IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Privacy-Preserving in XML Information Brokering using Automation Segmentation and Query Segment Encryption

N.Rajesh[1], Mrs.K.R.S.Chandrakala[2]

[1]PG Scholar, Computer Science and Engineering Department, Sriram Engineering College, Perumalpattu-602024, Tiruvallur, Tamil Nadu

[2]M.E, Computer Science and Engineering Department, Sriram Engineering College, Perumalpattu-602024, Tiruvallur, Tamil Nadu

Abstract

Organizations in many realms ranging from business to government agencies, there is an increasing need for interorganizational information sharing to facilitate extensive collaboration. Propose a novel IBS, namely Privacy Preserving Information Brokering (PPIB). It is an overlay infrastructure consisting of two types of brokering components, brokers and coordinators. The brokers, acting as mix anonymizer, are mainly responsible for user authentication and query forwarding. The coordinators, concatenated in a tree structure, enforce access control and query routing based on the embedded nondeterministic finite automata—the query brokering automata. To prevent curious or corrupted coordinators from inferring private information, design two novel schemes to segment the query brokering automata and encrypt corresponding query segments so that routing decision making is decoupled into multiple correlated tasks for a set of collaborative coordinators. While providing integrated in-network access control and content-based query routing, the proposed IBS also ensures that a curious or corrupted coordinator is not capable to collect enough information to infer privacy, such as "which data is being queried", "where certain data is located", or "what are the access control policies", etc. Experimental results show that PPIB provides comprehensive privacy protection for on-demand information brokering, with insignificant overhead and very good scalability.

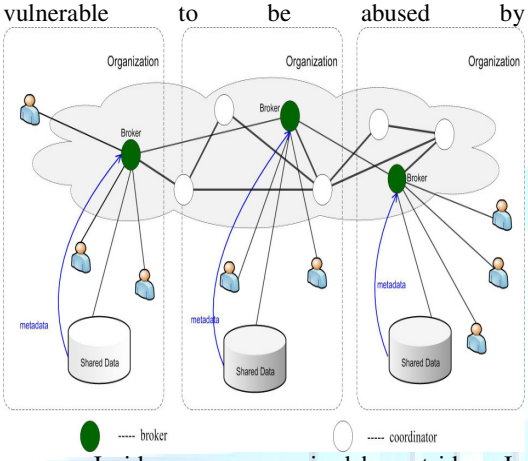
Index Terms- *Access control, information sharing,privacy*

I. INTRODUCTION

Along with the explosion of information collected by organizations in many realms ranging from business to government agencies, there is an increasing need for interorganizational information sharing to facilitate extensive collaboration. While many efforts have been devoted to reconcile data heterogeneity and provide interoperability, the problem of balancing peer autonomy and system coalition is still challenging. Most of the existing systems work on two extremes of the spectrum, adopting either the query-answering model toestablish pairwise client-server connections for on-demand information access, where peers are fully autonomous but there lacks systemwide coordination, or the distributed database model, where all peers with little autonomy are managed by a unified DBMS.

While being considered a solution between "sharing nothing" and "sharing everything", peer-to-peer information sharing framework essentially need to establish pairwise client-server relationships between each pair of peers, which is not scalable in large scale collaborative sharing. In the context of sensitive data and autonomous data providers, a more practical and adaptable solution is to construct a data-centric overlay consisting of data sources and a set of brokers that make routing decisions based on the content of the queries . Such infrastructure builds up semantic-aware index mechanisms to route the queries based on their content, which allows users to submit queries without knowing data or server location. In our previous study , such a distributed system providing data access through a set of brokers is referred to as *Information Brokering System* (IBS). applications atop IBS always involve some sort of consortium (e.g., RHIO) among a set of organizations. Databases of different organizations are connected through a set of brokers, and metadata (e.g., data summary, server locations) are "pushed" to the *local brokers*, which further "advertise" (some of) the metadata to other brokers. Queries are sent to the local broker and routed according to the metadata until reaching the right data server(s). In this way, a large number of information sources in different organizations are

loosely federated to provide an unified, transparent, and on-demand data access. While the IBS approach provides scalability and server autonomy, privacy concerns arise, as brokers are no longer assumed fully trustable—the broker functionality may be outsourced to third-party providers and thus vulnerable to be abused by



Insiders or compromised by outsiders. In this article, we present a general solution to the privacy- preserving information sharing problem. First, to address the need for privacy protection, we propose a novel IBS, namely *Privacy Preserving Information Brokering* (PPIB).

It is an overlay infrastructure consisting of two types of brokering components, *brokers* and *coordinators*. The brokers, acting as mix anonymizer, are mainly responsible for user authentication and query forwarding. The coordinators, concatenated in a tree structure, enforce access control and query routing based on the embedded nondeterministic finite automata—the *query brokering automata*. To prevent curious or corrupted coordinators from inferring private information, we design two novel schemes to segment the query brokering automata and encrypt corresponding query segments so that routing decision making is decoupled into multiple correlated tasks for a set of collaborative coordinators. while providing integrated in-network access control and content-based query routing, the proposed IBS also ensures that a curious or corrupted coordinator is not capable to collect enough information to infer privacy, such as "which data is being queried", "where certain data is located", or "what are the access control policies", etc.

II. THE PROBLEM

A. Vulnerabilities and the Threat Model

In a typical information brokering scenario, there are **three** types of stakeholders, namely *data owners*, *data providers*, and *data requestors*. Each stakeholder has its own privacy: (1) the privacy of a data owner (e.g., a patient in RHIO) is the identifiable data and sensitive or personal information carried by this data (e.g., medical records). Data owners usually sign strict privacy agreements with data providers to prevent unauthorized use or disclosure. (2) Data providers store the collected data locally and create two types of metadata, namely *routing metadata* and *access control metadata*, for data brokering. Both types of metadata are considered privacy of a data provider. (3) Data requestors may reveal identifiable or private information (e.g., information specifying her interests) in the querying content. For example, a query about AIDS treatment reveals the (possible) disease of the requestor. We adopt the *semi-honest*assumption for the brokers, and assume two types of adversaries, *external attackers* and *curious or corrupted brokering components*. External attackers passively eavesdrop communication channels. Curious or corrupted brokering components, while following the protocols properly to fulfil brokering functions, try their best to infer sensitive or private information from the querying process.

For instance, while data is protected over encrypted communication, external attackers still learn *query location* and *data location* from eavesdropping. Combining types of unintentionally disclosed information, the attacker could further infer the privacy of different stakeholders through *attribute-correlation attacks* and *inference attacks*.

Attribute-correlation attack. Predicates of an XML query describe conditions that often carry sensitive and private data (e.g., name, SSN, credit card number, etc.) If an attacker intercepts a query with multiple predicates or composite predicate expressions, the attacker can "correlate" the attributes in the predicates to infer sensitive information about data owner. This is known as the *attribute correlation attack*.

Inference attack. More severe privacy leak occurs when an attacker obtains more than one type of

sensitive information and learns explicit or implicit knowledge about the stakeholders through association. By "implicit", we mean the attacker infers the fact by "guessing".

B. Solution Overview

To address the privacy vulnerabilities in current information brokering infrastructure, we propose a new model, namely *Privacy Preserving Information Brokering* (PPIB). PPIB has three types of brokering components: *brokers*, *coordinators*, and a *central authority* (CA). The key to preserving privacy is to divide and allocate the functionality to multiple brokering components in a way that no single component can make a meaningful inference from the information disclosed to it. shows the architecture of PPIB. Data servers and requestors from different organizations connect to the system through local brokers. Brokers are interconnected through coordinators (i.e., the white nodes). A local broker functions as the "entrance" to the system. It authenticates the requestor and hides his identity from other PPIB components. It would also permute query sequence to defend against local traffic analysis. Coordinators are responsible for content-based query routing and access control enforcement. With privacy-preserving considerations, we cannot let a coordinator hold any rule in the complete form. Instead, we propose a novel *automaton segmentation scheme* to divide (metadata) rules into segments and assign each segment to a coordinator. Coordinators operate collaboratively to enforce secure query routing. A *query segment encryption scheme* is further proposed to prevent coordinators from seeing sensitive predicates. The scheme divides a query into segments, and encrypts each segment in a way that to each coordinator enroute only the segments that are needed for secure routing are revealed. Last but not least, we assume a separate *central authority* handles key management and metadata maintenance.

III. BACKGROUND

A. Related Work

Research areas such as information integration, peer-to-peer file sharing systems and publish-subscribe systems provide partial solutions to the problem of large-scale data sharing. Information integration approaches focus on providing an integrated view over a large number of heterogeneous data sources by exploiting the semantic relationship between schemas of different sources The PPIB study assumes that a global schema exists within the consortium, therefore, information integration is out of our scope. Peer-to-peer systems are designed to share files and data sets (e.g., in collaborative science applications). Distributed hash table technology is adopted to locate replicas based on keyword queries. Accordingly, the multicast solution in pub/sub systems does not scale in our environment and we need to develop new mechanisms One idea is to build an XML overlay architecture that supports expressive query processing and security checking a top normal IP network. In particular, specialized data structures are maintained on overlay nodes to route XML queries.

B. Preliminaries

1) XML Data Model and Access Control: The eXtensible Markup Language (XML) has emerged as the *de facto* standard for information sharing due to its rich semantics and extensive expressiveness. We assume that all the data sources in PPIB exchange information in XML format, i.e., taking XPath queries and returning XML data. Note that the more powerful XML query language, XQuery, still uses XPath to access XML nodes. In XPath, predicates are used to eliminate unwanted nodes, where test conditions are contained within square brackets "[]".

IV. PRIVACY-PRESERVING QUERY BROKERING SCHEME

The QBroker [9] approach has severe privacy vulnerability as we discussed in Section II. If the QBroker is compromised or cannot be fully trusted (e.g., under the honest-but-curious assumption as in our study), the privacy of both requestor and data owner is under risk. To tackle the problem, we present the PPIB infrastructure with two core schemes. In this section, we first explain the details of *automata segmentation* and *query segment encryption* schemes, and then describe the 4-phase query brokering process in PPIB.

A. Automaton Segmentation

In the context of distributed information brokering, multiple organizations join a consortium and agree to share the data within the consortium. While different organizations may have different schemas, we assume a global schema exists by aligning and merging the local schemas. Thus, the access control rules and index rules for all the organizations can be crafted following the same shared schema and captured by a global automaton. The key idea of automaton segmentation scheme is to *logically* divide the global automaton into multiple independent yet connected segments, and *physically* distribute the segments onto different brokering components, known as coordinators.

1) Segmentation: The trade-off between the processing complexity and the degree of privacy should be considered in deciding the granularity level. As privacy protection is of the primary concern of this work, we suggest a . To reserve the logical connection between the segments after segmentation, we define the following *heuristic segmentation rules*: (1) NFA states in the same segment should be connected via parent-child links; (2) sibling NFA states should not be put in the same segment without their parent state; and (3) the "accept state" of the original global automaton should be put in separate segments. To ensure the segments are logically connected, we also make the last states of each segment as "dummy" accept states, with links pointing to the segments holding the child states of the original global automaton

Algorithm 1 The automaton segmentation segmentation algorithm: deploySegment()

Input: Automaton State S
Output: Segment Address:adddr
1: for each symbol k in S.StateTransTable do
2: addr = deploySegment
(S.StateTransTable(k).nextState)
3: DS = createDummyAcceptState()
4: DS.nextState <- addr
5: S.StateTransTable(k).nextState <- DS
6:end for
7: Seg = createSegment()
8: Seg.addSegment()
9: Coordinator = getCoordinator()
10: Coorinator.assignSegment(Seg)
11: return Coordinator.address

2) Deployment: We employ physical brokering servers, called *coordinators*, to store the logical segments. To reduce the number of needed coordinators, several segments can be deployed on the same coordinator using different port numbers. Therefore, the tuple uniquely identifies a segment. For the ease of presentation, we assume each coordinator only holds one segment in the rest of the article. After the deployment, the coordinators can be linked together according to the relative position of the segments they store, and thus form a tree structure. The coordinator holding the root state of the global automaton is the root of the coordinator tree and the coordinators holding the accept states are the leaf nodes.

3) Replication: Since all the queries are supposed to be processed first by the root coordinator, it becomes a single point of failure and a performance bottleneck. For robustness, we need to replicate the root coordinator as well as the coordinators at higher levels of the coordinator tree. Replication has been extensively studied in distributed systems. We adopt the passive path replication strategy to create the replicas for the coordinators along the paths in the coordinator tree, and let the *centralized authority* to create or revoke the replicas (please see more details in Section V). The CA maintains a set of replicas for each coordinator, where the number of replicas is either a preset value or dynamically adjusted based on the average queries passing through that coordinator.

4) Handling the Predicates: In the original construction of NFA (similarly as described in QFilter and QBroker), a predicate table is attached to every child state of an NFA state as shown in Fig. 3. The predicate table stores predicate symbols (i.e., pSymbol), if any, in the corresponding query XPath step.

B. Query Segment Encryption

Informative hints can be learned from query content, so it is critical to hide the query from irrelevant brokering servers. However, in traditional brokering approaches, it is difficult, if not impossible, to do that, since brokering servers need to view query content to fulfill access control and query routing. Fortunately, the automaton segmentation scheme provides new opportunities to encrypt the query in pieces and only allows a coordinator to decrypt the pieces it is supposed to

process. The query segment encryption scheme proposed in this work consists of the *preencryption* and *postencryption* modules, and a special *commutative encryption*module for processing the double-slash ("//") XPath step in the query.

1) Level-Based Preencryption: According to the automaton segmentation scheme, query segments are processed by a set of coordinators along a path in the coordinator tree. A straightforward way is to encrypt each query segment with the public key of the coordinator specified by the scheme. Hence, each coordinator only sees a small portion of the query that is not enough for inference, but collaborating together, they can still fulfill the designed function. The key challenges in this approach is that the segment-coordinator association is unknown beforehand in the distributed setting, since no party other than the CA knows how the global automaton is segmented and distributed among the coordinators.

2) Postencryption: The processed query segments should also be protected from the remaining coordinators in later processing, so postencryption is necessary. In a simple scheme, we assume all the data servers share a pair of public and private keys, , where is known to all the coordinators. Each coordinator first decrypts the query segment(s) with its private level key, performs authorization and indexing, and then encrypts the processed segment(s) with so that only the data servers can view it.

3) Commutative Encryption for "//" Handling: When a query has the *descendant-or-self* axis (i.e., "//" in XPath expressions), a so-called *mismatching problem* occurs at the coordinator who takes the "//" XPath step as input. This is because that the "//" XPath step may recursively accepts several tokens until it finds a match. Consequently, the coordinator with the private level key may not be the one that matches the "//" token, and vice versa.

C. The Overall PPIB Architecture

The architecture of PPIB is shown in Fig. 7, where users and data servers of multiple organizations are connected via a broker-coordinator overlay. In particular, the brokering process consists of four phases:

- **Phase 1:** To join the system, a user needs to authenticate himself to the local broker. After that, the user submits an XML query with each segment encrypted by the corresponding public level keys, and a unique session key is encrypted with the public key of the data servers to encrypt the reply data.
- **Phase 2:** Besides authentication, the major task of the broker is metadata preparation: (1) it retrieves the of the authenticated user to attach to the encrypted query; (2) it creates a unique for each query, and attaches and its own address to the query for data servers to return data.
- **Phase 3:** Upon receiving the encrypted query, the coordinators follow automata segmentation scheme and query segment encryption scheme to perform access control and query routing along the coordinator tree as described in Sections IV-A and IV-B. At the leaf coordinator, all query segments should be processed and reencrypted by the public key of the data server. If a query is denied access, a failure message with will be returned to the broker.
- **Phase 4:** In the final phase, the data server receives a safe query in an encrypted form. After decryption, the data server evaluates the query and returns the data, encrypted by , to the broker that originates the query.

V. MAINTENANCE

A. Managing the Key

The CA is assumed for offline initiation and maintenance. With the highest level of trust, the CA holds a global view about all the rules and plays a critical role in automaton segmentation and key management. There are four types of keys used in the brokering process: query session key , public/private level keys , commutative level keys , and public/private data server keys . Except the query session keys created by the user, the other three types of keys are generated and maintained by the CA. The data servers are treated as a unique party and share a pair of public and private keys, while each of the coordinators has its own pairs of level key and commutative level key. Along with the automaton segmentation and deployment process, the CA creates key pairs for coordinators at each level and assigns the private keys with the segments. The level keys need to be revoked in a batch once a certificate expires or when a coordinator at the same level quits the system.

B. Brokering Servers Join/Leave

Brokers and coordinators, contributed by different organizations, are allowed to dynamically join or leave the PPIB system. Besides authentication, a local broker only works as an entrance to the coordinator overly. It stores the address of the root coordinator (and its replica) for forwarding the queries. When a new broker joins the system, it registers to the CA to receive the current address list from the CA and broadcasts its own address to the local users. When leaving the system, a broker only needs to broadcast a leave message to the local users.

C.Updating the Metadata

ACR and index rules should be updated to reflect the changes in the access control policy or the data distribution in an organization.

1) Index Rules: To add or remove a (set of) data object, a local server need to send an update message, in the form of , to the CA, where *object* is an XPath expression to describe a set of XML nodes, *address* is the location of the data object, and *action* is either "add" or "remove".

2) Access Control Rules: Any change in the access control policy can be described by (a set of) positive or negative access control rules. Therefore, we construct an message to reflect the change for a particular *role* and send it to the CA.

VI. PRIVACY AND SECURITY ANALYSIS

There are various types of attackers in the information brokering process. From their roles, we have abused insiders and malicious outsiders; from their capabilities, we have passive eavesdroppers and active attackers that can compromise any brokering server; from their cooperation mode, we have single and collusive attackers. In this section, we consider three most common types of attackers, local and global *eavesdroppers*, malicious *brokers* and malicious *coordinators*. We first analyze possible privacy breakages caused by each of them, and then summarize possible privacy exposures

1) Eavesdroppers: A local eavesdropper is an attacker who can observe all communication to and from the user side. Once an end user initiates an inquire or receives requested data, the local eavesdropper can seize the outgoing and incoming packets. However, it can only learn the location of local broker from the captured packets since the content is encrypted. Although local brokers are exposed to this kind of eavesdroppers, as a gateway of DIBS system, it prevents further probing of the entire DIBS. Although the disclosed broker location information can be used to launch DoS attack against local brokers, a backup broker and some recovery mechanisms can easily defend this type of attacks.

2) Single Malicious Broker: A malicious broker deviates from the prescribed protocol and discloses sensitive information. It is obvious that a corrupted broker endangers user location privacy but not the privacy of query content. Moreover, since the broker knows the root-coordinator locations, the threat is the disclosure of root-coordinator location and potential DoS attacks.

3) Collusive Coordinators: Collusive coordinators deviate from the prescribed protocol and disclose sensitive information. Consider a set of collusive (corrupted) coordinators in the coordinator tree framework. Even though each coordinator can observe traffic on a path routed through it, nothing will be exposed to a single coordinator because (1) the sender viewable to it is always a brokering component; (2) the content of the query is incomplete due to query segment encryption; (3) the ACR and indexing information are also incomplete due to automaton segmentation; (4) the receiver viewable to it is likely to be another coordinator. However, privacy vulnerability exists if a coordinator makes reasonable inference from additional knowledge. For instance, if a leaf-coordinator knows how PPIB mechanism works, it can assure its identity (by checking the automaton it holds) and find out the destinations attached to this automaton are of some data servers. Another example is that one coordinator can compare the segment of ACR it holds with the open schemas and make reasonable inference about its position in the coordinator tree. However, inference made by one coordinator may be vague and even misleading.

VII. CONCLUSION

The privacy of user, data, and metadata during the design stage, existing information brokering systems suffer from a spectrum of vulnerabilities associated with user privacy, data privacy, and metadata privacy. In this paper, we propose PPIB, a new approach to preserve privacy in XML information brokering. First, at present, site distribution and load balancing in PPIB are conducted in an ad-hoc manner. Our next step of research is to design an automatic scheme that does dynamic site distribution. Several factors can be considered in the scheme such as the workload at each peer, trust level of each peer, and privacy conflicts between automaton segments. Designing a scheme that can strike a balance among these factors is a challenge. Second, we would like to quantify the level of privacy protection achieved by PPIB. Finally, we plan to minimize (or even eliminate) the participation of the administrator node, who decides such issues as automaton segmentation granularity. A main goal is to make PPIB self-reconfigurable.

Acknowledgment

The authors would like to thank T. Yu, W.-C. Lee, P. Mitra, and M. Rabinovich for valuable discussions.

REFERENCES

[1] W. Bartschat, J. Burrington-Brown, S. Carey, J. Chen, S.Deming, and S. Durkin, “Surveying the RHIO landscape: A description of current {RHIO} models, with a focus on patient identification,” *J. AHIMA*, vol. 77, pp. 64A–64D, Jan. 2006.

[2] A. P. Sheth and J. A. Larson, “Federated database systems for managing distributed, heterogeneous, and autonomous databases,” *ACM Comput. Surveys (CSUR)*, vol. 22, no. 3, pp. 183–236, 1990.

[3] L. M.Haas, E. T. Lin, andM.A. Roth, “Data integration through database federation,” *IBM Syst. J.*, vol. 41, no. 4, pp. 578–596, 2002.

[4] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, “CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming,” in *Proc. IEEE INFOCOM*,Miami, FL, USA, 2005, vol. 3, pp. 2102–2111.

[5] A. C. Snoeren, K. Conley, and D. K. Gifford, “Mesh-based content routing using XML,” in *Proc. SOSP*, 2001, pp. 160–173.

[6] N. Koudas, M. Rabinovich, D. Srivastava, and T. Yu, “Routing XML queries,” in *Proc. ICDE’04*, 2004, p. 844.

[7] G. Koloniari and E. Pitoura, “Peer-to-peer management of XML data: Issues and research challenges,” *SIGMOD Rec.*, vol. 34, no. 2, pp.6–17, 2005.