Research Article

An XML Privacy-Preserving Data Disclosure Decision Scheme

Lihong Guo and Haitao Wu

Department of Information and Communications Engineering, Nanjing Institute of Technology, Nanjing, 211167, China

Correspondence should be addressed to Lihong Guo; guolihongnj@163.com

Received 29 September 2021; Revised 5 January 2022; Accepted 15 January 2022; Published 24 February 2022

Academic Editor: Helena Rifa-Pous

Copyright © 2022 Lihong Guo and Haitao Wu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In order to protect the sensitive data represented as XML documents in a trusted collaborative system where sensitive data are not shared, an XML privacy-preserving data disclosure decision scheme was proposed under the assumption of a trusted server. This scheme is inspired by the idea of separating storage structure and content. Temporary access matrix is used to represent structure authorization and the vector represents the content authorization of leaf node. According to the conversion rules, access matrix not only represents access authorization of all nodes but also keeps the main structure of the XML document. With the combination of the vector and matrix, it can provide different access views for different group users with different purposes. In addition, start-end encoding is used to encode all the nodes for locating nodes and the content; privilege matrix solves the problem of privacy synchronization change for all users. At the same time, authentication polynomials are used to verify different users and improve the security level. The experimental results show that the scheme not only effectively protects XML sensitive data but also reduces the storage pressure on the server side; at the same time, from the response time, we know that it is beneficial for the rapid search and information positioning.

1. Introduction

In 2020, Internet celebrated its 51st birth anniversary. After decades of rapid development, Internet has become an indispensable part of our lives. It has greatly changed the way of our lives and the way of thinking, and it has brought huge benefits to mankind enhancing many aspects of life. However, with the rapid development of network technology and the widespread popularity of mobile smart terminals, the demand for data business on the Internet is growing exponentially, which leads to the development of the network in the direction of ubiquitous, scene-based, and intelligent [1, 2].

There is no doubt that the cloud platform is the most suitable information sharing platform for storage, management, and data processing on the Internet. In 2017, 85% of the enterprises and institutions have implemented cloud strategy [3]. At present, the shared application model is generally divided into four categories [4]:

1.1. Model 1: Single User and Single Server Mode. In this model, only one user uploads or searches information from the server. It does not deal with the user's authorization, so it is very simple. In most cases, this model is suitable to store data in a private cloud server for companies and it is convenient and safe to query information. The model is shown in Figure 1(a).

1.2. Model 2: Multiple-to-One Single Server Mode. This is a typical share mode, where many users provide share information to the server, and the server is responsible for collecting and storing information. However, this kind of information sharing is limited, which needs the server to filter some sensitive data based on the user's privacy policy, so in this model, the user is not only the sender of information but also the inquirer of information. Therefore, the user's authorization is its core problem, which includes access and upload. The model is shown in Figure 1(b).

FIGURE 1: (a) Model 1: Single user and single server mode. (b) Model 2: Multiple-to-one single server mode. (c) Model 3: One-to-multiple single server mode. (d) Model 4: Multi-user single server mode.

1.3. Model 3: One-to-Multiple Single Server Mode. In this model, there are many senders that send shared information to the server, the sender and receiver are different entities, so the sender has secret or sensitive information to protect, which only needs the server to filter information based on the sender's privacy policy. From the view of privacy protection, this mode is simple and does not need to consider the authorization problem of multi-users. The detailed model is shown in Figure 1(c).

1.4. Model 4: Multi-User and Single Server Mode. In this model, one sender is responsible for uploading the shared information, and many users with different access purposes want to get the information. In order to protect the privacy of data in shared information, the server has to filter the privacy. Faced with so many different users, different access views need to be provided for users with different access purposes, to ensure that data are shared without compromising on the privacy aspect. The detailed model is shown in Figure 1(d).

In this paper, we take model 4 as the application model, and take semistructured XML data for storage, exchange of data, and information representation, to study all kinds of privacy protection schemes. In this sharing mode, the sender is the owner or the authorized agent of XML data, who wants to share some useful information for other users on the premise of ensuring the privacy of data.

According to the application scenario, we provide several different XML access authorization schemes. At the same time, based on the structural characteristics of the XML document and by means of various mathematical models and theories (including arrays, matrices, polynomials, etc.), almost all the privacy-preserving schemes are

transformed into mathematical models to solve, which not only greatly protects the sensitive data but also improves the speed of authorization query.

In summary, the contributions of this paper are as follows:

- (1) An XML privacy-preserving data disclosure decision scheme was proposed based on several authorization models and security mechanisms.
- (2) Provides the process of design and the detailed steps, including the creation of storage matrix, the vector authorization of leaf nodes, the identity authentication of users, the encoding of the node, etc.
- (3) In order to test the effectiveness of the scheme, a series of experiments was designed to verify the scheme. Experiment results show the performance of this scheme; at the same time, security analysis and verification are also provided.

The rest of the paper is organized as follows. Section 2 reviews the related work. In section 3, we classify and present some typical access authorization schemes. In section 4, from the view of the mathematical model, we analyze and compare several authorization schemes, and at the same time, further describe the procedure of the integration scheme. Section shows the simulation experiment and the test result, which verifies the performance of the scheme. Finally, we conclude in section 5.

2. Related Work

At present, many XML privacy-preserving authorization schemes have been proposed, according to the granularity level, and are divided into three categories: view-based

authorization, structure-based authorization, and content-based authorization. View-based authorization is for XML, structure-based authorization is for DTD (Document Type Definition) or XML schema, and content-based authorization is for leaf node.

2.1. View-Based Privacy-Preserving Schemes. In the literature [5–7], many privacy-preserving schemes were proposed. Jae-Gil proposed the first Hippocratic XML model based on the ancestor's authorization, and it solved the problem of finding the nearest ancestor nodes' authorization, and proposed an efficient access control method that used authorization index and the nearest neighbor search. View-based access control mechanism created and maintained a separate view for different group users, which contained the exact set of data elements authorized to access. Later, the access control method for XML documents in the workflow environment was presented, and access control policies of the workflow system are based on the RBAC model. There are two types of access control policies for every element: permit or deny.

For example, we assume that a company stores its entire core data with XML documents in the cloud server, which is convenient to access and share information. However, to protect data privacy, it must ensure that the user with different purposes can get different access views. This means that general manager can see all the information, technical staff cannot obtain financial tables from the financial department, and different users have different access views. This is shown in Figure 2.

In Figure 2, the left panel represents the entire XML document tree, node is represented by the circle, for example, A, B, C, ..., H, I, J. In those nodes, A, C, D, E are intermediate nodes, and B, F, G, H, I, J are leaf nodes. A node's content is represented by a box, for example, the content of node B is "xxx," and the content of node J is "ccc," and only leaf node has the content. The middle panel is the restricted XML document tree; for this group user, sensitive nodes D, F, and node G are deleted from the original tree. The right panel is another view for the group user, and node G and node J are the sensitive nodes. In this scheme, it provides different access views for different users. Therefore, the scheme suffered from high maintenance and storage costs, especially for a large number of different group users.

2.2. Structure-Based Privacy-Preserving Schemes. DTD and XML schema are two kinds of documents that define XML's structure. We can define legitimate and standard XML documents under their constraints. From DTD or XML schema, we can know about the framework of XML documents. At present, many authorization schemes are based on DTD or XML schema [8–11]; in these schemes, authorization is reflected in DTD or XML schema, so the search of authorization nodes is easy and fast. In addition, the storage

of DTD and XML schema is lightweight, which reduces the burden on the server. In DTD, it adds a symbol "+" or "-" to define the permit or deny after the element name. In XML schema, it uses an attribute to represent the authorization.

In addition, the literature [12] designed a language for specifying access control on XML documents. The update operations in this model are based on the W3C XQuery specification. Alternative language that supports access control annotations at the level of DTD is also presented.

For example, corresponding to the middle panel in Figure 2, the same structure-based authorization is shown in Figure 3. Figure 3(a) is the definition of DTD authorization, and Figure 3(b) is the authorization of XML schema. In both of the two authorization modes, the default authorization is the permit, which means that the default value of all the nodes is "permit." Symbol "+" denotes permit and symbol "-" denotes deny for some nodes in DTD. Similarly, the attribute of node ac = "yes" denotes permit and ac = "no" denotes deny in XML schema.

Although this mode is easy to achieve, structure-based authorization also has some disadvantages. Because DTD or XML schema do not always exist, sometimes these schemes cannot be used in the absence of DTD or XML schema. On the other hand, this approach is based on the structure, so this representation is relatively coarse.

2.3. Content-Based Privacy-Preserving Schemes. In the literature [13], Angela put forward a privacy model called P4A (Privacy for All) to capture collector's privacy practice and data providers' privacy preferences. In this model, a privacy policy considered two major elements: the data and the purpose of usage. This model offered more flexibility than current approaches, in that it allowed unconditional and conditional access. On the other hand, it aims for leaf nodes' access control and thinks that the leaf node is the core information, so the authorization of the leaf node is the main problem. However, this model only considered the leaf node's authorization and ignored the protection of the XML structure.

In addition, similar to P4A, the authorization index vector was proposed to represent the leaf node's authorization. In the index vector, the core task is to specify access control of the leaf node. Value 1 indicates that this leaf node is allowed to access, and value 0 means that this leaf node is not allowed to access. In order to speed up the search, the query request is also converted into a vector, and the creation of the query vector is also based on the leaf node information. The user first chooses each node that he or she wants to get and then filters through the authorization index vector. For example, in the middle panel and right panel of Figure 2, their respective authorization index vectors are index vector 1 and index vector 2, as shown in Figure 4. From the index vector, we can know that the leaf nodes are the core protection objects. This is a lightweight authorization scheme.

FIGURE 2: View-based authorization.

FIGURE 3: (a). DTD authorization. (b)XML schema authorization.

Leaf node info	B	F	G	H	I	J
Authorization index vector 1	1	0	0	1	1	1
Authorization index vector 2	1	1	0	1	1	0

FIGURE 4: Content-based authorization.

3. XML Privacy-Preserving Data Disclosure Decision Scheme

Based on the abovementioned privacy-preserving authorization schemes about XML documents, the authors know that as far as an XML document is concerned, its tree structure indicates that the XML structure contains the outline of information, and the leaf node's content is the core of the information. Therefore, how to select the appropriate authorization scheme depends on the application

requirements. If the purpose is to classify or search the outline information, it should focus on the structure authorization, and if the purpose is to obtain detailed information, the authors should care about content authorization. However, no matter whichever scheme, one of the purposes is to save as little information as possible under the premise of protecting sensitive data and the other is to be able to carry out the search of authorization quickly. Therefore, we consider combining the authorization scheme with mathematical theory, which can speed up information

query and facilitate to locate the nodes. Based on this, a privacy protection data disclosure decision scheme was proposed. The detailed infrastructure and procedure are given below. For the convenience of description, in this paper, the content refers to the content of the leaf node.

3.1. XML Access Control List. ACL (Access Control List) is a list of permissions associated with an object. It describes the access rights of an object for a list of subjects [14]. It explores the means of granting/rejecting access with particular rights (such as read, write, or execute) to subjects on certain objects. $ACL=(S, O, A)$; S is the set of subjects, O is the set of objects, and A is the access matrix. $A[s,o]$ is the operation about a subject to an object. An access matrix is a table that maps the permissions of a set of subjects to act upon a set of objects within a system. The matrix is a two-dimensional table with subjects down the columns and objects across the rows. The permissions of the subject to act upon a particular object are found in the cell that maps the subject to that object.

The example of ACL is shown in Table 1. It is the corresponding ACL for Figure 2. The first row represents access to all nodes, and it corresponds to the left panel in Figure 2, S_1 is the super user. The second row is the restricted access, it corresponds to the middle panel in Figure 2, and the user S_2 does not have the right to access nodes D, F, and G. The third row is the restricted access, it corresponds to right panel in Figure 2, and user S_3 does not have the right to access nodes G and J. The ACL table represents all users' authorization for all the nodes (including middle nodes and leaf nodes).

In the ACL table, all users' authorization is saved in an access matrix, just like Table 1. Although ACL can represent the authorizations of users for all the nodes including intermediate nodes and leaf nodes, it destroys the XML's tree structure and cannot indicate the XML's structure. So, the difficulty of ACL is in restoring XML's tree structure. But in ACL, this kind of access matrix is a very good way to map the authorization. Inspired by this, the authors take into account the necessity of finding an authorization method to use the access matrix, and it not only indicates the user's authorization but also represents the XML tree structure of the legitimate user. So in the next step, the authors design the authorization matrix.

3.2. Creating Privacy-Preserving Matrix. Access Control Matrix was formalized to help accurately describe the protection state. This simple model reflected the access control logic. Since then, the concept has been expanded and has morphed into various other access control models that can handle complex access control logic such as state dependent access control and hierarchical access [15, 16].

An access matrix represents the set of authorizations defined at a given time in the system. The access matrix model provides a framework for describing discretionary access control policies. First proposed by Lampson for the protection of resources within the context of operating systems, and later refined by Graham and Denning, the

TABLE 1: Access control list.

Node User	A	B	C	D	E	F	G	H	I	J
S_1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
S_2	Y	Y	Y	N	Y	N	N	Y	Y	Y
S_3	Y	Y	Y	Y	Y	Y	N	Y	Y	N

model was subsequently formalized by Harrison, Ruzzo, and Ullmann (HRU model), who developed the access control model proposed by Lampson for analyzing the complexity of determining an access control policy. The original model is called access matrix since the authorization state, meaning the authorization holding at a given time in the system, is represented as a matrix.

In this paper, the authors propose a method which uses an access matrix to save the authorizations of nodes, and at the same time, the access matrix can represent the XML's tree structure.

Definition 1. StrucTree. StrucTree is a convenient structure summary of the XML document. It describes every unique label path of a source exactly once, regardless of the number of times it appears in that source document.

Definition 2. StrucMatrix. StrucMatrix is a lower triangular matrix, and it is an authorization storage matrix for StrucTree. If all the nodes in StrucTree are permitted to access, then its corresponding StrucMatrix is a global matrix, and StrucTree is a global structure tree.

Definition 3. Storage rules. The storage rules of StrucTree in matrix are: (1) for the leaf nodes, they are saved sequentially in the diagonal from left to right and top to down; (2) for the intermediate nodes, they are saved in the lower left corner of all its children's position. Select the maximum value of the abscissa of all child nodes as the abscissa value of this node and select the minimum value of the ordinate of all child nodes as the ordinate value of this node [17].

From Definition 3, the authors can know that the number of leaf nodes in StrucTree decides the dimension of StrucMatrix, and leaf nodes of StrucTree are stored sequentially in the main diagonal. The intermediate nodes are saved in the lower left corner of all its children position.

The authors assume that left panel in Figure 2 is the corresponding StrucTree for some XML document. By this way, the authors can save StrucTree of the XML document in a matrix. Figure 5 displays the StrucMatrix corresponding to Figure 2.

- (1) In StrucTree, the number of leaf nodes (node B, F, G, H, I, J) is 6, based on the storage rules, the dimension of StrucMatrix is 6×6 . Leaf node "B" is the first leaf node, so it is saved in $[0, 0]$, node "F" (1, 1) is saved in $[1, \dots]$. Leaf node "J" is saved in $[5]$.
- (2) With regard to the intermediate node "D," it has two children nodes "F" and "G." We have known that "F" is saved in $[1]$, and "G" is saved in $[2]$. Assume that node "D" is saved in $[x, y]$, then based on the rules, we can calculate that $x = \text{maximum } (1, (2) = 2,$

$$\begin{bmatrix} B & & & & & \\ 0 & F & & & & \\ 0 & D & G & & & \\ 0 & 0 & 0 & H & & \\ 0 & 0 & 0 & 0 & I & \\ A & C & 0 & E & 0 & J \end{bmatrix}_{6 \times 6} \quad
\begin{bmatrix} B & & & & & \\ 0 & 0 & & & & \\ 0 & 0 & 0 & & & \\ 0 & 0 & 0 & H & & \\ 0 & 0 & 0 & 0 & I & \\ A & C & 0 & E & 0 & J \end{bmatrix}_{6 \times 6} \quad
\begin{bmatrix} B & & & & & \\ 0 & F & & & & \\ 0 & D & 0 & & & \\ 0 & 0 & 0 & H & & \\ 0 & 0 & 0 & 0 & I & \\ A & C & 0 & E & 0 & 0 \end{bmatrix}_{6 \times 6}$$

FIGURE 5: Corresponding StrucMatrix for Figure 2.

$y = \text{minimum}(1, (2)) = 1$. Hence, “D” is saved in position [1, 2] in matrix. Same rules used in node “E,” it is saved in position [3, 5]. As to node “C,” its child node “D” is in [1, 2], and another child node “E” is in [3, 5], so node “C” is saved in $[x, y]$, where $x = \text{maximum}(2, (5)) = 5$, $y = \text{minimum}(1, (3)) = 1$, so node “C” is in [1, 5].

3.3. Xml Privacy-Preserving Controlled By System Privileges. XML authorization matrix well reflects the authorization of the structure. Thus, the user with different access purpose has a different authorization matrix. But, privacy is a changing thing; information viewed as privacy for some time is later made public. Similarly, today, the information is public, a few days later, it can turn into private. Just like the authorization of a system, if some nodes’ authorization changes, then the authors can define the system’s privilege matrix; it can affect all the users’ authorization.

Assume that the system’s privilege matrix is A; currently, this system has two users, U_1 and U_2 . Original authorization matrix B belongs to the user U_1 , and original authorization matrix D belongs to the user U_2 . After filtering through the system privilege matrix, their final authorization matrixes are C and E.

As time goes by, the system privacy is changing, and the privacy of some nodes can be made public, so the system’s privilege matrix changed, and the original A changed to new matrix A’. Because matrix A is changed, the users’ authorization matrix is also updated as well.

The changing procedure is shown below. Here, “*” represents the operation. If some nodes’ privilege has changed, then we can change their corresponding values in privilege matrix A. By this way, it is easy to control the system’s public privilege and easily works on every authorization user.

If the privilege value is “+,” which means this node is permitted to access, then all users’ corresponding values are accessible. If the privilege value is “-,” which means this node is denied access, then all users’ corresponding values are 0.

Assume that $\text{Self}[i, j]$ is an user’s authorization value in matrix $[i, j]$. If this system has the privilege matrix, then it needs to adjust this user’s authorization, and the last authorization matrix is decided by the formula (1). Here,

$\text{StrucMatrix}[i, j]$ is the corresponding value node $[i, j]$ in the global authorization matrix.

$$R[i, j] = \begin{cases} \text{StrucMatrix}[i, j] & \text{if } A[i, j] = + \\ 0 & \text{if } A[i, j] = - \\ \text{Self}[i, j] & \text{if } A[i, j] = 0 \end{cases} \quad (1)$$

If the privilege value is “+,” which means this node is permitted to access, so all users’ corresponding values are accessible. If the privilege value is “-,” which means this node is denied access, so all users’ corresponding values are 0. If the privilege value is “0,” which means that this node’s authorization depends on its self-authorization, and there is no need to change.

For example, the user U_1 with authorization matrix B, under the joint operation with privileged matrix A, his or her final authorization matrix is matrix C.

$$\begin{aligned}
A &= \begin{bmatrix} 0 \\ 0 + \\ 0 + + \\ 0 0 0 0 \\ 0 0 0 0 - \\ 0 0 0 0 0 0 \end{bmatrix}, \\
B &= \begin{bmatrix} B \\ 0 0 \\ 0 0 0 \\ 0 0 0 H \\ 0 0 0 0 I \\ A C 0 E 0 J \end{bmatrix}, \\
C &= A * B = \begin{bmatrix} B \\ 0 F \\ 0 D G \\ 0 0 0 H \\ 0 0 0 0 0 \\ A C 0 E 0 J \end{bmatrix}.
\end{aligned} \quad (2)$$

For the user U_2 with authorization matrix D , under the joint operation with privileged matrix A , his or her final authorization matrix is matrix E .

$$\begin{aligned}
 A &= \begin{bmatrix} 0 \\ 0 + \\ 0 + + \\ 0 0 0 0 \\ 0 0 0 0 - \\ 0 0 0 0 0 0 \end{bmatrix}, \\
 D &= \begin{bmatrix} B \\ 0 F \\ 0 D 0 \\ 0 0 0 H \\ 0 0 0 0 I \\ A C 0 E 0 0 \end{bmatrix}, \\
 E = A * D &= \begin{bmatrix} B \\ 0 F \\ 0 D G \\ 0 0 0 H \\ 0 0 0 0 0 \\ A C 0 E 0 0 \end{bmatrix}.
 \end{aligned} \tag{3}$$

3.4. Generating Privacy-Preserving Vectors. In the above authorization matrix, the authorization matrix for the group user maintains the original tree structure and represents access authorization of all nodes. In the server, the authors must save all the different authorization matrices, when a different legitimate user accesses the server, the system should provide its corresponding authorization matrix. In fact, all the authorization matrices are similar, because they are clipped from the same global authorization matrix. Therefore, the authors suppose that if the system only saves one global authorization matrix and the user's basic information (for example, ID, authorization parameters, and so on), then by these information, the system can generate temporary authorization matrix for the users. By this way, it not only can reduce the burden of server but also could improve the security of the system due to the temporary matrix [18].

Inspired by this view, a method using the vector to represent authorization parameters was proposed. In the vector, every component of the vector represents the address of the content. The content is the core information, so if someone gets the content's address, which means that he or she can get the content and obtain the detailed data. So in this case, the content address is the important part.

Definition 4. Authorization Index Vector. The index vector $a = (a_0, a_1, a_2, \dots, a_n)$ is used to represent authorization information for the XML document, the dimension of the

vector is the number n of leaf nodes in StrucTree, every vector's subcomponent represents the authorization of each leaf node. For the XML document, the content is the core, so the authorization of the leaf node can well represent the authorization of the entire XML document.

For example, in the left panel in Figure 2, the number of leaf nodes is 6, from left to right and from top to bottom. "B" is the first leaf node, whose content's address is the coefficient a_0 . "F" is the second leaf node, whose content's address is the second coefficient a_1 , and so on. The number of leaf nodes decides the dimension of vectors. So the global vector is $(a_0, a_1, a_2, a_3, a_4, a_5)$; for the middle panel in Figure 2, its corresponding vector is $(a_0, 0, 0, a_3, a_4, a_5)$; for the unauthorized node, its vector value is 0. For the right panel in Figure 2, its vector is $(a_0, a_1, 0, a_3, a_4, 0)$. Given below is the authorization of the leaf node and its authorization index vector according to Figure 2.

$$L_0 = (B, F, G, H, I, J), V_0 = (a_0, a_1, a_2, a_3, a_4, a_5).$$

$$M_1 = (B, 0, 0, H, I, J), V_1 = (a_0, 0, 0, a_3, a_4, a_5).$$

$$R_2 = (B, F, 0, H, I, 0), V_2 = (a_0, a_1, 0, a_3, a_4, 0).$$

In the use of the XML document, it has many repeated nodes with the same structure, although the repeated nodes occur only once in StrucTree, and save in storage matrix. So in order to protect the content and link all the structure and content information, the authors use start-end encoding to encode all the nodes, and take the encoding of the leaf node as the content's address. Thus, in the processing of the search, the authors can calculate other repeated leaf node content encoding based on the first leaf node encoding in StrucTree. The authors assume that $R[0]$ is the first repeated node, its encoding is (p, q) . The repeated node $R[i]$'s encoding is labeled (m, n) , then its calculation formula is present in formula (2) [8].

$$R[i](m, n): \begin{cases} m = p + (q - p + 1) * i \\ n = m + (q - p) \end{cases}. \tag{4}$$

In this formula, The authors $= 0, 1, 2, 3, \dots$, it is the order of a repeated node. A detailed start-end encoding is shown in the literature.

In this encoding method, the authors take the leaf node encoding as the content's address. With the help of node encoding, it is easy to find the content of the corresponding node. In the server, the content stored is unordered, even if the attacker gets the content, he or she cannot link other information and get meaningful information, so it is meaningless.

3.5. Identity Authentication. In order to improve the security level, the authors design a kind of two-pass identity authentication method; it is related to the user's Id and its leaf node authorization vector, and the authors name it as authentication polynomials.

In this scheme, the authors take polynomials $f(x)$ as the authentication function; in this function, x is the user's Id and the parameters of polynomials are leaf nodes' addresses,

which are from the authorization vector. The identification process is described in Table 2

- (i) First, a user must match his or her username and password. After the first pass certification, he or she obtains his or her user id and authorization vector.
- (ii) Second, the system will calculate this user's authentication value, by his or her user id and authorization vector, and calculate the value of polynomial $f(x)$; in this $f(x)$, x is the user's id, and $a_0, a_1, \dots$ are the coefficients from his or her authorization vector.
- (iii) Finally, if the value of polynomial $f(x)$ is equal to the certified value, which is saved in Table 2, he or she will pass the identity authentication. He or she can go on to the next step, and so on. Otherwise, he or she will be denied service.

For example, in Table 2, for the user Alice, after inputting the username and password, she will get the user ID "0," then the system will calculate $f(x)$:

$$f(x) = a_0 + a_1x + a_2x + a_3x + a_4x + a_5x = a_0 = f(0) = 12.$$

From the authentication value, the authors know that Alice passes the identity authentication.

For Mary, after the first pass, she will get the user ID "12," then:

$$f(x) = a_0 + a_3x + a_4x + a_5x = f(12) = a_0 + a_3 \cdot 12 + a_4 \cdot 12 + a_5 \cdot 12 = 645.$$

If the calculated value is equal to her corresponding authentication value, then Mary passes the second pass, otherwise she is rejected.

In Table 2, to keep them safe, username and password are not initial values but their hashes. Here, $H(\)$ is Hash(). Only users who match their username and password hashes can get the corresponding polynomial $f(x)$.

$$f(x) = a_0 + a_1x + a_2x + a_3x + a_4x + a_5x$$

In the function $f(x)$: x is the user ID. The authors can calculate the value by the polynomial; if the calculation result is equal to authentication value, then the user is a legitimate user, and he or she can go on to the next step; otherwise, he cannot be authenticated and will be refused access.

3.6. Infrastructure and Process. About the implementation process and underlying structure of this scheme, in the server side, it includes several modules:

3.6.1. Register/Login Module. If the user is an unregistered user, he or she needs to register. The user sends a register request to the server; the message contains the client's identity, group categories, purpose of access, etc. If he or she is a valid user, he or she will send a login request to the server based on his or her username and password, and the server will respond the "ACK" value (y/n) to the user.

3.6.2. Authentication and Authorization Module. Authentication and Authorization Center, named AAC. To the legitimate user, based on his or her identity information

and purpose of access, with the help of global StrucTree and privacy policy, the server will assign an ID and the corresponding authorization vector to the user. At the same time, all the register information is recorded in the authorization table.

3.6.3. Security Search and Response Module. The valid authorized user can go on sending all kinds of requests to the server. The server will create a temporary authorization matrix view based on the user's authorization vector and the global StrucTree, and then combine the structure and content with the help of encoding tables and auxiliary tables. At last, the meaningful information will be returned to the user.

The infrastructure and the procedure of communication are shown in Figure 6.

The detailed process is shown in Figure 7(a) to Figure 7(c).

4. Results and Discussion

In order to assess the validity and test the performance of the scheme, a serial of experiments are conducted. The test is in Windows 10 operation system with Inter (R)Core(TM) i5-6500 CPU 3.20 GHz and 8 GB RAM.

We use XMLSpy as the editor to check out the grammar of XML, and XML Schema is used to constrain its structure. At last, we use Visual C++ 2010 and Matlab mixed programming to finish all the experiments.

In order to be convenient to obtain information, we first need to parse the XML document, and then encode for all the nodes. Considering that DOM (Document Object Model) technology needs the entire document to be read while parsing the XML document, it requires very high memory requirement, and it is suitable for the resolution of small XML document. SAX (Simple API for XML) is a kind of simple XML application interface, it reads and analyzes XML documents using flow mechanisms, and it consumes little memory, so it is suitable for the resolution of large XML document.

4.1. Test Dataset. In the experiments, we choose two sets of datasets as test documents; the detailed information about test datasets is shown in Table 3.

The first test dataset is from No.1 to No.4 in Table 3, which is from the XML authority testing website. It is about auction data, and it is converted to XML document from web sources [9]. They are some small XML documents.

The second test set is relatively complex, and the amount of data is relatively large. It is from No.5 to No.7 in Table 3. The detailed information is in given below:

- (1) Mondial is a world geographic database integrated from the CIA World Factbook, the International Atlas, and the Terra database, among other sources. This document has 22423 elements and 47423 attributes.

TABLE 2: User authorization table.

Hash value of name	Hash value of password	ID	Authorization vector	Authentication value
$H(\text{Alice})$	$H(\text{bc3})$	0	$(a_0, a_1, a_2, a_3, a_4, a_5)$	12
$H(\text{Mary})$	$H(\text{ay4})$	12	$(a_0, 0, 0, a_3, a_4, a_5)$	645
$H(\text{Bob})$	$H(\text{ob8})$	34	$(a_0, a_1, 0, a_3, a_4, 0)$	723
...	...	...	...	...

FIGURE 6: (a) The infrastructure. (b) The communication and search procedure.

FIGURE 7: (a) Register and login module. (b) Authentication and authorization center. (c) The procedure of safety search.

TABLE 3: Test dataset.

No.	Dataset name	Size	Node number	Max-depth	Average-depth
1	ubid.xml	20 KB	342	5	3.76608
2	321gone.xml	23 KB	311	5	3.76527
3	yahoo.xml	25 KB	342	5	3.76608
4	ebay.xml	34 KB	156	5	3.75641
5	Mondial.xml	1 MB	69846	5	3.59274
6	Nasa.xml	23 MB	532963	8	5.58314
7	SwissPort.xml	109 MB	10166890	5	3.55671

- (2) Nasa is a dataset converted from legacy float-file format into XML and made available to the public from GSFC/NASA XML Project. It has 476646 elements and 56317 attributes.
- (3) SwissPort is a curated protein sequence database which strives to provide a high level of annotations (such as the description of the function of a protein, its domains' structure, post-translational modifications, and variants), a minimal level of redundancy, and a high level of integration with other databases. It has 2977031 elements and 2189859 attributes.

In all the datasets, in order to deal with the information, we treat attribute nodes as element nodes. In most actual XML documents, they have many repeated nodes with the same structure. So the construction of structural matrix greatly reduces the redundancy of the document, and at the same time, the matrix accelerates the authorization search. In the experiments, we mainly assess the performance from the perspective of storage space, execution time, and security verification.

4.2. Space Efficiency. In this system, the storage space is composed of the following parts: (1) the structure stored in a matrix; (2) the content tables; and (3) the auxiliary information. The auxiliary table includes the authorization table, the encoding table, etc..

The storage size is shown in formula (5). " M_{struc} " stands for the bytes of structure information, and " M_{content} " stands for the bytes of content saved as a table; let us emphasize that the leaf nodes' content refers to the content of all leaf nodes in the shared XML document. " $M_{\text{auxiliary}}$ " is the bytes of the auxiliary table.

$$M = M_{\text{struc}} + M_{\text{content}} + M_{\text{auxiliary}}. \quad (5)$$

In order to test the space efficiency of this scheme, we first provide the definition: compression ratio Cr. The formula is shown in (6).

$$Cr = \frac{R_b}{R_a} * 100\%. \quad (6)$$

R_a is the size of the XML document after the compression, and R_b is the size of the XML document before the compression. For the compression ratio Cr, the smaller its value means the more efficient the compression ratio, and vice versa.

In order to evaluate the performance of storage space, here we compare the space storage of three schemes; one is the basic storage scheme, which stores different tailored source XML documents for users with different access purposes. The second scheme is named Matrix scheme [17], which stores different tailored storage matrices for users, and the third scheme is our solution proposed in this paper, we have named it as the integration scheme.

Figures 8(a) and 8(b) are the comparison of storage space about source XML document, matrix scheme, and integration scheme, respectively. The horizontal coordinates stand for the name of test dataset 1 and the vertical

coordinates represent the storage space. Figure 8(a) is the comparison in small XML documents. Figure 8(b) is the comparison in large documents in test dataset 2.

From the result shown in Figure 8, we know that in the matrix scheme, the compression ratio is 63.16% in ubid.xml, 65.22% in 321gone.xml, 76% in yahoo.xml, and 85.29% in ebay.xml. In our scheme the compression ratio is 47.37% in ubid.xml, 69.57% in 321gone.xml, 80% in yahoo.xml, and 88.23% in ebay.xml. Based on the compression ratio to analyze the structure for small XML documents, we can conclude that the structure of the document has much big influence in integration authorization scheme; for documents with duplicate structures and multiple nodes they have good compression ratio in our scheme.

For the large XML document, the compression ratio is 82.30% in Mondial.xml, 75.40% in Nasa.xml, and 69.64% in SwissPort.xml in the Matrix scheme. In our scheme, the compression ratio is 61.7% in Mondial.xml, 67.93% in Nasa.xml, and 52.14% in SwissPort.xml; due to the removal of the structural redundancy, our scheme has good compression ratio, saves storage space, and reduces the burden on the server.

In order to further test space efficiency, we test the storage size from different views. The number of views is 5, 10, and 15, respectively. 5 different views means that the system has 5 different group users, and others are similar.

The test result is shown in Figure 9. In the source XML scheme, it keeps for every group user a tailored XML document based on the privacy policy. In the matrix scheme, it keeps every group user an authorization matrix, and in our scheme, it keeps all users a global matrix and their respective authorization vector.

From the test result, we know that with the increase of group users, three schemes have variations in storage space. In the Source XML scheme, the system will keep different views for every group, so its storage space increases gradually with the amount of views. Take ubid.xml as an example, its storage space size is about 80 KB, 170 KB, and 250 KB in 5, 10, and 15 views, respectively.

In the matrix scheme, its storage space grows slowly, its storage space size is about 25 KB, 32 KB, and 42 KB in 5, 10, and 15 views, respectively. But in our scheme, it only keeps one global Structree and some auxiliary tables, so with the increase of group users, its storage space is almost invariant, and its space size is almost unaffected by the number of group users. Take ubid.xml as an example, its storage space size is about 9 KB, 9 KB, and 9 KB in 5, 10, and 15 views, respectively.

The same test is used on another set of large XML datasets. The test result is shown in Figure 10.

In Source XML scheme, its storage space increases gradually with the amount of views. Assume that the shared document is Swissport.xml, its storage space size is about 300 MB, 850 MB, and 1100 MB in 5, 10, and 15 views, respectively. In matrix scheme, for the Swissport.xml its storage space is about 78 MB, 81 MB, and 83 MB in 5, 10 and 15 views, respectively. But in our integration scheme, it only keeps one global Structree and some auxiliary tables, for the Swissport.xml its storage space is about 56 MB, 57 MB, and 57 MB in 5, 10, and 15 views, respectively.

FIGURE 8: (a) Comparison of storage space in test dataset 1. (b) Comparison of storage space in test dataset 2.

FIGURE 9: (a) 5 views comparison in dataset 1. (b) 10 views comparison in dataset 1. (c) 15 views comparison in dataset 1.

Therefore, from the result and analysis, we can know that our integration scheme has good space efficiency compared with source XML scheme and matrix scheme.

4.3. Time Efficiency. The query time is another standard to judge the performance; in order to test the time efficiency, we mainly test the time from two aspects.

4.3.1. The Time of Parsing and Encoding XML Documents. In this test, we use DOM to parse a small XML document, and use SAX to parse a large XML document. At the same time, start-end encoding method is adopted to encode all the nodes [8]. The detailed parsing and encoding time is shown in Table 4.

From the result shown in Table 4, we know that parsing of a document is the first step to encode a document. A large document has a long encoding time. Start-end region

encoding is used in this scheme, because this scheme is highly efficient when searching for leaf nodes in any subtree.

4.3.2. Query Time in Different Locations in a XML Document. The test is from several aspects: (1) The query result tree is a nonrepeated subtree and has a 3-layer depth; (2) The query result tree is a repeated subtree; (3) The query result is a single repeated leaf node.

For the sake of fairness, we test the query time and compare them with the matrix scheme and our scheme under the same experiment conditions. XPath language is used to express different query information. Table 5 is the search time for test documents. Here, MX represents the matrix scheme; ours is the new integration scheme in this paper.

From the test results we know that the efficiency of the matrix scheme is not much different from ours in a small test

FIGURE 10: (a) 5 views comparison in dataset 2. (b) 10 views comparison in dataset 2. (c) 15 views comparison in dataset 2.

TABLE 4: Parsing and encoding time of test document.

NO.	Name	time(sec)
1	ubid.xml	0.073484
2	321gone.xml	0.102327
3	yahoo.xml	0.086543
4	ebay.xml	0.041523
5	Mondial.xml	2.486085
6	Nasa.xml	4.853410
7	SwissPort.xml	7.256712

TABLE 5: Search time of test XML documents.

NO.	Name	XPath	MX(s)	Ours(s)
1	Ubid	/Root/listing/seller_info	0.065853	1.534871
		auction_info	0.056238	0.276527
		auction_info/high_bidder	0.053587	0.076543
2	321gone	/Root/listing/auction_info	1.451320	1.253621
		seller_info	0.265445	0.365128
		auction_info/location	0.191922	0.087826
3	Yahoo	/Root/listing/bid_history	1.334671	1.634290
		item_info/brand	0.615505	0.577170
		Brand	0.120452	0.099234
4	EBay	/Root/listing/seller_info	1.217286	1.534871
		auction_info	0.925112	0.876536
		auction_info/num_bids	0.328130	0.296840
5	Mondial	/Mondial/continent	0.555891	0.165273
		/Mondial/country/name	3.055213	2.766082
		mountain/located	2.143377	1.756410
6	Nasa	/Datasets/dataset/subject	4.673392	3.756415
		dataset/keywords	3.208854	2.682302
		Reference/source/other/author	2.350644	1.250863
7	SwissPort	/root/Entry/AC	7.125266	5.180725
		Entry/Features	4.701453	3.148321
		Org	2.442136	1.540928

dataset. They all have good search efficiency. But in a large dataset, our solution is obviously superior in efficiency. The search time is mainly related to two points, one is the location of the query node, and the other is the repeatability of the node. Experiment result proves that our scheme has good response time for both small documents and large datasets. Of course, some preprocessing time is not discussed and tested here.

4.4. Security Analysis and Verification. In traditional schemes or models, almost all of them save the XML document in the same location, but in this scheme, it is based on the idea of separating storage of the structure and content, which disperses the information into two parts. Hence, from this aspect, we can conclude that this separation method is much safer than the previous scheme. In other words, for the adversary, this scheme is harder to decode and obtain useful or meaningful information than the traditional algorithm.

For the convenience of description and understanding, a medical XML document is used as an example; it has been encoded with the start-end encoding method. The detailed information is shown in Figure 11(a). The encoding information is shown in the upper left corner, for example: Node “EMR,” its encoding is (0, 8604).

Based on the idea of separating storage, Figure 11(b) is its corresponding global structure tree (StrucTree), and its corresponding unordered content is shown in Figure 11(c). The nodes marked with the symbol “*” are the repeated sub nodes.

From the structural particularity of the XML document, we know that the structure lacking content and the content lacking structure all are meaningless.

In this part, we suppose and simulate several intrusion scenarios from the security perspective. Through security analysis and verification, we further prove this scheme's security. Here attackers can be classified into three types.

4.4.1. Type I. The attacker knows all of the structure information, but has no content data. Assume that: There is m different structure information, they are $S_1, S_2, \dots, S_m$. From S_1 to S_m , we can know that they are different structure views for the shared XML document. Each is a cut of StrucTree through the filter of the privacy policy.

In the worst-case scenario, all the structure information is merged into one global view; the attacker can complete the reconstruction of the global structure tree StrucTree. The formula is shown in (5). Operator “ $\cup$ ” expresses merge operation, all the $S_1, S_2, \dots, S_m$ are merged into an view that removes the structural redundancy in $S_1, S_2, \dots, S_m$.

$$S = \bigcup_{i=1}^m S_i. \quad (7)$$

We know $S \leq \text{StrucTree}$. In the worst-case scenario, $S = \text{StrucTree}$. That means the attacker knows the entire structure information, and it indicates that he or she knows the framework about the XML document, but the framework is meaningless without content information. Through

the example, the attacker knows Figure 11(b). From this structure view, the attacker only knows that this is an EMR document, which includes all the patients' information, but no detailed patient data. For the attacker, the obtained information is meaningless.

4.4.2. Type II. The attacker obtains all the content data, but no structure information. Assume that there is n different content information, they are $C_1, C_2, \dots, C_n$. From C_1 to C_n , we can know that they are some content data of leaf nodes in the shared XML document. Each is a part of detailed information.

In the worst case, just like the example in Figure 11(c) even if the attacker knows part of the leaf node content or even all the leaf node content, they cannot infer any useful information. The attacker can know information, and the formula is shown in (6). Operator “ Σ ” expresses accumulation operation, all the $C_1, C_2, \dots, C_n$ are in a group.

$$C = \sum_{i=1}^n C_i. \quad (8)$$

Although the attacker knows the entire content data, but these disordered content is meaningless without structure information, and the attacker cannot distinguish detailed corresponding information.

4.4.3. Type III. The attacker gets partial structure and partial content data. If he or she knows the data's storage format, which means he or she may be able to guess or speculate some useful information.

In fact, from the analysis of the intrusion scenarios, we can conclude that Type I and Type II are the special cases of Type III. If Type III is secure, then Type I and Type II are also secure. So, we further verify the security in Type III.

We assume that the attacker obtains the global structure information StrucTree and some content information, at the same time, granting that the attacker knows the storage format of data.

- (1) Assume that: There is q core units' information in the entire XML document. For example, in Figure 11(a), the subtree with “Patient” as the root is a core unit. Correspondingly, for every core unit, there are r items to describe this unit, but there are s sensitive items in r items, here: $s \leq r$. For q core units, their content items of core units are $r * q$.
- (2) Assume that the attacker knows the StrucTree, meanwhile, he or she randomly obtains n items' content from $r * q$ items, and there are m group core units in n items. The probability p of obtaining one unit's sensitive information is in the following formula:

$$p = \frac{m}{C_{r * q}^n P_n^s}. \quad (9)$$

Alice, Mary,Bob, Smith, People hospital, 123-234,
123-567, A1, B2, 1966.12, 12, 1978.08, 26,

(c)

FIGURE 11: (a) The entire shared XML document with encoding. (b) Corresponding StructTree for Figure 11(a). (c) Corresponding unordered content for Figure 11(a).

- (3) In formula (6), P_n^s is the frequency that randomly gets s items from n items orderly. C_{r*q}^n is the frequency that randomly obtained n items from $r*q$ items. In the actual XML document, q is very large.
- (4) From formula (6), we know that when we set the variable s, n, r, m to be fixed values, for the large value q , we can calculate that the probability p is excessively small. Therefore, from safety analysis and certification of the abovementioned, we can conclude that the disclosure of information is very difficult, so this scheme is secure enough.

5. Conclusions

All kinds of data storage systems in shared environments have shown great potential and provide great convenience to people [10, 11]. At present, storage platforms can be violated by various malicious attacks. In this paper, we propose a new XML privacy-preserving data disclosure decision scheme.

First, this scheme adopts the idea of structure and content separation storage; based on this, the structure adopts temporary authorization matrix, and the core information, which is the content of leaf nodes, its access authorization, adopts authorization vector, and thus, with the integration of matrix and vector authorization, it completes the final authorization for different group users.

Second, to protect the content and link all the structure and content information, we use start-end encoding to encode all the nodes, and take the leaf node encoding as the content's address. Thus, in the server side, the content is stored in random order, the service providers can see the information, but they cannot get meaningful data. When the legitimate users obtain the validated access authorization, through the integration authorization and encoding information, they can obtain meaningful information.

In addition, the address of content and user ID are connected together to achieve authentication; this kind of polynomial authentication method also effectively enhances security. The experiment result shows that the scheme performs well with regard to storage space, response time, and security level.

In a word, at present, almost all the schemes are based on the hypothesis that a shared XML document is stored and running in a trusted server. With the development of computing services, the service provider is no longer a fully credible entity; on the contrary, it may even be a potential enemy. Servers that are not absolutely secure present great challenges to the traditional protection method.

Therefore, we still need to find good privacy-preserving schemes and research good search algorithms for XML documents in a shared and collaborative environment [10, 19].

Data Availability

The data that support the findings of this study are openly available in the University of Washington XML Repository, <http://www.cs.washington.edu/research/xmldatasets/>, refer

ence number: [9] or are available on request from the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the Jiangsu Provincial Industry-University-Research Cooperation Project (BY2019011).

References

- [1] S. Hammoudi, Z. Aliouat, and S. Aliouat, "Challenges and research directions for internet of things," *Telecommunication Systems*, vol. 67, no. 2, pp. 367–385, 2018.
- [2] A. Triantafyllou, P. Sarigiannidis, and T. D. Lagkas, "Network protocols, schemes, and mechanisms for internet of things: features, open challenges, and trends," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 5349894, 24 pages, 2018.
- [3] A. Robert and B. Georg Herzwurm, "Cloud manufacturing: a state-of-the-art survey of current issues-ScienceDirect," *Procedia CIRP*, vol. 72, pp. 947–952, 2018.
- [4] Z. Fu, Y. Liu, X. Sun, Y. Liu, and Z. Tian, "Confusing-keyword based secure search over encrypted cloud data," *Mobile Networks and Applications*, vol. 25, no. 1, pp. 125–132, 2020.
- [5] B. Safababar 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.
- [6] M. Thimma, F. Liu, J. Lin, and B. Luo, "HyXAC: hybrid XML access control integrating view-Based and query-rewriting approaches," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 190–2202, 2015.
- [7] B. Luo, D. Lee, W.-C. Lee, and P. Liu, "QFilter: rewriting insecure XML queries to secure ones using non-deterministic finite automata," *The VLDB Journal*, vol. 20, no. 3, pp. 397–415, 2011.
- [8] L. Guo, J. Wang, and H. Du, "XML privacy protection model based on cloud storage," *Computer Standards & Interfaces*, vol. 36, no. 3, pp. 454–464, 2014.
- [9] University of Washington XML Repository: <http://www.cs.washington.edu/research/xmldatasets/>.
- [10] K. Seol, Y.-G. Kim, E. Lee, Y.-D. Seo, and D.-K. Baik, "Privacy-preserving attribute-based access control model for XML-based electronic health record system," *IEEE Access*, vol. 6, pp. 9114–9128, 2018.
- [11] 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, no. 11, pp. 7147–7173, 2019.
- [12] Y. Qin, Q. Z. Sheng, H. Wang, and N. J. G. Falkner, "Organizing XML data in a wireless broadcast system by exploiting structural similarity," *Wireless Personal Communications*, vol. 98, no. 1, pp. 1299–1329, 2018.
- [13] C. D. Angela and K. Barker, "P4A: a new privacy model for XML," in *Proceedings Of the Twenty Second Annual IFIP WG 11.3 Working Conference On Data And Applications Security*, pp. 65–80, London, UK, July 2008.
- [14] H. C. Cankaya, "Access control lists," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds., Springer, Boston, MA, USA, 2011.

- [15] H. Bruneliere, F. M. de Kerchove, G. Daniel, S. Madani, D. Kolovos, and J. Cabot, "Scalable model views over heterogeneous modeling technologies and resources," *Software and Systems Modeling*, vol. 19, no. 4, pp. 827–851, 2020.
- [16] M. Javani and 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, no. 1, pp. 159–182, 2017.
- [17] L. Guo, J. Wang, H. Wu, and H. Du, "eXtensible markup language access control model with filtering privacy based on matrix storage," *IET Communications*, vol. 8, no. 11, pp. 1919–1927, 2014.
- [18] G. Z. Qadah, "Indexing techniques for processing generalized XML documents," *Computer Standards & Interfaces*, vol. 49, pp. 34–43, 2017.
- [19] S. Subramaniam, S.-C. Haw, and L.-K. Soon, "Improved centralized XML query processing using distributed query workload," *IEEE Access*, vol. 9, Article ID 29127, 2021.