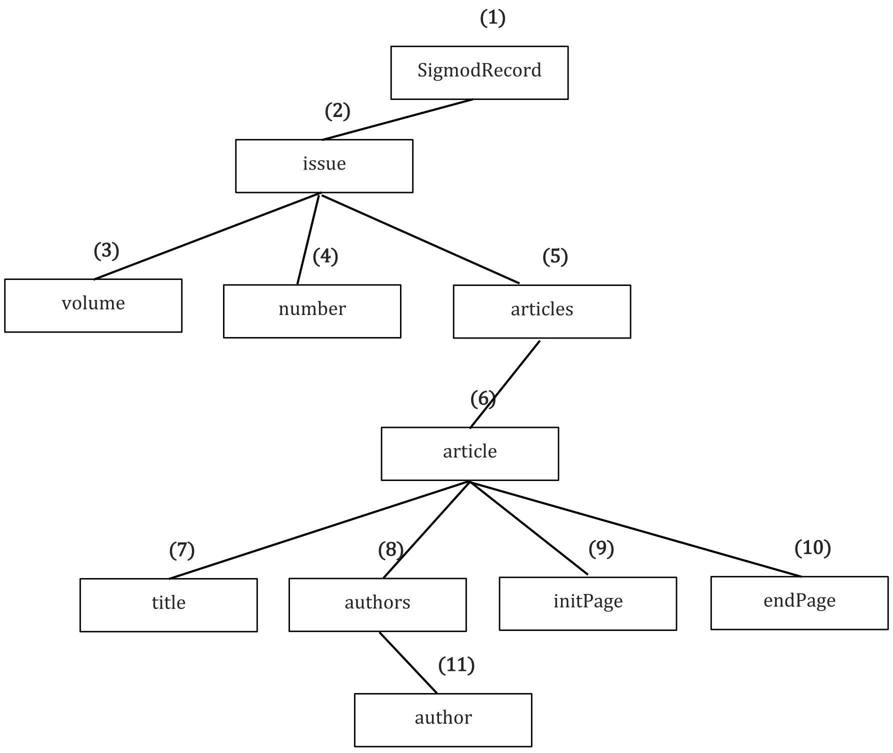

Fig. 4 The tree generated based on the path summarization and separation concept

For example, in Fig. 3, the root node "SigmodRecord" is accessible to mobile clients of the user groups g1 and g2; therefore, its accessibility code will be: $AC_{\text{SigmodRecord}} = 11$.

After labeling the XML nodes by the pre/post labeling scheme, the accessibility code (AC) for each XML node is calculated using the Definition 2.

Definition 3 The XML nodes are equivalent if they have the same root-to-node path.

For example, all the XML nodes with the node name "author" in Fig. 3 have the same root-to-node path (i.e., "/SigmodRecord/issue/articles/article/authors/author"). Therefore, these XML nodes are equivalent XML nodes.

In our proposed stream structure, the equivalent XML nodes are grouped together. By grouping the equivalent XML nodes, the size of encrypted XML data stream can be reduced.

After calculating the accessibility code (AC) of each XML node in the XML tree, the accessibility code (AC) of the equivalent XML nodes grouped together must be calculated.

Definition 4 The accessibility code (AC) of equivalent XML nodes grouped together with the node name e is defined by $\mathrm{AC}_e = \mathrm{AC}_{e1}$ **OR** AC_{e2} **OR** … **OR** AC_{ek} where **OR** is the OR-bit operator, $\forall\ 1 \le i \le k$, e_i is an XML node in the XML tree, and $\forall\ 1 \le i, j \le k$, e_i and e_j are the equivalent XML nodes.

It should be noted that the OR-bit operator is applied on all the accessibility codes (ACs) of equivalent XML nodes to determine whether each equivalent XML nodes is accessible to mobile clients of a user group or not.

For example, as shown in Fig. 3, the set of equivalent XML nodes with the node name "author" is the XML nodes with the preorder 9, 10, 20, 21, and 27. The accessibility codes (ACs) of these equivalent XML nodes are 01, 11, 10, 10, and 11, respectively. Therefore, $\mathrm{AC}_{\mathrm{Author}} = 01$ **OR** 11 **OR** 10 **OR** 10 **OR** $11 = 11$.

Definition 5 Assume that $E = \{e_1, e_2, \ldots, e_k\}$ is the set of equivalent XML nodes with the node name e ordered by preorder sequence and $\forall\ 1 \le i \le k$, the accessibility code (AC) of equivalent XML node e_i is $\mathrm{AC}_{ei} = a_{i1}a_{i2}\ldots\ a_{im}$ where $\forall\ 1 \le j \le m$, a_{ij} can be 0 or 1 which determines the accessibility of mobile clients of the jth user group to the equivalent XML node $\mathrm{e_i}$. $\forall\ 1 \le i \le k$ and $1 \le j \le m$, the accessibility code of equivalent XML nodes with the node name e for mobile clients of the jth user group is defined by $\mathrm{AC}_{e,j} = a_{1j} \mid a_{2j} \mid \ldots \mid a_{kj}$ where $\mid$ is the concatenation operator.

It should be noted that $\mathrm{AC}_{e,j}$ specifies that which of the equivalent XML nodes with the node name e are accessible/inaccessible to users of the jth user group.

For example, as shown in Fig. 3, the set of equivalent XML nodes with the node name "author" is the XML nodes with the preorder 9, 10, 20, 21, and 27. The accessibility codes (ACs) of these equivalent XML nodes are 01, 11, 10, 10, and 11, respectively. Therefore, $\mathrm{AC}_{\mathrm{author,g1}} = 0 \mid 1 \mid 1 \mid 1 \mid 1 = 01111$ (meaning that the first equivalent XML node is only inaccessible to mobile clients of the user group "g1" from 5 equivalent XML nodes with the node name "author") and $\mathrm{AC}_{\mathrm{author,\ g2}} = 1 \mid 1 \mid 0 \mid 0 \mid 1 = 11001$ (meaning that the third and fourth equivalent XML nodes are inaccessible to mobile clients of the user group "g2" from 5 equivalent XML nodes with the node name "author").

Generally, XML nodes in an XML document may have a text and some attributes. Each text has a value, and each attribute has a name and a value. By grouping the texts and attributes, the size of encrypted XML data stream can be reduced.

In our proposed stream structure, all the text values and attributes along with their names and values are extracted and grouped together. Figure 3 shows the attributes names of the XML document shown in Fig. 2 after extracting them. They are placed in the black rectangles.

Generally, our proposed stream structure contains three segments as follows:

- Index Segment: This segment contains information about the equivalent XML nodes and their child and sibling nodes.

Table 1 Structure of index segment

Field	Description
AC_e	The set of accessibility codes for the equivalent XML nodes with the node name *e*
$NXT\text{-}ADD_e$	The address of a set of equivalent XML nodes in the index segment. This set of the equivalent XML nodes is the next of the equivalent XML nodes with the node name *e*
$KeyID_e$	The cryptography key identifier for encryption/decryption of the equivalent XML nodes with the node name *e*
LEN_e	The length of equivalent XML nodes with the node name *e*
NEN_e	The name of equivalent XML nodes with the node name *e*
$Depth_e$	The depth of equivalent XML nodes with the node name *e*
ADD_e	The address of the first bit of information for the set of equivalent XML nodes with the node name *e* in the data segment
Flags	
$HAS\text{-}Child_e$	The set of equivalent XML nodes with the node name *e* has child node or not?
$HAS\text{-}Sibling_e$	The set of equivalent XML nodes with the node name *e* has sibling node or not?
$COUNT\text{-}Child_e$	The total number of child nodes of the equivalent XML nodes with the node name *e* in the index segment
$ADD\text{-}Child_e$	The set of addresses of child nodes for the equivalent XML nodes with the node name *e* in the index segment
$COUNT\text{-}Sibling_e$	The total number of sibling nodes of the equivalent XML nodes with the node name *e* in the index segment
$ADD\text{-}Sibling_e$	The set of addresses of sibling nodes for the equivalent XML nodes with the node name *e* in the index segment

- Data Segment: This segment contains the structural information of equivalent XML nodes. It should be noted that the text and attributes in the text & attribute segment can be accessible via the information existing in this segment.
- Text & Attribute Segment: This segment contains the text and attributes values for each equivalent XML node.

The index segment contains the fields shown in Table 1 for each set of equivalent XML nodes grouped together.

The field AC_e contains a set of accessibility codes for the equivalent XML nodes with the node name *e*. The field $NXT\text{-}ADD_e$ is the address of next set of equivalent XML nodes in the index segment. This field is used to skip from the irrelevant data in the index segment when the mobile client does not access to the current set of equivalent XML nodes. The field $KeyID_e$ specifies the cryptography key identifier. It is used in the process of encryption/decryption of the equivalent XML nodes with the node name *e*. The fields LEN_e and NEN_e are the length and the name of equivalent XML nodes with the node name *e*, respectively. The field $Depth_e$ is the depth of equivalent XML nodes with the node name *e*. Table 2 shows the root-to-node paths, the equivalent XML node names, the length of equivalent XML nodes, and the depth of equivalent XML nodes of the XML document shown in Fig. 2. The field ADD_e specifies the arrival time of the

Table 2 Root-to-node paths and their properties for the XML document shown in Fig. 2

Number	Root-to-node path	Equivalent XML node name	Length of equivalent XML node	Depth of equivalent XML node
1	/SigmodRecord	SigmodRecord	12	1
2	/SigmodRecord/issue	issue	5	2
3	/SigmodRecord/issue/volume	volume	6	3
4	/SigmodRecord/issue/number	number	6	3
5	/SigmodRecord/issue/articles	articles	8	3
6	/SigmodRecord/issue/articles/article	article	7	4
7	/SigmodRecord/issue/articles/article/title	title	5	5
8	/SigmodRecord/issue/articles/article/authors	authors	7	5
9	/SigmodRecord/issue/articles/article/initPage	initPage	8	5
10	/SigmodRecord/issue/articles/article/endPage	endPage	7	5
11	/SigmodRecord/issue/articles/article/authors/author	author	6	6

first bit of information for the set of equivalent XML node with the node name *e* in the data segment. In the index segment, there are two flags called HAS-Child$_e$ and HAS-Sibling$_e$ which can have values 0 or 1. A value of 0 means that the equivalent XML nodes with the node name *e* have no child or sibling node, while a value of 1 means that the equivalent XML nodes with the node name *e* have at least one child or a sibling node. If the field HAS-Child$_e$ has the zero value, the two fields COUNT-Child$_e$ and ADD-Child$_e$ are deleted from the index segment. If the field HAS-Sibling$_e$ has the zero value, the two fields COUNT-Sibling$_e$ and ADD-Sibling$_e$ are deleted from the index segment. The fields COUNT-Child$_e$ and COUNT-Sibling$_e$ specify the total number of child nodes and sibling nodes for the equivalent XML nodes with the name *e* in the index segment, respectively. The fields ADD-Child$_e$ and ADD-Sibling$_e$ specify the arrival times of the set of child nodes and sibling nodes for the equivalent XML nodes with the node name *e* in the index segment, respectively.

Definition 6 Suppose that $A = \{a_1, a_2, \ldots, a_p\}$ is a set of child nodes for the equivalent XML nodes with the node name *e* and $\mathrm{AC}_a = \{\mathrm{AC}_{a_1}, \mathrm{AC}_{a_2}, \ldots, \mathrm{AC}_{a_p}\}$ is a set of accessibility codes of the child nodes and $\mathrm{ADD}_a = \{\mathrm{ADD}_{a_1}, \mathrm{ADD}_{a_2}, \ldots, \mathrm{ADD}_{a_p}\}$ is a set of addresses (i.e., the arrival times) of the child nodes in the index segment. The set of addresses of child nodes for the equivalent XML nodes with the node name *e* is defined as follows:

$$\text{ADD-Child}_e = \left\{\forall 1 \le i \le p, (\mathrm{AC}_{a_i}, \mathrm{ADD}_{a_i}) | \mathrm{AC}_{a_i} \ne 0^m (m \text{ occurrences of } 0)\right\}$$

Table 3 Structure of data segment

Field	Description
COUNT$_e$	The total number of equivalent XML nodes with the node name *e*
PRE-ARR$_e$	An array of the preorder values for the equivalent XML nodes with the node name *e*
POST-ARR$_e$	An array of the postorder values for the equivalent XML nodes with the node name *e*
Flags	
HAS-Text$_e$	The set of equivalent XML nodes with the node name *e* has text or not?
Text-Key$_e$	If the set of equivalent XML nodes with the node name *e* has text, the code of 0 or 1 is built for each of them to determine whether it has text or not
HAS-Attribute$_e$	The set of equivalent XML nodes with the node name *e* has attribute or not?
COUNT-ATT$_e$	The total number of attributes for the equivalent XML nodes with the node name *e*
ATT-Info$_e$	If the set of equivalent XML nodes with the node name *e* has attribute(s), a table is created for them (see Table 4)
TXT-ATT-ADD$_e$	The address of the first bit of information about the text and/or attribute in the text & attribute segment in the case that the set of equivalent XML nodes with the node name *e* has at least a text or attribute

Table 4 Structure of the field ATT-Info in the data segment

Field	Description
LAN$_{\text{ATT}}$	The length of attribute *ATT*
NAN$_{\text{ATT}}$	The name of attribute *ATT*
KAN$_{\text{ATT}}$	The key of attribute *ATT*. If the set of equivalent XML nodes with the node name *e* has the attribute *ATT*, the code of 0 or 1 is built for each of them to determine to determine whether it has attribute *ATT* or not

where m represents the total number of user groups UG and a pair of $(\text{AC}_{a_i}, \text{ADD}_{a_i})$ is ordered in an ascending order based on the arrival times of child nodes in the index segment.

Definition 7 Suppose that $A = \{a_1, a_2, \ldots, a_p\}$ is a set of sibling nodes for the equivalent XML nodes with the node name e and $\text{AC}_a = \{\text{AC}_{a_1}, \text{AC}_{a_2}, \ldots, \text{AC}_{a_p}\}$ is a set of accessibility codes of the sibling nodes and $\text{ADD}_a = \{\text{ADD}_{a_1}, \text{ADD}_{a_2}, \ldots, \text{ADD}_{a_p}\}$ is a set of addresses (i.e., the arrival times) of the sibling nodes in the index segment. The set of addresses of sibling nodes for the equivalent XML nodes with the node name e is defined as follows:

$$\text{ADD-Sibling}_e = \left\{\forall 1 \le i \le p, (\text{AC}_{a_i}, \text{ADD}_{a_i}) | \text{AC}_{a_i} \neq 0^m (m \text{ occurrences of } 0)\right\}$$

where m represents the total number of user groups UG and a pair of $(\text{AC}_{a_i}, \text{ADD}_{a_i})$ is ordered in an ascending order based on the arrival times of sibling nodes in the index segment.

Table 5 Structure of text and attributes segment

Field	Description
TXT-ARR$_e$	An array of text values for the equivalent XML nodes with the node name e
ATT-ARR$_e$	An array of attribute values for the equivalent XML nodes with the node name e

The second segment in our proposed stream structure is the data segment. This segment includes fields shown in Table 3 for each set of equivalent XML nodes with the node name e. It should be noted that if there is at least an attribute ATT for one of the equivalent XML nodes with the node name e, the attribute ATT contains the fields shown in Table 4.

The third segment in our proposed stream structure is the text & attributes segment. This segment includes fields shown in Table 5 for each equivalent XML node.

In general, it may be possible that some equivalent XML nodes have text and attributes.

Definition 8 Assume that $E = \{e_1, e_2, \ldots, e_k\}$ is the set of equivalent XML nodes with the node name e ordered by preorder sequence and $\exists\ 1 \leq i \leq k$, the equivalent XML node e_i has text. Therefore, each text of equivalent XML node e_i has the following fields in the field TXT-ARR$_e$:

- TXT-Value-Length$_{e_i}$: The length of text value in the equivalent XML node e_i.
- TXT-Value$_{e_i}$: The value of text in the equivalent XML node e_i.

Definition 9 Assume that $E = \{e_1, e_2, \ldots, e_k\}$ is the set of equivalent XML nodes with the node name e ordered by preorder sequence and $\exists\ 1 \leq i \leq k$, the equivalent XML node e_i has m attributes. Therefore, $\forall\ 1 \leq j \leq m$, the jth attribute of the equivalent XML node e_i has the following fields in the field ATT-ARR$_e$:

- ATT-Value-Length$_{e_i,j}$: The length of jth attribute value in the equivalent XML node e_i.
- ATT-Value$_{e_i,j}$: The value of jth attribute in the equivalent XML node e_i.

4 XML query processing based on our proposed XML data stream structure

In this section, the process of XML querying over a broadcast channel based on our proposed stream structure is explained.

4.1 The process of simple path XML queries

Generally, a simple path XML query is defined in the form of "$/e_1/e_2/\ldots/e_n$ [text() = "…"]," whereas $\forall 1 \leq k \leq n$, e_k is an XML node name. However, the part of "[text() = "…"]" is optional in the general form of simple path XML query. In order to process a simple path XML query over a broadcast channel based on our proposed stream structure, the following steps are taken by the mobile client u:

Step 1 When the mobile client u sends the simple path XML query Q in the form of "$e_1/e_2/\ldots/e_n$" over a broadcast channel, the accessibility of the mobile client u to access a set of the equivalent XML nodes is checked by the field AC in the index segment. In the case that the mobile client u does not access to a set of the equivalent XML nodes, the next set of equivalent XML nodes is checked. The field NXT-ADD is used by the mobile client u to skip from the irrelevant data in the index segment. This process is continued until the mobile client u reaches to a set of accessible equivalent XML nodes. Then, the mobile client u decrypts the information of the equivalent XML nodes using the field KeyID.

Step 2 If the name of equivalent XML nodes with the node name e (i.e., NEN_e) is equal to the last node of the simple path XML query (i.e., e_n) and the depth of the equivalent XML nodes with the node name e (i.e., Depth_e) is equal to the depth of simple path XML query (i.e., n), the result of simple path XML query Q is the set of equivalent XML nodes with the node name e. Therefore, the field ADD of the equivalent XML nodes with the node name e (i.e., ADD_e) is read to obtain the arrival time of information of these equivalent nodes in the data segment. In the case that the depth of equivalent XML nodes with the node name e (i.e., Depth_e) is not equal to the depth of simple path XML query Q (i.e., n), the set of addresses of child nodes of the equivalent XML nodes with the node name e must be checked (i.e., ADD-Child_e) to obtain the arrival times of the child nodes in the index segment. In the case that the name of equivalent XML nodes with the node name e (i.e., NEN_e) is different from the last node of the simple path XML query Q (i.e., e_n), but their depths are the same, the set of addresses of sibling nodes of the equivalent XML nodes with the node name e must be checked (i.e., ADD-Sibling_e) to obtain the arrival times of sibling nodes in the index segment.

Step 3 When the mobile client u obtains the arrival time of the first bit of information for the set of equivalent XML nodes with the node name e (i.e., ADD_e) in the data segment, the mobile client u goes into the doze mode until the related data arrives over the broadcast channel. When the data arrives over the broadcast channel, the mobile client u is activated to retrieve the data. In the case that the equivalent XML nodes with the node name e have the text and/or attribute(s) and the simple path XML query needs to retrieve the information of text and/or attribute(s), the mobile client u can obtain the arrival time of the first bit of information in the text & attribute segment using the field TXT-ATT-ADD_e.

Example 1 Assume that the mobile client u from the user group g2 submits the simple path XML query Q = "/SigmodRecord/issue/volume" over the broadcast channel. Therefore, the accessibility of mobile client u to access the first set of the equivalent XML nodes with the node name "SigmodRecord" is checked by the field

$AC_{SigmodRecord}$ in the index segment. Since the mobile client u has access to the set of equivalent XML nodes with the node name "SigmodRecord," the mobile client u decrypts the information of this set of equivalent XML nodes using the field $KeyID_{SigmodRecord}$. Since the name of equivalent XML node "SigmodRecord" is different from the last node of simple path XML query Q (i.e., "volume"), the mobile client u has to wait for the next set of equivalent XML nodes. This process is continued until the set of equivalent XML nodes with the node name "volume" receives over the broadcast channel. Then, the mobile client u checks its accessibility to this set of equivalent XML nodes with the field AC_{volume}. Since the mobile client u has access to a subset of the equivalent XML nodes with the node name "volume," the mobile client u decrypts the information of this set of equivalent XML nodes using the field $KeyID_{volume}$. Next, the mobile client u obtains the arrival time of the first bit of information for the equivalent XML nodes with the node name "volume" (i.e., ADD_{volume_e}) in the data segment. Therefore, the mobile client u goes into the doze mode until the related data arrives over the broadcast channel. When the data arrives over the broadcast channel, the mobile client u is activated to retrieve the data. These equivalent XML nodes (i.e., XML nodes with the preorder 3 and 14) are candidates to be the result of the simple path XML query Q. Based on the accessibility of the mobile client u which is shown in Fig. 3, the mobile client u has access only to the XML node with the preorder 3. Therefore, this XML node is retrieved from the broadcast channel as the result of simple path XML query Q.

4.2 The process of twig pattern XML queries

In general, the twig pattern XML queries are divided into two categories:

- The twig pattern XML queries having a predicate condition at the end of XPath expression. This type of twig pattern XML queries is defined in the form of "$/e_1/e_2/\ldots/e_{n-i}$ [$e_{n-i+1}/\ldots/e_n$/text() = "…"]," whereas $\forall 1 \le k \le n$, e_k is an XML node name.
- The twig pattern XML queries having a predicate condition at the middle of XPath expression. This type of twig pattern XML queries is defined in the form of "$/e_1/e_2/\ldots/e_{i-1}$ [$e_i/\ldots/e_{i+p}$/text() = "…"]$/e_{i+p+1}/\ldots/e_n/$," whereas $\forall 1 \le k \le n$, e_k is an XML node name.

In order to process the twig pattern XML queries in the form of "$/e_1/e_2/\ldots/e_{n-i}$ [$e_{n-i+1}/\ldots/e_n$/text() = "…"]," the twig pattern XML query is divided into two simple path XML queries Q1 = "$/e_1/e_2/\ldots/e_{n-i}$" and Q2 = "$e_1/e_2/\ldots/e_n$[text() = "…"]." These two simple path XML queries can be processed by following the steps explained in the previous section. After downloading the results of the simple path XML queries Q1 and Q2, the mobile clients use the preorder, postorder, and depth values of the XML nodes in the query results to find the results set of the twig pattern XML query. This process is done by exploiting the following properties:

Property 1 *An XML node α labeled by (preorder$_\alpha$, postorder$_\alpha$, depth$_\alpha$) is the ancestor of XML node β labeled by (preorder$_\beta$, postorder$_\beta$, depth$_\beta$) if and only if preorder$_\alpha$<preorder$_\beta$ and postorder$_\alpha$>postorder$_\beta$.*

For example in Fig. 3, the XML node "issue" with the label (2, 11 2) is the ancestor of the XML node "initPage" with the label (11, 7, 5) since $2<11$ and $11>7$.

Property 2 *An XML node α labeled by (preorder$_\alpha$, postorder$_\alpha$, depth$_\alpha$) is the descendant of XML node β labeled by (preorder$_\beta$, postorder$_\beta$, depthβ) if and only if preorder$_\alpha$>preorder$_\beta$ and postorder$_\alpha$<postorder$_\beta$.*

For example in Fig. 3, the XML node "title" with the label (7, 3, 5) is the descendant of the node "articles" with the label (5, 10, 3) since $7>5$ and $3<10$.

Property 3 *An XML node α labeled by (preorder$_\alpha$, postorder$_\alpha$, depth$_\alpha$) is the parent of XML node β labeled by (preorder$_\beta$, postorder$_\beta$, depth$_\beta$) if and only if preorder$_\alpha$<preorder$_\beta$, postorder$_\alpha$>postorder$_\beta$ and depth$_\alpha$=depth$_\beta$ − 1.*

For example in Fig. 3, the XML node "authors" with the label (8, 6, 5) is the parent of the node "author" with the label (9, 4, 6) since $8<9$, $6>4$ and also $5=6-1$.

Property 4 *An XML node α labeled by (preorder$_\alpha$, postorder$_\alpha$, depth$_\alpha$) is the child of XML node β labeled by (preorder$_\beta$, postorder$_\beta$, depth$_\beta$) if and only if preorder$_\alpha$>preorder$_\beta$, postorder$_\alpha$<postorder$_\beta$ and depth$_\alpha$=depth$_\beta$+ 1.*

For example in Fig. 3, the XML node "number" with the label (15, 13, 3) is the child of the node "issue" with the label (13, 28, 2) since $15>13$, $13<28$ and also $3=2+1$.

Example 2 Assume that the mobile client u from the user group g1 submits the twig pattern XML query Q = "/SigmodRecord/issue[number/text() = "1"]." The mobile client u divides this query to two simple path XML queries Q1 = "/SigmodRecord/issue" and Q2 = "/SigmodRecord/issue/number[text() = "1"]." The mobile client u can process these two simple path XML queries. The result of the query Q1 is the nodes "issue" with the labels (2, 11, 2) and (13, 28, 2). Also the result of the query Q2 is the node "number" with the label (4, 2, 3). Therefore regarding that the node "issue" with the label (2, 11, 2) is the parent of the node "number" with the label (4, 2, 3) based on the Property 3, it is concluded that the final result of twig pattern XML query Q will be equal to the node "issue" with the label (2, 11 2).

To process a twig pattern XML query in the form of "$/e_1/e_2/\ldots/e_{i-1}[e_i/\ldots/e_{i+p}/$text() = "…"]$/e_{i+p+1}/\ldots/e_n$," this query is first divided into two XML queries: Q1 = "$/e_1/e_2/\ldots/e_{i-1}[e_i/\ldots/e_{i+p}/$text() = "…"]" and Q2 = "$/e_1/e_2/\ldots/e_n$." Then, the twig pattern XML query Q1 is divided into two simple path XML queries Q11 = "$/e_1/e_2/\ldots/e_{i-1}$" and Q12 = "$/e_1/e_2/\ldots/e_{i-1}/e_i/\ldots/e_{i+p}/$text() = "…"]." The XML

Table 6 Characteristics of the XML data sets

Data set	Size (KB)	Number of elements	Number of attributes	Max depth	Max fan out	Number of paths
Mondial	1743	22,423	47,423	5	955	33
SigmodRecord	467	11,526	3737	6	89	11
Shakespeare	1061	25,339	0	7	162	31

queries Q11, Q12, and Q2 are simple path XML queries, and they can be processed by following the steps explained in the previous section. After downloading the results of the simple path XML queries Q11, Q12, and Q2, the mobile client uses the preorder, postorder, and depth values of the XML nodes in the results of the queries Q11, Q12, and Q2 to find the result of the twig pattern XML query. This process is done by exploiting the Properties 1, 2, 3, and 4.

5 Performance evaluation

In this section, we evaluate the efficiency of our proposed encrypted XML data stream structure in processing the different types of XML queries over a mobile wireless broadcast channel.

All the experiments were conducted on a system with an Intel (R) Core i5-3337U-180 GHz processor, 4 GB RAM running Windows 10 Professional, and the server and client modules were implemented in Java.

5.1 Experimental settings

We logically modeled the encrypted XML data stream as a binary file, where the broadcast server writes a byte stream on the file and the mobile clients read the file and process the XML queries. We used the algorithm 3-DES for encryption/decryption. We assume that the cryptography key is first exchanged between the parties and only their KeyID is sent over the broadcast channel.

In our simulation model, we assumed that the broadcast bandwidth is fully utilized for broadcasting the encrypted XML data stream. To measure the access time and tuning time, we considered only the activity of a mobile client since the activity of a mobile client does not affect the performance of the XML querying at the other mobile clients [6–9].

We assumed that the encrypted XML data stream is disseminated and accessed in units with a fixed size (i.e., buckets), and thus, we measured the access time and tuning time in processing the XML queries by the number of buckets. A bucket is the smallest logical unit in a mobile wireless broadcast channel. In the view of assumption that the network speed is fixed, the number of buckets can be converted into time since the elapsed time for reading a bucket is computed as the bucket size divided by the network speed [5, 23]. To measure the performance variation based on the number of buckets, we used three different bucket

Table 7 XPath queries used in the experiments

XML data set	Query name	XPath expression
Mondial	MO1	/mondial/country/religions
	MO2	/mondial/country/city/population
	MO3	/mondial/country/province/city/name
	MO4	/mondial/country/languages[text() = "Kurdish"]
	MO5	/mondial/desert/located[@id = "f0_37198"]
	MO6	/mondial/country/city[name/text() = "Aarhus"]
	MO7	/mondial/country[name/text() = "Germany"]/province/city/population
SigmodRecord	SI1	/SigmodRecord/issue/volume
	SI2	/SigmodRecord/issue/articles/article/title
	SI3	/SigmodRecord/issue/articles/article/authors/author
	SI4	/SigmodRecord/issue/articles/article/initPage[text() > "50"]
	SI5	/SigmodRecord/issue/articles/article/authors/author[@position = "01"]
	SI6	/SigmodRecord/issue/articles/article[endPage/text() < "100"]
	SI7	/SigmodRecord/issue[number/text() > "2"]/articles/article/title
Shakespeare	SH1	/PLAYS/PLAY/ACT
	SH2	/PLAYS/PLAY/PERSONAE/PGROUP
	SH3	/PLAYS/PLAY/ACT/EPILOGUE/SPEECH/SPEAKER
	SH4	/PLAYS/PLAY/ACT/EPILOGUE/TITLE[text() = "EPILOGUE"]
	SH5	/PLAYS/PLAY/ACT/SCENE/SPEECH/STAGEDIR[@ SPEAKER = "BERTRAM"]
	SH6	/PLAYS/PLAY/ACT/SCENE/SPEECH[SPEAKER/text() = "CHAR-MIAN"]
	SH7	/PLAYS/PLAY/ACT/[TITLE/text() = "ACT II"]/SCENE/TITLE

sizes: 64, 128, and 256 bytes in our experiments. However, we only present the experimental results for the cases that the bucket size is set to 128 bytes since the experimental results were not dependent on the bucket size.

To measure the performance variation based on the types of the XML data sets, we used several real XML data sets. Table 6 shows the characteristics of the different XML data sets used in our experiments.

To measure the performance variation based on the types of the XML queries, we used different types of XPath queries [50]. The list of XPath queries used in our experiments is shown in Table 7.

In our simulation model, we randomly and uniformly selected the set of XML nodes from the XML data sets as the accessible nodes. We controlled the accessibility of the XML nodes with the accessibility ratio [37, 38]. The accessibility ratio is the fraction of the XML nodes in the XML data set which are accessible. To measure the performance variation based on the accessibility ratio of the XML nodes, we varied the accessibility ratio from 10 to 90% with an interval 20%.

Our proposed encrypted XML data stream structure was compared with the SecNode structure proposed by [37, 38] since the SecNode structure is the only

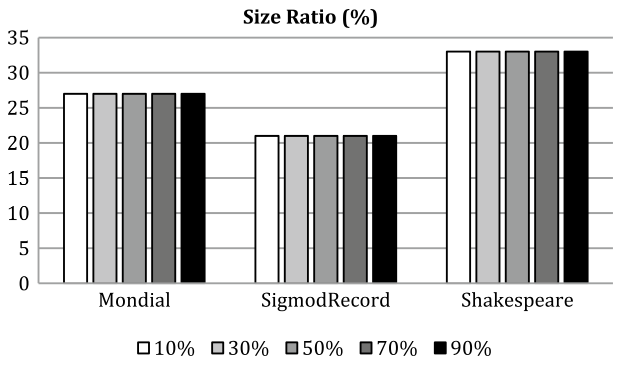

Fig. 5 Size ratio on the different XML data sets

structure that broadcasts an encrypted XML data stream over a mobile wireless broadcast channel. To do this comparison, we need to implement the SecNode structure as well. In order to reach the most accuracy in experimental results and comparison, this simulation has been conducted in the most similar state of the running computer, operating system, programming language, implementation of algorithms, and even type of variables.

The performance metrics used in our experiments were the size ratio, the tuning time ratio, and the access time ratio. They are defined as follows:

$$\text{Size ratio} = \frac{\left|SoS_{\textit{Proposed Structure}}\right|}{\left|SoS_{\textit{SecNode Structure}}\right|} \times 100 \tag{1}$$

$$\text{Tuning time ratio} = \frac{NoB_{\textit{Proposed Structure,Doze Mode}}}{NoB_{\textit{SecNode Structure,Doze Mode}}} \times 100 \tag{2}$$

$$\text{Access time ratio} = \frac{NoB_{\textit{Proposed Structure,Active Mode}} + NoB_{\textit{Proposed Structure,Doze Mode}}}{NoB_{\textit{SecNode Structure,Active Mode}} + NoB_{\textit{SecNode Structure,Doze Mode}}} \times 100 \tag{3}$$

where $\left|SoS_\alpha\right|$ represents the size of stream in the α structure and $NoB_{\alpha,\beta}$ represents the number of buckets to read in the encrypted XML data stream in the α structure once the mobile client is in the β mode.

5.2 Experimental results on the size ratio

Figure 5 shows the size ratio on the different XML data sets when the accessibility ratio is varied from 10 to 90%. As it is shown in Fig. 5, the size ratio is 26% in the Mondial data set, 21% in the SigmodRecord data set, and 33% in the Shakespeare data set. Therefore, the size of encrypted XML data stream in

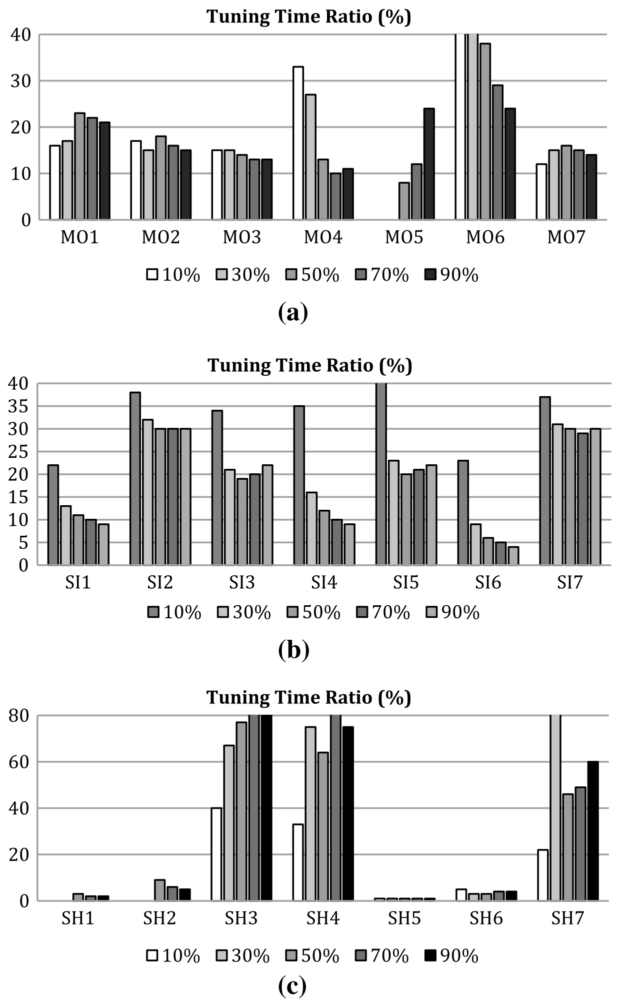

Fig. 6 **a** Tuning time ratio on the Mondial data set. **b** Tuning time ratio on the SigmodRecord data set. **c** Tuning time ratio on the Shakespeare data set

our proposed structure is less than the size of encrypted XML data stream in the SecNode structure. The reason is that the size of encrypted XML data stream in our proposed structure is reduced by grouping the paths, XML nodes, texts, and attributes together, while this information is repeated for each XML node in the SecNode structure. Moreover, it should be noted that the accessibility ratio does not affect the size of encrypted XML data stream in both structures (i.e., our proposed structure and the SecNode structure) in all the XML data sets since the XML nodes in the both structures are encrypted by the algorithm 3-DES with the same encryption key size.

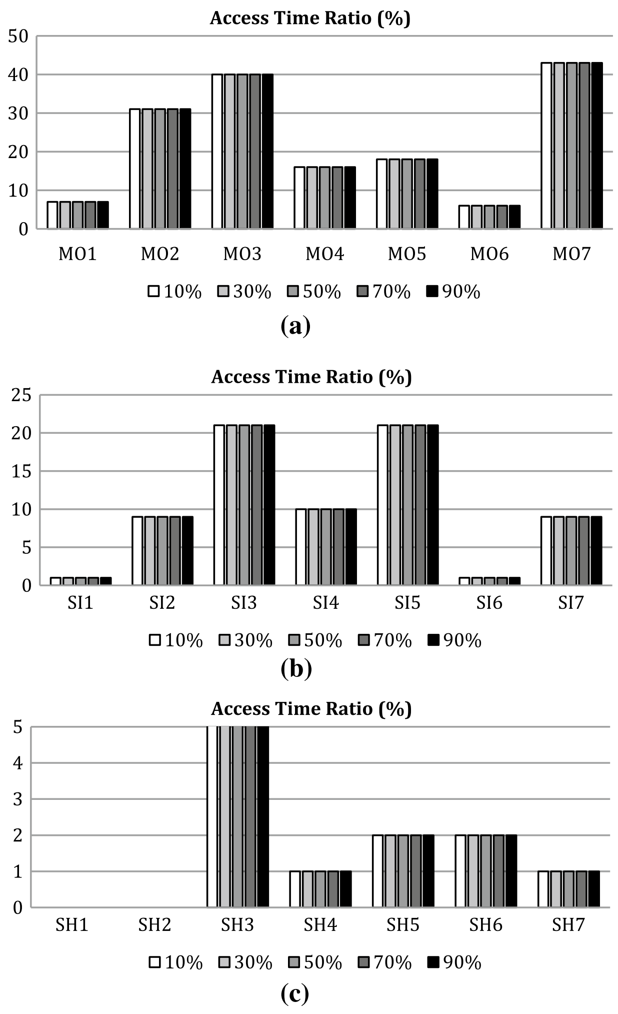

Fig. 7 **a** Access time ratio on the Mondial data set. **b** Access time ratio on the SigmodRecord data set. **c** Access time ratio on the Shakespeare data set

5.3 Experimental results on the tuning time ratio

Figure 6a–c shows the tuning time ratio in processing the different types of XML queries when the accessibility ratio varies from 10 to 90%. From Fig. 6a–c, it is clear that our proposed encrypted XML data stream structure in comparison with the SecNode structure always presents a better tuning time.

The reason for this reduction is because of the existence of several indexes in our proposed structure to skip from the irrelevant data over the stream. However, the improved value of tuning time is entirely dependent on the types of XML queries, the properties of XML data sets, the accessibility ratios, and the locations of accessible XML nodes over the encrypted XML data stream.

5.4 Experimental results on the access time ratio

Figure 7a–c shows the access time ratio in processing the different types of XML queries when the accessibility ratio varies from 10 to 90%.

From Fig. 7a–c, it is clear that our proposed encrypted XML data stream structure in comparison with the SecNode structure always presents a better access time. The reason for this reduction is that the size of encrypted XML stream in our proposed structure is less than that in the SecNode structure. It means that mobile clients should wait less time to retrieve the desired data from the broadcast channel in comparison with the SecNode structure. However, the improved value of access time is entirely dependent on the types of XML queries, the properties of XML data sets, the accessibility ratios, and the locations of accessible XML nodes over the encrypted XML data stream.

6 Conclusion and future works

In this paper, a new encrypted XML data stream structure was proposed to securely disseminate the XML data in a mobile wireless broadcast channel. The proposed stream structure contains several indexes to skip from the irrelevant data over the encrypted XML data stream. In the proposed stream structure, the paths, XML nodes, texts, and attributes were grouped together in order to reduce the length of encrypted XML data stream.

The simulation results showed that the proposed structure improves the performance of XML query processing over the encrypted XML data stream in terms of access time and tuning time in comparison with the SecNode structure. The reason was that the length of encrypted XML data stream in our proposed structure was less than that in the SecNode structure.

In the future, we intend to investigate other issues which were not considered in this paper. In this paper, only a mobile wireless broadcast channel has been used to disseminate the encrypted XML data. As a future research, this assumption can be changed and the number of broadcast channels can be increased and the efficiency of mobile clients to process the different types of XML queries in this case can be investigated. In this paper, it is assumed that the broadcast channel is reliable and there is not any bucket lost over the channel. As a research work in the future, this assumption can also be changed and an indexing method can be offered for reliable broadcasting the encrypted XML data over a mobile wireless broadcast channel.

References

1. Extensible Markup Language (XML) (2016) https://www.w3.org/XML
2. Boroujeni AB, Mirabi M (2016) A novel replication strategy for efficient XML data broadcast in wireless mobile networks. J Inf Sci Eng 32(2):309–327
3. Mirabi H, Mirabi M, Boroujeni AB (2016) An efficient XML data placement scheme over multiple wireless broadcast channels. J Inf Sci Eng 32(5):1183–1203
4. Javani M, Mirabi M (2017) An efficient index and data distribution scheme for XML data broadcast in mobile wireless networks. J Inf Sci Eng 33(1):159–182
5. Chung YD, Lee JY (2007) An indexing method for wireless broadcast XML data. Inf Sci 177(9):1931–1953
6. Imieĺinski T, Badrinath BR (1993) Data management for mobile computing. SIGMOD Record 22(1):34–39
7. Imielinski T, Viswanathan S, Badrinath BR (1994) Energy efficient indexing on air. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, USA, pp 25–36
8. Acharya S, Alonso R, Franklin M, Zdonik S (1995) Broadcast disks: data management for asymmetric communication environments. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, San Jose, CA, USA, pp 199–210
9. Imielinski T, Viswanathan S, Badrinath BR (1997) Data on air: organization and access. IEEE Trans Knowl Data Eng 9(3):353–372
10. Chung YD, Kim MH (2001) Effective data placement for wireless broadcast. Distrib Parallel Databases 9(2):133–150
11. Chung YD, Bang SH, Kim MH (2002) An efficient broadcast data clustering method for multipoint queries in wireless information systems. J Syst Softw 64(3):173–181
12. Chen M-S, Wu K-L, Yu PS (2003) Optimizing index allocation for sequential data broadcasting in wireless mobile computing. IEEE Trans Knowl Data Eng 15(1):161–173
13. Lee C-C, Leu Y (2005) Efficient data broadcast schemes for mobile computing environments with data missing. Inf Sci 172(3–4):335–359
14. Qin Y, Sun W, Zhang Z, Yu P (2007) An efficient document-split algorithm for on-demand xml data broadcast scheduling. In: Proceedings of the IET Conference on Wireless, Mobile and Sensor Networks (CCWMSN07), Shanghai, China, pp 759–762
15. Sun W, Qin Y, Yu P, Zhang Z (2007), On-demand XML data broadcast in wireless computing environments. In: Proceedings of the Third International Conference on Wireless Communications, Networking and Mobile Computing, Shanghai, China, pp 3035–3038
16. Qin Y, Sun W, Zhang Z, Yu P, He Z, Query-grouping based scheduling algorithm for on-demand xml data broadcast. In: Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing, (2008), Dalian, China, pp 1–4
17. Qin Y, Sun W, Zhang Z, Yu P, He Z, Chen W (2009) A novel air index scheme for twig queries in on-demand XML data broadcast. In: Proceedings of the 20th International Conference on Database and Expert Systems Applications (DEXA 2009), LNCS 5690, Linz, Austria, pp 412–426
18. Sun W, Yu P, Qing Y, Zhang Z, Zheng B (2009) Two-tier air indexing for on-demand XML data broadcast. In: Proceedings of the 29th IEEE International Conference on Distributed Computing Systems, Montreal, Quebec, Canada, pp 199–206
19. Qin Y, Wang H, Xia J (2011) Effective scheduling algorithm for on-demand XML data broadcasts in wireless environments. In: Proceedings of the Australasian Database Conference (ADC '11), Perth, Australia, pp 95–102
20. Park S-H, Choi J-H, Lee S (2006) An effective, efficient XML data broadcasting method in a mobile wireless network. In: Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA 2006), LNCS 4080, Krakow, Poland, pp 358–367
21. Park JP, Park C-S, Chung YD (2010) Energy and latency efficient access of wireless XML data stream. J Database Manag 21(1):58–79
22. Park JP, Park C-S, Chung YD (2013) Lineage encoding: an efficient wireless XML data streaming supporting twig pattern queries. IEEE Trans Knowl Data Eng 25(7):1559–1573
23. Mirabi M, Ibrahim H, Fathi L (2014) PS + Pre/Post: a novel structure and access mechanism for wireless XML data stream supporting twig pattern queries. Pervasive Mob Comput 15:3–25

24. Park C-S, Kim CS, Chung YD (2005) Efficient stream organization for wireless broadcasting of XML data. In: Proceedings of the 10th Asian Computing Science Conference on Advances in Computer Science: Data Management on the Web, LNCS 3818, Kunming, China, pp 223–235
25. Su T-C, Liu C-M (2010) On-demand data broadcasting for data items with time constraints on multiple broadcast channels. In: Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA 2010), LNCS 6193, (2010), Tsukuba, Japan, pp 458–469
26. Chung YD, Yoo S, Kim MH (2010) Energy and latency efficient processing of full-text searches on a wireless broadcast stream. IEEE Trans Knowl Data Eng 22(2):207–218
27. Safabahar B, Mirabi M (2017) A new structure and access mechanism for secure and efficient XML data broadcast in mobile wireless networks. J Syst Softw 125(3):119–132
28. Mirabi M, Ibrahim H, Fathi L, Udzir NI, Mamat A (2015) A dynamic compressed accessibility map for secure XML querying and updating. J Inf Sci Eng 31(1):59–93
29. Mirabi M, Ibrahim H, Udzir NI, Mamat A (2012) A compact bit string accessibility map for secure XML query processing. Procedia Comput Sci 10:1172–1179
30. Mirabi M, Ibrahim H, Udzir NI, Mamat A (2012) XML access control models and mechanisms: a survey. Int Rev Comput Softw 7(4):1518–1527
31. Mirabi M, Ibrahim H, Mamat A, Udzir NI (2011) Integrating access control mechanism with EXEL labeling scheme for XML document updating. In: Fong S (ed) Networked digital technologies. NDT 2011. Communications in Computer and Information Science, vol 136. Springer, Berlin
32. Mirabi M, Ibrahim H, Fathi L, Udzir NI, Mamat A (2011) An access control model for supporting XML document updating. In: Fong S (ed) Networked digital technologies. NDT 2011. Communications in Computer and Information Science, vol 136. Springer, Berlin
33. Bertino E, Ferrari E (2002) Secure and selective dissemination of XML documents. ACM Trans Inf Syst Secur 5(3):290–331
34. Geuer-Pollmann C (2002) XML pool encryption. In: Proceedings of the 2002 ACM Workshop on XML Security, Washington, DC, USA, pp 1–9
35. Miklau G, Suciu D (2003) Controlling access to published data using cryptography. In: Proceedings of the 29th International Conference on Very Large Data Bases, Berlin, Germany, pp 898–909
36. Ko H-K, Kim M-J, Lee S (2007) On the efficiency of secure XML broadcasting. Inf Sci 177(24):5505–5521
37. Fathi L, Ibrahim H, Mirabi M (2013) An air indexing method for encrypted XML data broadcast in mobile wireless network. In: Proceedings of the 12th International ACM Workshop on Data Engineering for Wireless and Mobile Access, New York, USA, pp 28–35
38. Fathi L, Ibrahim H, Mirabi M (2014) An energy conservation indexing method for secure XML data broadcast in mobile wireless networks. Pervasive Mob Comput 13:125–141
39. Haw S-C, Lee C-S (2009) Extending path summary and region encoding for efficient structural query processing in native XML databases. J Syst Softw 82(6):1025–1035
40. Haw S-C, Lee C-S (2011) Data storage practices and query processing in XML databases: a survey. Knowl Based Syst 24(8):1317–1340
41. Wong K-F, Yu JX, Tang N (2006) Answering XML queries using path-based indexes: a survey. World Wide Web 9(3):277–299
42. Mirabi M, Ibrahim H, Udzir NI, Mamat A (2012) An encoding scheme based on fractional number for querying and updating XML data. J Syst Softw 85(8):1831–1851
43. Mirabi M, Ibrahim H, Udzir NI, Mamat A (2011) Label size increment of bit string based labeling scheme in dynamic XML updating. In: Ariwa E, El-Qawasmeh E (eds) Digital enterprise and information systems. Communications in Computer and Information Science, vol 194. Springer, Berlin, pp 466–477
44. Mirabi M, Ibrahim H, Mamat A, Udzir NI, Fathi L (2010) Controlling label size increment of efficient XML encoding and labeling scheme in dynamic XML update. J Comput Sci 6(12):1535–1540
45. Mirabi M, Ibrahim H, Fathi L, Mamat A, Udzir NI (2011) A fractional number based labeling scheme for dynamic XML updating. In: Proceedings of the 3rd International Conference on Computing and Informatics (ICOCI 2011), Bandung, Indonesia, pp 194–200
46. Eastlake D, Reagle J, Hirsch F, Roessler T (2012) XML Encryption Syntax and Processing Version 1.1. http://www.w3.org/TR/xmlenc-core1/

47. World Wide Web Consortium (W3C) https://www.w3.org
48. Gao J, Wang T, Yang D (2008) XFlat: query-friendly encrypted XML view publishing. Inf Sci 178(3):774–787
49. Lee J-G, Whang K-Y (2006) Secure query processing against encrypted XML data using query-aware decryption. Inf Sci 176(13):1928–1947
50. Berglund A, Boag S, Chamberlin D, Fernández MF, Kay M, Robie J, Siméon J (2010) XML Path Language (XPath) 2.0, 2nd edn. http://www.w3.org/TR/2010/REC-xpath20-20101214