A Novel encrypted XML streaming technique for indexing data on multiple channels

Vinay K. Ahlawat[1], Gaurav Agarwal[2], Vikas Goel[3], Kueh Lee Hui[4*], and Mangal Sain[5*]
[1] Research Scholar, Department of CS&E, Invertis University
Bareilly, India
[1] Associate Dean, Department of ITSS, KIET Group of Institutions,
Ghaziabad, India
[e-mail: vinahlawat@gmail.com]
[2] Professor, Department of CS&E, Invertis University
Bareilly, India
[e-mail: gaurav.a1@invertis.org]
[3] Professor, Department of IT, KIET Group of Institutions,
Ghaziabad, India
[e-mail: vikas.goel@kiet.edu]
[4]Department of Electrical Engineering, Dong-A University, Busan 49236, Korea
[e-mail: leehkueh@dau.ac.kr]
[5]Division of Computer Engineering, Dongseo University, Busan 47011, South Korea
[e-mail: mangalsain1@gmail.com]
[*]Corresponding author: Kueh Lee Hui, Mangal Sain

Received June 27, 2023; revised December 9, 2023; revised January 19, 2024; revised March 5, 2024; revised April 8, 2024; accepted June 12, 2024; published July 31, 2024

Abstract

In this study, we focus on addressing the functional domain of research related to indexing XML data in wireless networks, emphasizing ensuring data confidentiality. The abstract outlines a novel indexing method designed for broadcasting encrypted XML data over wireless networks. The proposed technique involves two channels: one for indexing and another for transmitting the actual XML data. The method ensures data security by encrypting the XML stream, allowing mobile devices to access only authorized bits based on their access permissions. Despite an increase in data access time and device tuning time, the study concludes that the proposed indexing technique significantly enhances the security of transmitting XML data over mobile wireless networks.

Keywords: XML data; encrypted data; indexing technique; security of XML data.

1. Introduction

As wireless technologies expand, wireless information systems are gaining prominence. Technology development increases the ability to share information with more individuals. A variety of mobile gadgets: laptops, tablets, and smartphones access information while moving. The information must go to people as fast as possible in the case of an environmental hazard like a flood, earthquake, or any emergency for instance [6].

In the process of indexing, data arrival information is added to the data to establish once the necessary data is available on the channel. Indexing approaches are influenced by three elements: the broadcast environment i.e. single channel or multiple channels, the data type i.e. simple or XML, and the broadcast schedule. A single-channel environment just sends data items and indexes on one channel, as opposed to a multichannel environment that transmits data across multiple channels. Depending on the data type, a group of data items may all be the same size or may be modifiable. How data is distributed across broadcast channels is determined by broadcast scheduling [3,4].

XML, or Extensible Markup Language, is a widely used format for storing and transmitting structured data, including documents, databases, and web services. One of the key challenges in broadcasting XML data over mobile wireless networks is the limited bandwidth and varying quality of service that these networks can provide. To address this challenge, indexing techniques can be used to reduce the amount of data that needs to be transmitted and improve the speed and efficiency of data retrieval.

There are several approaches to indexing XML data for broadcasting in mobile wireless networks. One approach is to use content-based indexing, which involves analyzing the content of XML documents to identify relevant keywords and metadata. Another approach is to use structure-based indexing, which involves analyzing the structure of XML documents to identify relationships between elements and attributes. Because XML data is kept in plain language, it is becoming a standard for online data delivery [1,2]. This provides a method of data storage that is independent of hardware and software. There are a few sets of guidelines offered to access data, including the tree structure initially created from the XML data. It is then transmitted after being created as a stream. Access time and tuning time are calculated depending on the reception time of data at the client.

In the various research work, index management is done by B+ tree. B+ tree is the traditional, common index type which is balanced and indexes the data [22]. The challenges associated with managing encoded content in XML, emphasizing the hierarchical tree structure that leads to varying levels of confidentiality and integrity for different parts of the content. To address issues of efficiency and scalability in dissemination, a tailored approach for XML data is needed. The proposed Signature scheme in [23] aims to achieve secure and selective distribution of XML content while maintaining the overall security and privacy of the data.

However, with advanced data structures like Bpoint tree, constraints like space efficiency, query response time etc are no longer exist [21]. But these may be always the performance measure of any indexing scheme. Battery power must be regulated to minimize access time and tuning time. Therefore, power consumption and access efficiency are the two key considerations. The measuring terms for the access time and tuning time are as follows:

- Access time is the sum of all available time when a client sends a query and receives the response.
- The length of time a client tunes in to a channel is known as tuning time.

Data security on the wireless channel is a significant issue that needs to be addressed. When data is transmitted over wireless networks, it is vulnerable to interception and manipulation by

unauthorized parties. Several methods can be used to improve data security on wireless channels. One common approach is to use encryption. Another approach is to use authentication and authorization mechanisms. These controls aid in making sure that only permitted users may access the wireless network and the data being transmitted over it. Overall, ensuring data security on wireless channels is essential for protecting sensitive information and preventing unauthorized access [12-16, 25, 26].

The following is a summary of this paper's significant contributions:

1. We are proposing an encrypted XML stream scheme for broadcasting XML data on wireless channels. First, we encrypt the XML data to be broadcast and then distribute the data across multiple wireless channels. Our proposed approach improves the security of the data transmission and reduces the risk of interception or eavesdropping. Additionally, by indexing the XML nodes over multiple wireless broadcast channels, our proposed approach helps to accelerate XML searching and retrieval time.
2. We have provided various algorithms: algorithm1 creates a global index for XML documents, algorithm2 encrypts and decrypts XML documents, algorithm3 constructs tree structure for XML documents, algorithm4 creates an index segment for XML documents, and algorithm5 broadcasts data segment for XML documents. Users rapidly retrieves the location of the wireless broadcast channel's XML query results by using indexes in the generated XML data stream.
3. By conducting experiments with various XML data sets and different queries, we assess the effectiveness of the proposed encrypted XML data stream technique over multiple wireless broadcast channels. For processing XML queries, our proposed scheme evaluates the two parameters: access time and tuning time.
4. We have compared our proposed indexing scheme to existing XML data indexing systems, which often rely on centralized servers or databases to manage and index the data. The proposed approach performs better in efficiency parameters: access time and tuning time. Our proposed indexing approach encrypts the XML data and distributes the indexing across multiple wireless broadcast channels.

The rest of this research is meant to accomplish as follows: Sample XML data model along with sample XML query language are presented in Section 2. The relevant work that presented in Section 3 along with a thorough comparison of the various XML indexing methods currently in use. The suggested XML streaming method's structure, which explains how the stream is created for broadcast, is provided in Section 4. The comparison study of the parameters together with an examination of the suggested indexing strategy is presented in Section 5. The conclusion of the procedures in Section 6 was reached through analysis and comparison.

2. Sample XML Data Model & Query Language

Typically, a tree structure is used to represent an XML document. The document's items are represented as nodes in the tree structure, and the parent-child relationships between the elements are represented as edges. The paper uses a sample XML document from the ACM SIGMOD RECORD publications database [17] to demonstrate this relationship, and it provides **Figs. 1** and **2** that display the document and the matching tree structure. The tree structure is created by the Simple API for XML. Overall, since it makes manipulating the document's elements and their interactions simple, modeling an XML document using a tree structure may be a helpful strategy [20]. The challenges in disseminating XML data through wireless broadcast due to the impact of certain query elements like "*" and "//" on performance.

To address this issue, the paper [24] introduces a new indexing method called Deterministic Finite Automaton-based Index (DFAI) specifically designed for XPath queries.

In a sample XML document, the following XPath symbols, are used for selecting XML nodes:

- "/" : two XML nodes having a parent-child relationship is specified by this symbol in an XML tree.
- "//": two XML nodes having the ancestor-descendant relationship is specified by this symbol in the XML tree.
- "@": an XML node's attribute is specified by this symbol in XML tree.
- "*": any XML node with any given node name is specified by this symbol in the XML tree.
- "[]", In the XML query, to provide a predicate condition, this symbol is used.

```
<SigmodRecord>
        <issue>
                <volume>13</volume>
                <number>4</number>
                <articles>
                        <article>
                                <title> Time and databases. </title>
                                <initPage>243</initPage>
                                <endPage>245</endPage>
                                <authors>
                                        <author position="00">Gad Ariav</author>
                                        <author position="01">James Clifford</author>
                                        <author position="02">Matthias Jarke</author>
                                </authors>
                        </article>
                        <article>
                                <title> Top down statistical estimation on a database. </title>
                                <initPage>135</initPage>
                                <endPage>145</endPage>
                                <authors>
                                        <author position="00">Neil C. Rowe</author>
                                </authors>
                        </article>
                        <article>
                                <title> Top down statistical estimation on a database. </title>
                                <initPage>4</initPage>
                                <endPage>5</endPage>
                                <authors>
                                        <author position="00">Erich J. Neuhold</author>
                                </authors>
                        </article>
        </issue>
<SigmodRecord>
```

Fig. 1. Sample XML document.

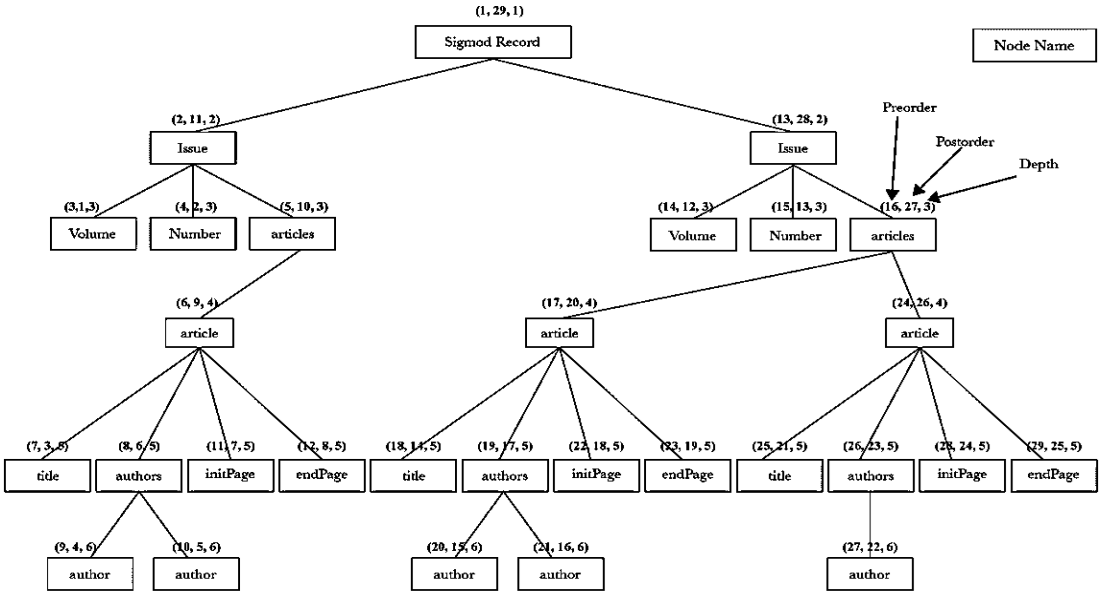

Fig. 2. XML Tree structure corresponding to the XML document in **Fig. 1**.

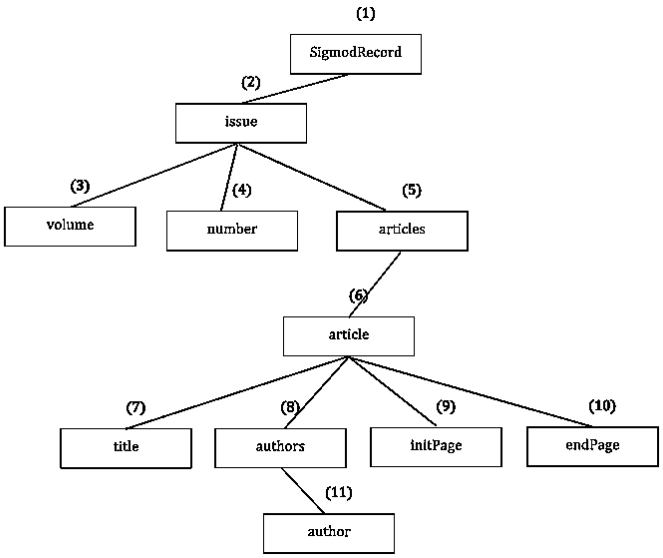

Fig. 3. The tree was created using the principles of path summarization and separation.

3. Related Work

In [7], the authors provided three indexing techniques by using simple path XML queries: One Sibling Address (OSA) is a technique used in XPath to select nodes in an XML document based on their position relative to their parent node. In an XML document, nodes are organized hierarchically, with each node having a parent node and zero or more child nodes. OSA allows us to select a specific child node based on its position relative to its siblings.

The syntax for OSA in XPath is: parent::node/*[position]

The position value in OSA starts from 1 for the first child node, and increments by 1 for each subsequent child node. If the position value is out of range (e.g., greater than the number of child nodes), then no node will be selected.
Two Sibling Address (TSA) is a technique used in XPath to select nodes in an XML document based on their position relative to their parent and grandparent nodes. In an XML document, nodes are organized hierarchically, with each node having a parent node and zero or more child nodes. TSA allows us to select a specific child node based on its position relative to its siblings and its grandparent node.

The syntax for TSA in XPath is: grandparent::parent/*[position]

TSA is useful when there are multiple levels of hierarchy in an XML document, and when we need to specify the exact location of a node relative to its siblings and grandparent node.
SPA (Same-Path-Address): This allows each S-Node to have a second address. This address is associated with the nearest node (cousin or sibling) with the same path address. This indexing technique can be more successful at locating the necessary XML nodes and an additional level of organization to the XML stream from the XML stream by building a chain of S-Nodes with the same path address. This can make it easier and more efficient to retrieve specific nodes from the stream, as the path address provides a quick and easy way to locate the desired node.[7]
An XML streaming structure proposed by Park et al. [8] for broadcasting the XML data in the push-based that utilizes the path summary approach. The basic idea is to use a technique called path summary that pre-computes a summary of the structure of the XML data. It includes the number of nodes and their paths and sends this summary to the clients. This allows the clients to efficiently navigate and extract data from the XML structure as it is being streamed to them in a push-based manner, without requiring the server to recompute the structure for each client [9].
The XML streaming structure itself can be thought of as a hierarchical tree, where each node represents a portion of the XML document. The root of this hierarchical tree represents the entire document, and each subsequent level represents a subset of the document, with each node containing summary information about its children. As the XML data is being streamed to the clients, the server sends updates to the path summary information, indicating which nodes have been added or removed. The clients use this information to efficiently navigate the XML structure and extract the data they need [9].
Park et al. [9] devised a distributed indexing approach called DIX. DIX is a distributed indexing approach to efficiently index and query XML data in a distributed environment. In DIX, each XML node is associated with several types of indexing information, including Foreign Node Link (FNL), two indexes called Content Index (CL) and Foreign Node Link Index (FL), and Location Path Information (LPI) for the FL index. The FNL is used to link related nodes across different documents or nodes distributed across different nodes in a cluster. The CL index is a traditional inverted index that indexes the content of each XML node, allowing for efficient keyword-based searches. The FL index is used for indexing the FNLs of nodes. It allows for efficient navigation and retrieval of related nodes. Finally, the LPI provides additional information such as the location path of the nodes in the XML document for the FL index.

By using multiple types of indexing information and distributing the index across multiple nodes, DIX is able can efficiently index and query XML data in a distributed environment. Additionally, by using the FNL to link related nodes, DIX can handle XML data that is distributed across multiple nodes or documents, which is a common scenario in many real-world applications [10].

The MD5 hash algorithm is used to compress the XML stream, which helps in the reduction of the state's size and makes it more manageable. This compression process converts a variable-size LPI (location path index) into a fixed size of 16 bytes. This fixed size allows for more efficient indexing and retrieval of data from the stream. The indexing techniques C-DIX based on clustering groups XML nodes with the same depth, which helps to organize the data in the stream and makes it easier to search for specific nodes. By clustering nodes together based on their depth, mobile clients can search a subset of the XML stream rather than having to search the entire stream. This can help to reduce processing time and improve the performance of XML queries [11].

One limitation is that these methods don't handle very complex XML queries. The complex queries involve wildcards, descendant axes, and predicate conditions in different lo-cation path steps. This is because the fixed-size LPIs generated by the MD5 hash function cannot capture the full complexity of the queries. Another limitation is that the XML stream's size may still be too large, even after compression and clustering. Each XML node still has a 16-byte LPI and an XML tag name, which can add up to a significant amount of data in large XML streams [11].

For the aim of managing twig pattern XML searches across the XML data stream, a unique indexing system was presented in [10]. The indexing mechanism is designed to efficiently handle queries that involve complex twig patterns, which are commonly used in applications such as XML data integration and web search engines. In an XML document, the twiglet-based indexing mechanism is a small twig pattern that captures the structural relationships between a subset of nodes. For indexing the XML data stream, the twiglet is allowed for efficient retrieval of data that matches the twig pattern.

The indexing mechanism involves several steps. First, the XML data stream is preprocessed to identify and extract all twiglets that occur in the data. Next, each twiglet is indexed using a compact data structure that captures its structural relationships. Finally, queries that involve twig patterns are processed by using the indexed twiglets to efficiently locate the nodes that match the query. One benefit of this indexing mechanism is that it is designed to handle twig patterns that involve multiple levels of nesting and complex structural relationships, which can be challenging to handle using traditional indexing techniques [10].

In [11], the authors presented a novel XML stream structure called PS+Pre/Post, which is designed to handle various sorts of simple and complex pattern XML queries over an XML stream. The structure includes multiple indexing algorithms to support efficient query processing. The PS+Pre/Post structure consists of two main components: the Path Summary (PS) and the Pre/Post indexes. The PS is a compact summary of the structure of the XML data stream, including information about the nodes and their relationships. The Pre/Post indexes are two complementary indexing algorithms that are used to index the PS and support efficient querying.

The Pre-index is used to index the paths in the PS that correspond to the starting points of a query. This allows the system to efficiently locate the nodes that match the starting points of a query. The Post index, on the other hand, is used to index the paths in the PS that correspond to the ending points of a query. This allows the system to efficiently locate the nodes that match the ending points of a query. By using both the Pre and Post indexes in combination,

the PS+Pre/Post structure can efficiently manage a wide variety of simple and complex pattern XML queries over an XML stream. Additionally, the compact size of the PS and the efficient indexing algorithms used by the Pre/Post indexes allow the system to handle large XML streams with low memory overhead [11-18].
The authors described a new structure for processing XML streams that incorporates the ideas of path summary and pre/post-labeling methods. The structure consists of a data segment and an index segment, which are created from the XML stream structure described in [11]. The index segments provide information on the root-to-node pathways in the XML stream, which can be used to quickly locate XML nodes that have that root-to-node path using some indexes. The data segment, on the other hand, provides information about the XML nodes that are connected by a root-to-node route. Nevertheless, some information about XML nodes in the XML tree, such as the content of messages and attributes, is not summarized by the new structure mentioned in the paragraph. As a result, the size of the XML stream is not reduced by this indexing technique. Several methods for effectively broadcasting and replicating XML data via wireless mobile networks. XML stream structure-based strategies were described in a previous publication [18,19].
Some of the strategies mentioned in the literature use a single broadcast channel [12-14] to copy and disseminate the XML data, while others use multiple channels [15]. Using multiple channels allows for the duplication and dispersion of XML data, which can help to reduce the size of the XML stream in each channel. By lowering access time, this size reduction helps accelerate the execution of XML queries.
The authors suggested that there are multiple approaches to efficiently disseminating XML data via wireless networks. The choice of strategy depends on various factors, such as the number of available broadcast channels and the processing speed required for XML queries. The choice of strategy may also depend on the size and complexity of the XML stream being disseminated, as well as the capabilities of the mobile devices receiving the XML data.

4. Comparative Study

The following list of variables should be taken into account while comparing the various indexing techniques for XML data broadcast:

- Method Type: Is the suggested approach a replication/distribution method or an indexing method?
- Size of XML Queries: This indicates if the proposed indexing technique can handle various XML queries with basic paths and twig patterns.
- XML Stream Size: How much may the XML stream be compressed using the specified indexing strategy?
- The kind of improvement in XML query processing: Whether the recommended may shorten access time and tuning time to process XML query.
- Traversal method used for data access
- Relationships like parent-cRelationships-2 child, child-child between nodes in the XML tree.
- Type of indexing i.e. kind of indexing tree generation.

Various indexing techniques along with their parameters of concern are summarized in **Table 1** for XML data broadcast in wireless channels.

Table 1. Listing of current XML indexing techniques with parameters to compare

S. No.	References	Broadcasting Type	Indexing Method name	Type of parameters		XML stream Size	Traversal of nodes	Relationship between various nodes of the tree	Different Indexing Techniques used to index the data
				Access time	Tuning time				
1	[7]	Indexing	OSA	✓	✓	Large	BFS/ DFS	Parent-Child	binary tree structure
2	[7]	Indexing	TSA	✓	✓	Large	BFS/ DFS	Parent-2 child (same tree)	binary tree structure
3	[7]	Indexing	SPA	✓	✓	Large	-	Parent-2 child (different tree)	Directed acyclic graph
4	[8]	Indexing	PS	✓	✓	Very small	-	Parent-Child	(breadth-first order of a path summary) List
5	[9]	Indexing	DIX	✓	✓	Very Large	-	Parent-Child	XML Tree
6	[9]	Indexing	C-DIX	✓	✓	Very Large	-	Parent-Child	XML Tree
7	[10]	Indexing	LE	✓	✓	Small	-	Attribute Value List - Text List	List

8	[11]	Indexing	PS+Pre/Post	✓	✓	Small	-	Parent-Child	XML Tree
9	[12]	Triangle-based Replication and distribution		✓	✓	Large	BFS	Parent-Child	XML Tree
10	[13]	Partial Replication and Distribution	(1, Xm)	✓	✓	Large	BFS	Parent-Child	XML Tree
11	[14]	greedy data placement algorithm		✓	✓	Large	-	-	Structural XML document
12	[15]	Replication and Distribution on multiple channels		✓	✓	Small	BFS	Root-node	XML Tree
13	[16]	Group based Indexing	GBI	✓	✓	Small	BFS	Root-node	XML Tree
14	[6]	Indexing	(1, Xm)	✓	✓	Large	BFS	Root-node	XML Tree
15	-	Proposed Indexing on multiple channels	Security (Encryption & Decryption based)	✓	✓	Large	BFS	Root-node	XML Tree

The performance of the indexing systems [8–14, 16] in terms of access time increases as the size of the broadcasted data increases. The data's sequential access is the cause of the size increase. The indexing may be different but the data placement is sequential. In [15] an indexing scheme with multiple broadcast channels is proposed that improves the access time

significantly by increasing the number of broadcasting channels. However, the issue with this multiple indexing scheme is a security concern. The indexing scheme has no authentication for the broadcasted data anyone on the channel may access the data. The above **Table 1** summarizes all the proposed indexing schemes with their key factors.

5. The proposed Encrypted XML Stream Scheme

In XML (eXtensible Markup Language) databases or systems, querying for specific information involves navigating through XML nodes, which are hierarchical structures that organize the data. XML encryption might be utilized to ensure the data secrecy of transmitted communications. You have the option of completely encrypting a communication or just a portion of it. However, employing XML encryption (either apart from or together with XML digital signatures) may have security repercussions.

A method for preserving the integrity and secrecy of data stored in XML documents is XML encryption. It has advantages in terms of security, but it also raises some difficulties and problems. Here are a few examples of the issues that using XML encryption may result in:

1. Increased Complexity: The building of applications and the processing of data can both become much more complex when using XML encryption. Data encryption and decryption require additional code, and this complexity might make the XML structure more challenging to manage and comprehend. Additionally, the system becomes more sophisticated due to the use of cryptographic libraries and encryption techniques, which could result in higher development and maintenance expenses.
2. Key Management: Data that has been encrypted must be securely managed keys. The secure distribution, storage, and revocation of encryption keys are prerequisites for XML encryption. It can be difficult to manage keys for several encrypted XML documents while maintaining their security. Ineffective management of encryption keys can result in data loss and breaches.
3. Searchability and Indexing: It is difficult to conduct searches, queries, and indexing operations on the material contained in XML documents when data is encrypted using XML encryption. Without initially decrypting the full document, it is impossible to look for certain values or elements within encrypted data because it appears as random cipher text. This may make it more difficult for programs to retrieve data effectively.
4. Data Size Increase: The size of the XML documents typically increases with XML encryption. Larger file sizes are the result of the encryption process, which adds metadata, padding, and other information to the contents. This may affect the amount of storage space and network bandwidth needed. The performance of programs can also be impacted by the growth in data size, particularly when moving data across networks or storing it in databases.
5. Interoperability: When attempting to interact with systems or programs that do not support XML encryption standards or make use of other encryption methods, XML encryption may encounter difficulties. This may cause integration issues and obstruct the efficient transfer of data across various systems.
6. Performance Overhead: Computing resources are used in the encryption and decryption of XML data, which might result in performance overhead. This is especially important for systems that frequently need to encrypt and decrypt huge amounts of XML data.

When determining whether to use XML encryption in your systems, it's crucial to give these concerns significant thought. The advantages of data security may outweigh the difficulties presented by XML encryption, depending on your particular use case and circumstances. To ensure the secure and effective usage of XML encryption in our proposed indexing XML technique, it is essential to plan for key management, data indexing, and other potential concerns.

Because the receiver key is public when using public key cryptography, XML encryption protects message confidentiality but not message integrity. The following are XML encryption best practices

- Verify the sender's identity by using digital signatures and XML encryption.
- Data elements within a message can be encrypted to prevent brute force attacks by employing a powerful encryption cipher with a long enough key length.
- Encrypt messages that include sensitive data with a powerful encryption cipher (either with transport encryption message encryption, or both).
- Use strong data encryption to encrypt messages that include sensitive data (which must stay encrypted at rest once the message is received) (not transport encryption).
- An EncryptedData element is encrypted by a procedure known as super-encryption. Zero or more EncryptedData components may be present in an XML document. A parent or child of another EncryptedData element cannot be another EncryptedData element. However, anything, even the EncryptedData and EncryptedKey parts, can represent the encrypted data itself. You must encrypt the entire EncryptedData or EncryptedKey element during super-encryption.
- Data or encrypted keys can utilize a security token to identify the encryption key that corresponds to the key needed for decryption. The decryption key can be identified without specifying a trust path or the precise contents of the certificate.

Our proposed scheme is done in two phases: (a) constructing a secured stream structure for the XML document; and (b) running XML queries using this proposed encrypted XML stream structure. This allows the XML data to be distributed across numerous wireless broadcast channels.

5.1 The Proposed Technique for Indexing the XML Data

This proposed encrypted XML streaming approach is being deployed, and compared to other earlier ways; the proposed scheme shortens access and tuning times. Its effectiveness is increased by using a variety of channels.

The concept of index segment and data segment in the proposed encrypted XML data indexing methods refers to how the XML data stream is divided and stored. The index segment is a section of the data stream that contains metadata and information about the structure of the XML document. The names of the elements, attributes, and their values are all part of this information. The index segment serves as a roadmap to help locate specific elements or attributes in XML documents. The actual data in the XML document is included in the data segment which is a data stream’s portion. This contains the attributes' values as well as the text contained in the elements. The data segment takes place where the actual content of the XML document is stored.

This proposed encrypted XML data placement method uses these two segments to enable more efficient and secure storage and retrieval of XML data. By encrypting both the index and data segments separately, and then storing them in different locations or using different encryption methods, the security of the XML data can be enhanced. Additionally, by separating the index and data segments, the retrieving and storing of XML data can be optimized, as only the

necessary segment needs to be accessed at any given time.
In our proposed encrypted XML data indexing scheme, the index part is always broadcasted on the first channel: the index channel. The data part is always transmitted on the other channel: the data channel. The index and data part of the XML data indexing scheme are encrypted for authentication. Because of index information, mobile clients may utilize the index information for determining the channel number and the time of arrival of XML data on wireless channels. The details of the index and data segment may be defined as:

- Index Segment: Information on the related XML nodes, their offspring, and siblings, is provided in this segment.
- Data Segment: The associated XML nodes' structural information is contained in the data segment. It should be noted that the information in this segment can be used to access the text and attributes in the text & attribute segment.

The proposed encrypted XML data indexing algorithm consists of the following four phases. The global index for an XML document is created in the following steps:

Step 1: A document in XML format is used to create the global index.
Step 2: After dividing the XML document into manageable pieces, the data buckets are first established. These data packets are then encrypted for broadcast security.
Step 3: The index channel is the first broadcast channel to receive the index segment.
Step 4: The data segments are sent to the broadcast second channel: the data channel.

Algorithm 1: Create a Global Index for XML Document

Step 1: XML Document Preparation

- Input: XML Document
- Output: Prepared XML Document

- Actions:
  1. Load the XML document.
  2. Perform any necessary preprocessing or cleaning of the XML data.
  3. Ensure the XML document is structured and well-formed.

Step 2: Data Bucket Creation and Encryption

- Input: Prepared XML Document
- Output: Encrypted Data Buckets

- Actions:
  1. Divide the XML document into manageable data buckets (segments).
  2. Encrypt each data bucket for broadcast security.

Step 3: Index Segment Broadcast

- Input: Index Segment
- Output: None (Broadcast to Index Channel)

- Actions:
  1. Create an index segment that contains information about the structure and location of data within the XML document.
  2. Broadcast the index segment on the index channel for all recipients to access.

Step 4: Data Segment Broadcast

- Input: Encrypted Data Segments
- Output: None (Broadcast to Data Channel)

- Actions:
  1. Encrypt specific data segments of the XML document.
  2. Broadcast these encrypted data segments on the data channel.
  3. Recipients can access the index segment to locate and request specific data segments from the data channel.

Algorithm 2: Encryption and Decryption Process of XML Document

Encryption Process:

1. Divide XML Document into Data Buckets:
   - The XML document is divided into smaller, manageable data buckets or segments.
2. Generate Encryption Key and IV:
   - For each data bucket, a unique symmetric encryption key is generated. Additionally, a random Initialization Vector (IV) is created.
3. Encrypt Data Bucket:
   - Using the encryption key and IV, the data bucket is encrypted with the chosen symmetric encryption algorithm.
4. Store Encrypted Data Bucket:
   - The encrypted data bucket is stored or transmitted securely.

Decryption (at Recipient's End):

1. Receive Encrypted Data Bucket:
   - The recipient receives the encrypted data bucket.
2. Use Encryption Key and IV:
   - The recipient uses the corresponding encryption key and IV to decrypt the data bucket.
3. Obtain Original Data:
   - The decrypted data bucket contains the original XML data, which can be processed or used as needed.

Algorithm 3: Construction of Tree Structure for XML Document

Tree Structure Construction:

The tree structure is constructed based on the hierarchical organization of the XML document, and the index segment provides the necessary information for recipients to query and retrieve specific encrypted data segments using the tree structure. The tree structure is likely used to organize and efficiently retrieve specific data segments from the encrypted XML document. Below is an explanation of how the tree structure might be constructed and utilized:

1. Define a Hierarchical Structure:
    - The XML document's hierarchical structure is reflected in the tree structure. An element or collection of elements in the XML hierarchy is represented by each node in the tree.
2. Create Nodes for XML Elements:
    - Nodes in the tree represent different elements in the XML doc. For instance, each node might correspond to an XML tag or a set of related tags.
3. Parent-Child Relationships:
    - Determine parent-child relationships between nodes by using the XML document's element nesting structure. This ensures that the tree structure mirrors the hierarchical organization of the XML data.

Algorithm 4: Creation of Index Segment for XML Document

Index Segment Creation:

1. Information Storage:
    - The index segment, created in Step 3, contains information about the structure and location of data within the XML document.
2. Mapping to Tree Nodes:
    - The index segment likely includes mappings between specific elements in the XML document and corresponding nodes in the constructed tree structure.
3. Node Position and Size:
    - Information about the position and size of each data segment (or bucket) within the encrypted XML document is stored in the index segment. This allows recipients to locate specific data segments efficiently.

Algorithm 5: Broadcasting Data Segment for XML Document

Data Segment Broadcast:

1. Encrypt Specific Data Segments:
    - In Step 4, specific data segments of the XML document are encrypted individually. These segments correspond to nodes or subtrees in the constructed tree.

2. Broadcast Encrypted Data Segments:
    - The encrypted data segments are broadcast on the data channel. Each segment is associated with a specific node or set of nodes in the tree structure.
3. Querying Using Index Information:
    - Recipients can access the index segment (broadcasted in Step 3) to understand the structure of the XML document and locate the nodes or elements they are interested in.
4. Efficient Retrieval:
    - The tree structure enables efficient retrieval of specific data segments. Recipients can navigate the tree based on the information provided in the index segment to find and request the desired encrypted data segments from the data channel.

5.2 Structure of Index & Data Nodes

XML stream has two sorts of nodes i.e. data nodes and index nodes. The data of the document are stored in the data node along with additional information describing its form. The information about the data's form and location is contained in the Index node. In the sample XML tree described in **Fig. 3** and **Fig. 4**, the structure of both nodes is set. These nodes are referred to as buckets: the smallest unit used for data broadcasting.

5.2.1 Index node structure

Many fields in the index node are used to specify various purposes. **Fig. 4** depicts the index node's form. The first field STRT is used to specify whether or not the node is the starting one. The bucket type field is the second node that is used to learn what type of material, such as data or an index, is stored in the field. The channel number is the third field that provides the channel count. Clients can switch to another channel using this field based on the data they need. The index table is the fourth field in this. The index tree's nodes are contained in this, which is of the array type. The final field in this shows the connection to the next node in the index [6].

STRT	BKT TYPE	CH_FIELD	INDX_TABLE[ ]	LINK_TO_SUCCESSSIVE_INDEX

Fig. 4. Structural representation of index node.

5.2.2 Data node structure

This provides a structure of a data node that has five fields each field serves a different purpose. **Fig. 5** shows the data node's structure. The STRT field is the first field and is used to provide the starting point of the segment. The Bkttype field is the second field and is used to indicate what kind of bucket it is. The CH FIELD is the third field that lets the client know which channel needs to be switched. The DT_ Type is the fourth field and contains the primary data in the form of an array taken from an XML tree. Connection to the following index node is contained in the final field [6].

STRT	BKT TYPE	CH_FIELD	DT_TYPE[ ]	LINK_TO_SUCCESSSIVE_DATA

Fig. 5. Structural representation of data node.

Table 2. In the XML document shown in **Fig. 2**, the Path from root to node and each node's attributes are shown below [18]

No.	The path	Corresponding XML node name	Equivalent XML node's length	Equivalent XML node's Depth
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

Table 3. Features of the XML data sets

Dataset	Size of the Dataset (KB)	No. of elements	No. of attributes	Max depth	Max fan out	No. of paths	Avg. Path
Mondial	1033	22,423	47,423	5	955	33	3.59274
SigmodRecord	467	11,526	3737	6	89	11	5.14107
University Courses	277	10546	0	4	142	21	3.19979

6. Analysis

The effectiveness of this suggested encrypted XML data stream structure in responding to various XML query kinds through a wireless broadcast channel for mobile devices is evaluated in this section. The architecture consists of a broadcast server that publishes a byte stream to the encrypted XML data stream, which is conceptually represented as a binary file. The XML queries are executed by the mobile clients after reading the file.

To ensure the security of the data, the authors have utilized the 3-DES encryption/decryption technique. They presume that only the parties' KeyIDs are delivered via the broadcast channel once the parties have first exchanged the cryptographic key. This approach ensures that the data is secure and that only authorized parties can access it.

The evaluation of the proposed structure's efficiency is critical in determining its usefulness in practical applications. The authors have assessed its performance in responding to various XML query types, which is essential for mobile devices' efficient data retrieval. The results of this evaluation will help to determine the effectiveness and suitability of the proposed structure for use in real-world scenarios.

The server and client components were created in Java. The tests were performed on a machine running Windows 11 Professional with an Intel (R) Core i7 CPU and 8 GB of RAM using a JavaTM Cryptography Extension (JCE) policy file to strengthen cryptographic operations.

We assume that units of a fixed size are broadcasted and accessed from the XML stream. Then, the number of buckets is utilized to determine the tuning time and access time for different sorts of XML queries. A bucket is the smallest logical unit of the wireless broadcast channel that is used to send data.

The authors assume that the network speed is constant to convert the number of buckets into time. By dividing the bucket size by the network speed, they may calculate the amount of time needed to read a bucket. This allows us to evaluate the time it takes to read and process a bucket of data, which is an essential metric in determining the efficiency of the proposed structure. This information is crucial in optimizing the structure's design and implementation to ensure efficient data retrieval and processing.

To examine the performance variance dependent on the types of XML data sets, we have employed multiple real and syntactic XML data sets. **Table 3** lists the characteristics of the XML data sets that we used for our tests. To measure the performance variance brought about by distinct XML query types, we used a variety of XPath query types. A list of the XPath queries we used in our experiments is shown in **Table 4**.

6.1 Calculation of average tuning time and access time

By evaluating the access time and tuning time, we simply took into account the activities of one mobile client. This is so that the performance of XML querying at the other mobile clients is unaffected by the actions of one mobile client. By focusing on the activity of one client, the authors can evaluate the structure's performance in a controlled environment and obtain accurate and meaningful results.

The authors counted the number of buckets to establish the access time and tuning time for processing the XML queries. This approach allows them to evaluate the structure's efficiency in handling different query types based on the number of buckets required to process each query. By using this metric, the authors can determine the optimal number of buckets required for efficient data retrieval and processing.

Table 4. XPath queries employed in the tests

XML dataset Name	Query notation	XPath query expression
Mondial	MD1	/mondial/country/ethanicgroups
	MD2	/mondial/country/city/name
	MD3	/mondial/country/provinance/city/population[@year= "94"]
	MD4	/mondial/country/languages[text()="German"]
	MD5	/mondial/desert/located[@id="f0_157"]
	MD6	/mondial/country/city[name/text()="Belarus"]
	MD7	/mondial/country[name/text()="Austria"]/province/city/population
SigmodRecord	SR1	/SigmodRecord/issue/volume
	SR2	/SigmodRecord/issue/articles/article
	SR3	/SigmodRecord/issue/articles/article/authors
	SR4	/SigmodRecord/issue/articles/article/initPage[text()>"80"]
	SR5	/SigmodRecord/issue/articles/article/authors/author[@position="05"]
	SR6	/SigmodRecord/issue/articles/article[endPage/text()<"200"]
	SR7	/SigmodRecord/issue[number/text()>"4"]/articles/article/title
University Courses	UC1	/UniversityCourses/course/reg_num
	UC2	/UniversityCourses/course/titles
	UC3	/UniversityCourses/time/start_time
	UC4	/UniversityCourses/place/room
	UC5	/UniversityCourses/course/titles[text()= "Organic Chemistry I"]
	UC6	/UniversityCourses/course/place/room[@id= "121"]
	UC7	/UniversityCourses/course/subj[name/text()= "PSY "]/time/place

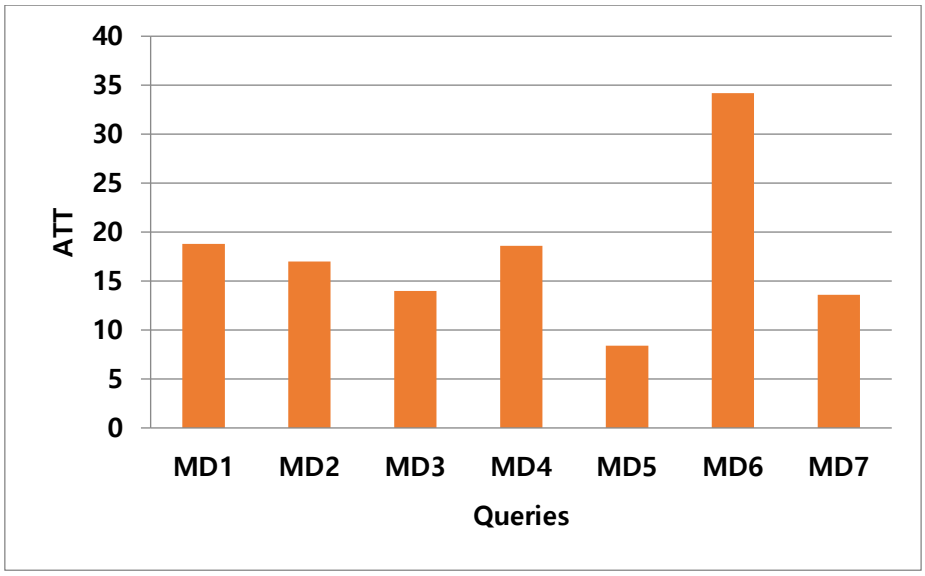

Fig. 6. Analysis of Average Tuning Time (ATT) on Mondial.

Fig. 6 to **Fig. 8** depict the tuning time of a mobile client to conduct various XML queries on various XML data sets: mondial, sigmodrecord, and university courses. In all the experiments, the first channel is used to broadcast the index information and others are used to broadcast data information. In our proposed indexing scheme, we have considered only the simple path XML queries. In the future, the twig pattern XML queries may be considered. Both our proposed indexing scheme performs well due to the multiple broadcast environment. In some queries, the proposed indexing stream is lacking due to the time taken by the encryption decryption. The inclusion of several indexes in our suggested structure allows the stream to pass over the useless material, which accounts for the size reduction. The types of XML queries, the size reduction of XML data sets, and the positions of accessible XML nodes inside the encrypted XML data stream, however, completely determine the increased value of tuning time.

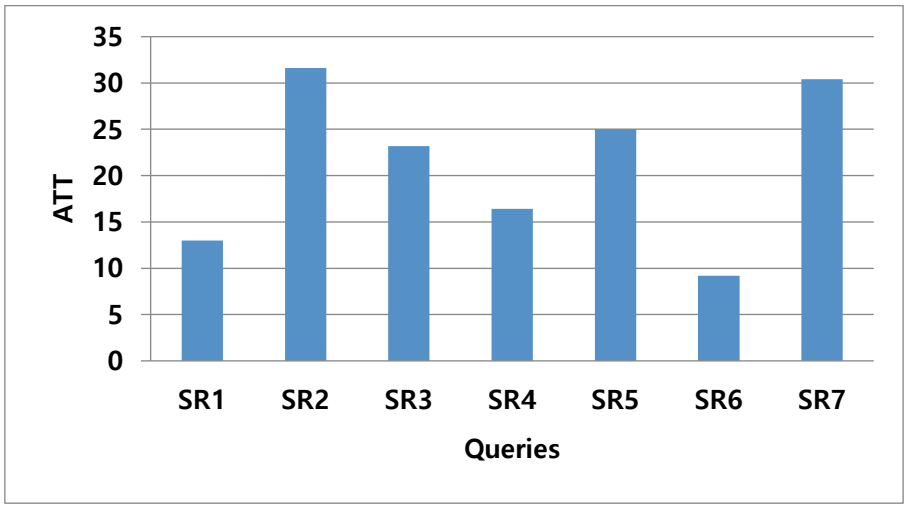

Fig. 7. Analysis of Average Tuning Time (ATT) on SigmodRecord.

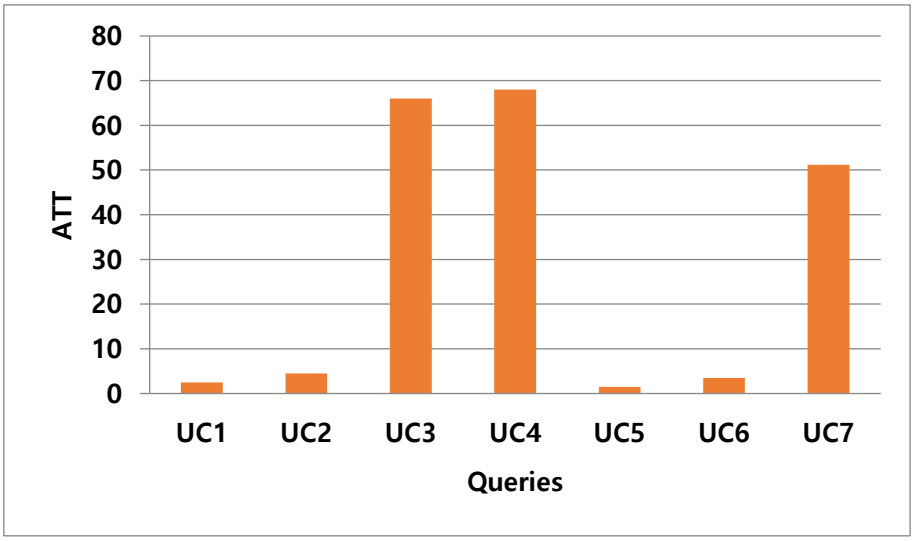

Fig. 8. Analysis of Average Tuning Time (ATT) on University Courses.

The average access time ratio for processing various XML queries on various data sets: mondial, sigmodrecord, and university courses are shown in **Fig. 9** to **Fig. 11**. In all the experiments, the first channel is used to broadcast the index information and others are used to broadcast data information. In our proposed indexing scheme, we have considered only the simple path XML queries. In the future, the twig pattern XML queries may be considered. Both our proposed indexing scheme performs well due to the multiple broadcast environment. In some queries, our proposed encrypted indexing scheme lacks due to the time taken by the encryption & decryption process. In our proposed encrypted indexing technique, the average access time for accessing various queried data in data sets has been calculated and analyzed. It means that mobile clients should have to wait less time to obtain the queried data from the server's broadcast channels. The types of XML queries, the characteristics of XML data sets, and the positions of accessible XML nodes inside the encrypted XML data stream, however, are wholly responsible for the increased value of access time. The XML data placement scheme initially prioritizes broadcasting XML nodes within root-to-node paths that contain substantial amounts of XML data. This prioritization is based on the rationale that these paths with extensive XML data warrant higher dissemination priority across broadcast channels. Hence, the retrieval time for the results of these specific XML queries is reduced compared to others, as the XML nodes associated with these queries are broadcasted earlier in the data dissemination process.

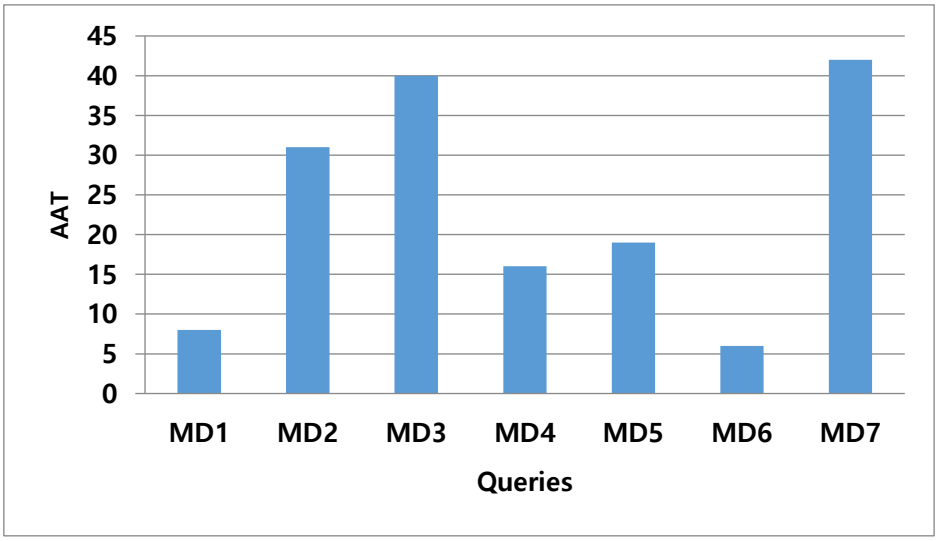

Fig. 9. Analysis of Average Access Time (AAT) on Mondial.

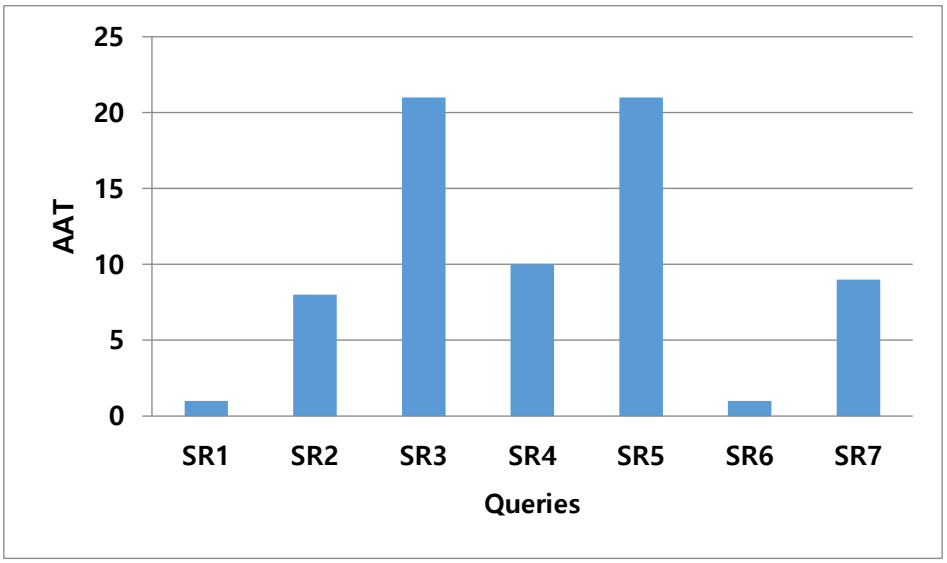

Fig. 10. Analysis of Average Access Time (AAT) on SigmodRecord.

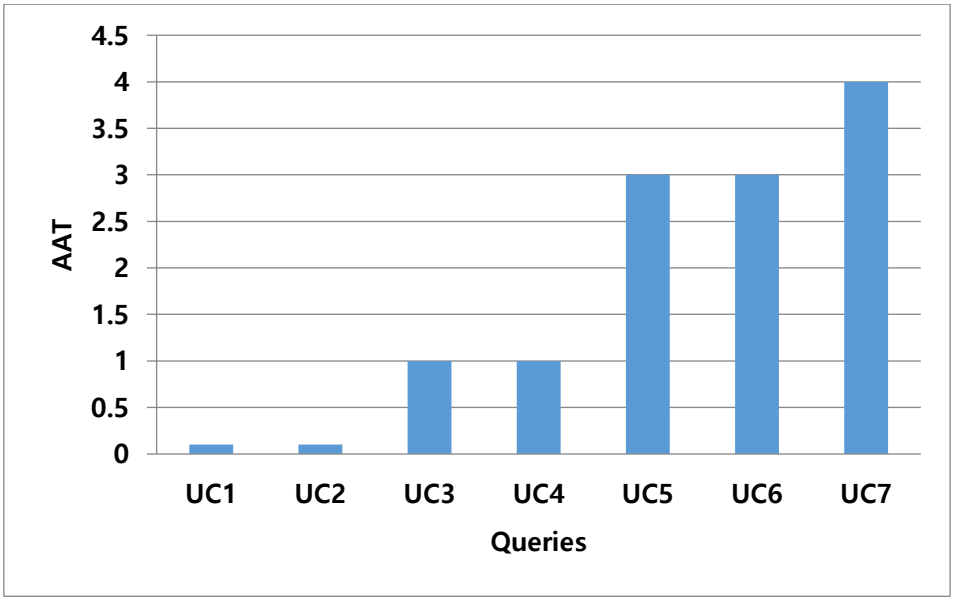

Fig. 11. Analysis of Average Access Time (AAT) on University Courses.

6.2 Assessment of Average Access Time (AAT) and Average Tuning Time (ATT) between proposed Encrypted Indexing Technique and Existing Techniques

Fig. 12 and **Fig. 13** display the findings of our comparison of the results obtained in **Table 5** between the proposed encrypted XML stream technique and other existing techniques, such as (1, X_m) replication, replication with PP, and replication with TP, XML indexing scheme proposed in [15], both (first & second) XML indexing schemes in [19] and XML indexing scheme in [16]. The indexing scheme in [19] is an XML indexing scheme with multiple broadcast channels. The indexing scheme in [16] is an XML indexing scheme with single channel. In our proposed indexing scheme as well as schemes in [16] & [19], we have considered only the simple path XML queries. In the future, the twig pattern XML queries may be considered. Both our proposed indexing scheme and indexing scheme in [19] perform well due to the multiple broadcast environments.

The average access time of our proposed encrypted indexing scheme outperforms the XML indexing schemes proposed in [15] and the first & second schemes proposed in [19]. However, the average access time is more than the XML indexing schemes proposed in [6], [9] & [16]. In some queries, our proposed encrypted indexing scheme lacks due to the time taken by the encryption & decryption process. In comparison to the existing XML indexing techniques [6], [9] & [19], it means that mobile clients should have to wait more time to obtain the needed data from the broadcast channels.

The average tuning time of the proposed encrypted indexing scheme outperforms all the indexing schemes except the indexing scheme proposed in [5]. However, the proposed scheme in [5] has not addressed the security concern. The data available on broadcast channels is prone to exposure. In our proposed encrypted indexing scheme, an encryption-decryption method is used to provide security in terms of authorization for the data available on broadcast channels. Today, the availability of high-configuration devices is very sound in terms of computation and memory. A little bit of delay in data accessing may be accepted as compared to the security and authority of the data.

Table 5. Assessment of Average Access Time and Average Tuning Time of proposed encrypted XML streaming Technique and existing XML indexing Technique

S. No.	Indexing Schemes	Average Access Time (AAT)	Average Tuning Time (ATT)
1	Proposed Encrypted Indexing Scheme	530924.88	39.9
2	(1, Xm) Replication with multiple channel Indexing scheme [6]	442437.4	31.5
3	Replication with PP Indexing scheme[9]	405999	1382
4	Replication with TP Indexing scheme[9]	406026	1371
5	XML indexing scheme in [15]	795000	1276
6	XML indexing scheme (first) in [19]	687500	1276
7	XML indexing scheme (second) in [19]	550000	1276
8	XML indexing scheme (second) in [16]	358400	490

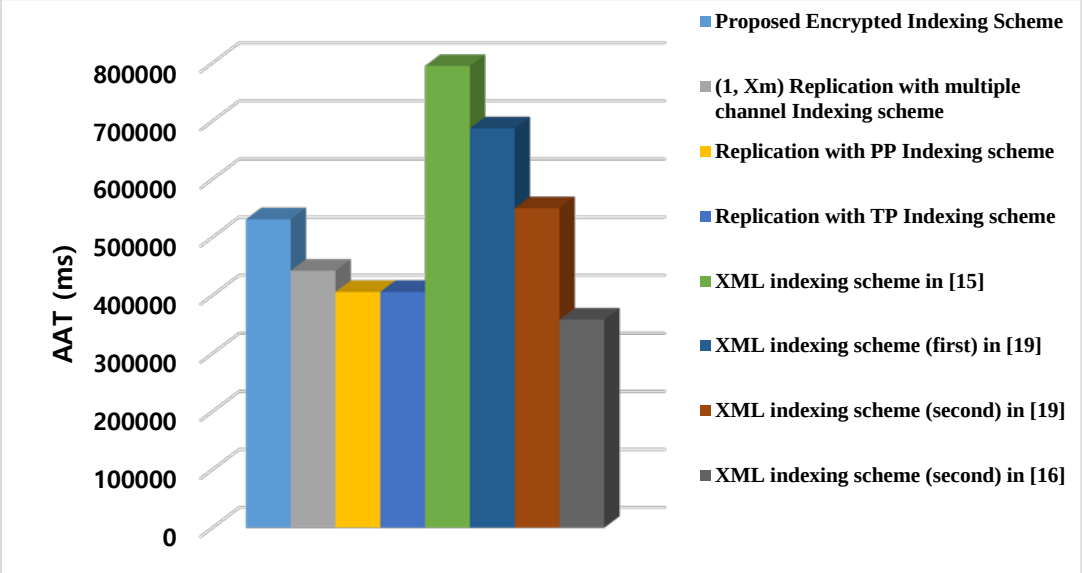

Fig. 12. Assessment of Average Access Time (AAT) of Proposed Encrypted Stream Technique and Existing XML Indexing Techniques.

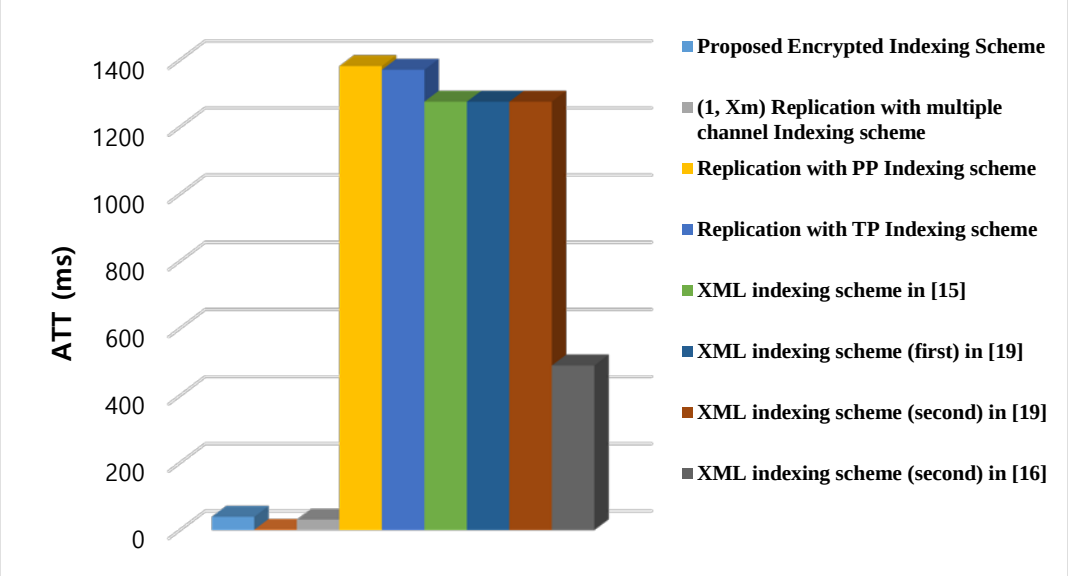

Fig. 13. Assessment of Average Tuning Time (ATT) of Proposed Encrypted Stream Technique and Existing XML Indexing Techniques.

7. Conclusion

Our work is focused on addressing the issue of secure indexing techniques for wireless broadcasting XML data. The authors have proposed a novel structure for encrypting and storing XML data securely and efficiently that can be broadcasted wirelessly and accessed by mobile devices. By focusing on the security and efficiency of the indexing techniques used for wireless broadcasting XML data, the authors aim to address a critical issue in mobile computing. While the suggested structure enables effective data retrieval and processing, the use of encryption techniques guarantees that the data is secure and accessible only to authorized persons.

We now know that the XML data has the characteristics of being semi-structured and encrypted. Together with advantages, this trait also creates a problem in that it might not use the conventional indexing techniques currently in use. By restricting the use of disc-based XML indexing approaches and proposing a bucket-based structure for wireless broadcasting XML data, the authors aim to provide a more efficient and secure solution for mobile computing. The proposed structure allows for efficient data retrieval and processing, while also ensuring the security of the data through the use of encryption techniques.

The index and data buckets are encrypted before being transmitted on numerous channels in the proposed secure indexing technique. Using the bandwidth in a very effective manner was an additional advantage of the multiple-channel setup. The index channel and data channel are used in the proposed encrypted XML streaming technique. The index channel has the encrypted index bucket, whereas the encrypted data bucket is broadcast on the data bucket.

According to the simulation findings, the suggested encrypted XML streaming technique performed better in terms of tuning time and access time than the current XML structure. This improvement can be attributed to the lengthier encrypted XML data stream in the existing XML format compared to the proposed one. The suggested method decreased the size of the encrypted XML data stream by utilizing a more effective bucket-based structure, which enhanced the effectiveness of data processing and retrieval.

The optimization of retrieval time for specific XML queries by leveraging a broadcasting mechanism in the data dissemination process. As a result, when users or applications submit these optimized queries, the required information is already available and can be retrieved more quickly, leading to a more efficient and responsive system. This optimization strategy may be added in the future to enhance the overall performance of XML data retrieval by prioritizing and proactively broadcasting the relevant nodes for specific queries.

Acknowledgments

This work was supported by the Dong-A University research fund.

References

[1] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler and F. Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C. November 26, 2008. https://www.w3.org/TR/xml/

[2] T.S. Chung, H.J. Kim, "Techniques for the evaluation of XML queries: a survey," *Data & Knowledge Engineering*, vol.46, no.2, pp.225-246, 2003. Article (CrossRef Link)

[3] V. Goel, A. K. Ahalawat, M. N. Gupta, "Distributed Air Indexing Scheme for Full-Text Search on Multiple Wireless Channel," in *Proc. of Intelligent Systems Technologies and Applications*, vol.385, pp.125-135, 2015. Article (CrossRef Link)

[4] V. Goel, A. K. Ahlawat, M. N. Gupta, "Partial index replicated and distributed scheme for full-text search on wireless broadcast," *Sādhanā*, vol.40, pp.2129-2142, 2015. Article (CrossRef Link)

[5] D. Gautam, and V. Goel, "XML Based Streaming Strategies for Indexing the Wireless Broadcast Data," in *Proc. of 2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE)*, pp.629-634, 2016. Article (CrossRef Link)

[6] V. Goel, D. Gautam, A. Gupta, and S. Kumar, "An improvised indexing technique for XML data over multiple channels in wireless environment: (1, Xm) method," *International Journal of Communication Systems*, vol.32, no.16, 2019. Article (CrossRef Link)

[7] C.S. Park, C.S. Kim, Y.D. Chung, "Efficient Stream Organization for Wireless Broadcasting of XML Data," in *Proc. of Advances in Computer Science - ASIAN 2005. Data Management on the Web*, vol.3818, pp.223-235, 2005. Article (CrossRef Link)

[8] S.H. Park, J.H. Choi, S. Lee, "An Effective, Efficient XML Data Broadcasting Method in a Mobile Wireless Network," in *Proc. of Database and Expert Systems Applications (DEXA 2006)*, vol.4080, pp.358-367, 2006. Article (CrossRef Link)

[9] J.P. Park, C.S. Park, Y.D. Chung, "Energy and Latency Efficient Access of Wireless XML Stream," *Journal of Database Management*, vol.21, no.1, pp.58-79, 2010. Article (CrossRef Link)

[10] J.P. Park, C.S. Park, Y.D. Chung, "Lineage Encoding: An Efficient Wireless XML Streaming Supporting Twig Pattern Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol.25, no.7, pp.1559-1573, 2013. Article (CrossRef Link)

[11] M. Mirabi, H. Ibrahim, L. Fathi, "PS+Pre/Post: A novel structure and access mechanism for wireless XML stream supporting Twig pattern queries," *Pervasive and Mobile Computing*, vol.15, pp.3-25, 2014. Article (CrossRef Link)

[12] A. B. Boroujeni, M. Mirabi, "A Novel Replication Strategy for Efficient XML Data Broadcast in Wireless Mobile Networks," *Journal of Information Science & Engineering*, vol.32, no.2, pp.309-327, 2016. Article (CrossRef Link)

[13] M. Javani, M. Mirabi, "An Efficient Index and Data Distribution Scheme for XML Data Broadcast in Mobile Wireless Networks," *Journal of Information Science and Engineering*, vol.33, pp.159-182, 2017. Article(CrossRefLink)

[14] Y. Qin, Q. Z. Sheng, H. Wang, N.J.G Falkner, "Organizing XML Data in a Wireless Broadcast System by Exploiting Structural Similarity," *Wireless Personal Communications*, vol.98, pp.1299-1329, 2018. Article (CrossRef Link)
[15] H. Mirabi, M. Mirabi, A.B. Boroujeni, "An Efficient XML Data Placement Scheme over Multiple Wireless Broadcast Channels," *Journal of Information Science and Engineering*, vol.32, pp.1183-1203, 2016. Article (CrossRef Link)
[16] M. Shekarriz, S.M. Babamir, M. Mirabi, "Query processing optimization in broadcasting XML data in mobile communications," *The Journal of Supercomputing*, vol.77, no.6, pp.5354-5380, 2021. Article (CrossRef Link)
[17] http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html#sigmod-record (accessed on 09-Feb-23).
[18] M. Shokri, and M. Mirabi, "An efficient stream structure for broadcasting the encrypted XML data in mobile wireless broadcast channels," *The Journal of Supercomputing*, vol.75, pp.7147-7173, 2019. Article (CrossRef Link)
[19] S. F. Ozonbolagh, and M. Mirabi, "Efficient XML data placement schemes over multiple mobile wireless broadcast channels," *The Journal of Supercomputing*, vol.78, no.1, pp.168-199, 2022. Article (CrossRef Link)
[20] M. Gopianand, P. Jaganathan, "An effective quality analysis of XML web data using hybrid clustering and classification approach," *Soft Computing*, vol.24, pp.2139-2150, 2020. Article (CrossRef Link)
[21] S. Almajali, G. Al-Naymat, and A. Atiyah, "Bpoint-tree: An Indexing Structure for Efficient Search in Data Retrieval," in *Proc. of 2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA)*, pp.1-6, 2018. Article (CrossRef Link)
[22] M. Kvet and J. Papan, "The Complexity of the Data Retrieval Process Using the Proposed Index Extension," *IEEE Access*, vol.10, pp.46187-46213, 2022. Article (CrossRef Link)
[23] V.N. Waghmare, and R.C. Thool, "Implementation and Verification of Dissemination of Tree Structure Data using Signature Scheme for XML Data," *International Journal of Database Theory and Application*, vol.8, no.4, pp.215-230, 2015. Article (CrossRef Link)
[24] P. Liu, W. Sun, J. Zhang, and B. Zheng, "An automaton-based index scheme supporting twig queries for on-demand XML data broadcast," *Journal of Parallel and Distributed Computing*, vol. 86, pp.82-97, 2015. Article (CrossRef Link)
[25] B. Safabahar, and M. Mirabi, "A new structure and access mechanism for secure and efficient XML data broadcast in mobile wireless networks," *Journal of Systems and Software*, vol.125, pp.119-132, 2017. Article (CrossRef Link)

Mr. Vinay Kumar Ahlawat holds MCA and M.Phil degrees in Computer Science & Engineering and is currently pursuing a Ph.D. in the same field. He is the Associate Director of the Centre for Distance and Online Education at Chandigarh University, India. With over 22 years of experience in both industry and academia, Mr. Ahlawat has worked with various premium organizations. His expertise includes the analysis, design, and development of eLearning modules, multimedia content generation, and digital branding and promotions. Mr. Ahlawat has an extensive publication record, including over 10 books, 5 research papers in journals and international conferences, 1 patent, and multiple research book chapters in Scopus Indexed journals. His research interests encompass mobile computing, wireless computing, indexing of data broadcasting in mobile devices, distributed computing, wireless broadcasting, and human-computer interaction.

Dr. Gaurav Agarwal is currently working as the Associate Professor and Head of the Department in the Department of Computer Science and Engineering at Invertis University, Bareilly, Uttar Pradesh, India. Dr. Agarwal earned his B.Tech., M.Tech., and Ph.D. in Computer Science and Engineering from prestigious institutions in India. Dr. Agarwal began his academic career in 2006 at Northern India Engineering College, Delhi, India. In 2007, he joined Invertis University in Bareilly as a faculty member in the Department of Computer Science and Engineering. Over the course of his 17-year teaching career, Dr. Agarwal has published over 65 research articles in various national and international journals and conferences, many of which are indexed in SCOPUS and SCI. Under his guidance, three research scholars have successfully completed their Ph.D. degrees, and he is currently mentoring six Ph.D. scholars. Dr. Agarwal's research interests include Cryptography and Network Security, Cloud Computing, and Multimedia Systems. He is also an active member of several academic societies, including He is the member of various academic societies like IAENG, ISTE, IACR, IFREP etc.

Dr. Vikas Goel received B.Tech. degree in Information Technology, M.Tech. degrees in computer science & engineering and Ph.D. in computer science & engineering. He is currently Professor & Addl. HoD in the IT Department at KIET Group of Institutions, Ghaziabad, India. He has a vast experience of 22+ years of teaching in various premier institutes. Under his guidance, three research scholars have successfully completed their Ph.D. degrees. He has published more than 60 research papers in SCI-indexed journals, Scopus-indexed journals, and international conferences like IEEE, Springer, ACM, and Elsevier. He has published 7 patents and various research book chapters in Scopus Indexed journal. His research interests include mobile computing, wireless computing, indexing of data broadcasting in mobile devices, distributed computing, sentiment analysis, and Blockchain technology.

Kueh Lee Hui is working as an assistant professor at the department of Electrical Engineering, Dong-A University since 2012. She completed her PhD Degrees from Department of Electrical Engineering, Dong-A University, Korea. In 2009 she completed her BS degree in Electronic and Communication, Department of Electronic Engineering, University Malaysia of Sarawak, Malaysia. She also has done MS in 2007 from Malaysia. Her research interests are image processing, face recognition, digital image forensic, intelligent control and control application, power system.

Mangal Sain received the M.Sc. degree in computer application from India, in 2003, and the Ph.D. degree in computer science, in 2011. Since 2012, he has been an Assistant Professor with the Department of Computer Engineering, Dongseo University, South Korea. He has authored over 100 international publications, including journals and international conferences. His research interests include wireless sensor networks, cloud computing, the Internet of Things, embedded systems, and middleware