IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. XI (Mar-Apr. 2014), PP 76-82
www.iosrjournals.org

Security Enforcement with query routing Information Brokering in Distributed Information Sharing

A. Banu Prabha
Student SRM University

***Abstract**: Information brokering system (IBS) shares information via on-demand access. IBS connect large-scale loosely federated data sources via a brokering overlay. It is a peer-to-peer overlay network that comprises diverse data servers and brokering components, that helps client queries to locate the data server(s). Peer-to-peer (P2P) systems are gaining increasing popularity as a scalable means to share data among a large number of autonomous nodesand to balance the load. Here brokers routes the decisions to direct client queries to the requested data servers. The proposed novel approach is to preserve privacy of multiple stakeholders involved in the information brokering process. The system defines two privacy attacks, namely attribute-correlation attack and inference attack. The Privacy-Preserving Information Brokering in Distributed Information Sharing is achieved by an innovative automaton segmentation scheme and query segment encryption scheme.*

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 inter organizational information sharing to facilitate extensive collaboration. The effort spent for reunite data heterogeneity and interoperability providing. Balancing peer autonomy and system coalition is still challenging. Most of the existing systems work on on-demand information access, where peers are fully autonomous peers and managed by a unified DBMS.

Take healthcare information systems as example. Regional Health Information Organization (RHIO) aims to facilitate access to and retrieval of clinical data across collaborative healthcare providers that include a number of regional hospitals, outpatient clinics, payers, etc. The provider requires full control on the data and the access to the data. A healthcare provider requesting data from other providers expects to preserve her privacy (e.g., identity or interests) in the querying process.

II. The Problem

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. For example, a query about AIDS treatment reveals the (possible) disease of the requestor.The attacker could further infer the privacy of different stakeholders through *attribute-correlation attacks* and *inference attacks*.

Attribute-correlation attack. An XML query describe conditions that 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". For example, an attacker can guess the identity of a requester from her query location (e.g., IP address) Attackers can also obtain publicly-available information to help his inference.

For example, if an attacker identifies that a data server is located at a cancer research center, he can tag the queries as "cancer-related".

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 apportion the functionality to multiple brokering components in a way that no single component can make a meaningful inference from the information disclosed to it.

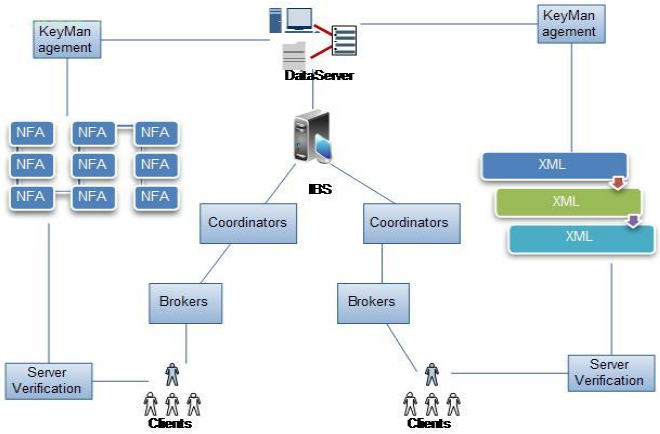


Fig. shows the architecture of PPIB. Data servers and requestors from different organizations connect to the system through local brokers Brokers are interconnected through coordinators. A local broker functions as the "entrance" to the system. It authenticates the requestor and hides his identity from other PPIB components. It would also permute query sequence to defend against local traffic analysis.

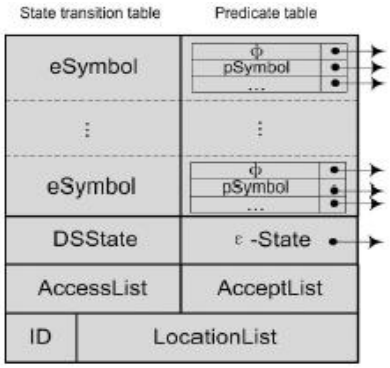
We propose a novel *automaton segmentation scheme* to divide (metadata) rules into segments and assign each segment to a coordinator. Coordinators operate collaboratively to enforce secure query routing. A *query segment encryption scheme* is further proposed to prevent coordinators from seeing sensitive predicates. The scheme divides a queryinto 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

Privacy concerns arise in interorganizational information brokering since one can no longer assume brokers controlled by other organizations are fully trustable. As the major source that may cause privacy leak is the metadata (i.e., indexing and access control), secure index based search schemes may be adopted to outsource metadata in encrypted form to untrusted brokers. Brokers are assumed to enforce security check and make routing decision without knowing the content of both query and metadata rules.
Research on anonymous communication provides a way to protect information from unauthorized parties. Many protocols have been proposed to enable the sender node dynamically select a set of nodes to relay its requests. These approaches can be incorporated into PPIB to protect location of data requestors and data servers from irrelevant or malicious parties.

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. The policy consists of a set of access control rules**(ACR)={subject, object, action, size, type}**

2) Content-Based Query Brokering: Indexing schemes have been proposed for content-based XML retrieval **I={object,location}**,



When an user queries the system, the XPath query is matched with the *object* field of the index rules, and the matched query will be sent to the data server specified by the *location* field of the rule(s). While other

techniques can be used to implement content-based indexing, we adopt the model with the NFA-based access control enforcement scheme.We call the integratedNFAthat captures access control rules and index rules *content-based query broker* (QBroker). QBroker is constructed in a similar way as QFilter. Fig. shows the data structure of each NFA state in QBroker, where the state transition table stores the child nodes specified by the XPath expression as the child states in **eSymbol** .QBroker adds two binary arrays to each state to capture rules for multiple roles: **AccessList** determines the roles that are allowed to access this state and **AcceptList** indicates for which role(s) the state is an accept state, A **LocationList** is attached to each accept state.

IV. Privacy-Preserving Query Brokering Scheme

The QBroker approach has severe privacy vulnerability as. If the QBroker is compromised or cannot be fully trusted, the privacy of both requestor and data owner is under risk. To tackle the problem, we present the PPIB infrastructure with two core schemes that are *automata segmentation* and *query segment encryption* schemes.

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. 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 atomic unit in the segmentation is an NFA state of the original automaton. Each segment is allowed to hold one or several NFA states. To reserve the logical connection between the segments afterbsegmentation, 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.

2) Deployment: We employ physical brokering servers, called *coordinators*, to store the logical segments. To reducethe number of needed coordinators, several segments can be deployed on the same coordinator using different port numbers.

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.

4) Handling the Predicates: In the original construction of NFA (similarly as described in QFilter [36] and QBroker [9]), 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. An empty symbol **ø** means no predicate.

Query Segment Encryption

Brokering servers need to view query content to fulfill access control and query routing. 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. That is encrypt each query segment with the public key of the coordinator specified by the scheme. Therefore 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 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*: A*ssume all the data servers share a pair of public and private keys, {pkDS, skDS, where pkDS 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 pkDSso that only the data servers can view it.

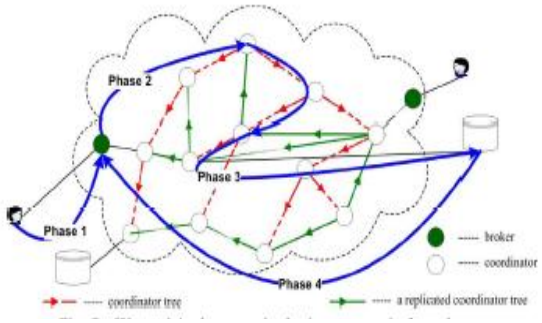
3) Commutative Encryption for "//" Handling:When a query has the *descendant-or-self* axis (i.e., "//" in XPath expressions), a so-called *mismatching problem* occurs at the coordinator who takes the "//" XPath step as input. This is because that the "//" XPath step may recursively accepts several tokens 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.

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 submitsan XML query with each segment encrypted by the corresponding public level keys, and a unique session key **Kq**.**Kq** 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 **role** ofthe authenticated user to attach to the encrypted query; (2) it creates a unique $\mathbf{Q_{rD}}$for each query, and attaches$\mathbf{Q_{rD}}$, $(Kq)_{pkDS}$ 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 querysegment encryption scheme to perform access control and query routing along the coordinator tree as described inSections 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 $\mathbf{Q_{rD}}$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 dataserver evaluates the query and returns the data, encrypted by $\mathbf{K_Q}$, to the broker that originates the query.

V. Maintenance

Key Management

The CA is assumed for offline initiation and maintenance. There are four types of keys used in the brokering process: query session key $\mathbf{K_Q}$, public/private level keys **{pk,sk}**, commutative level keys **{e,d}**, and public/private data server keys $\mathbf{\{pk_{DS}, sk_{DS}\}}$. Except the query session keys created by the user, the other three types of keys are generated and maintained by the CA. 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.

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. Thing are more complicate for the coordinators. Once joining the system, a new coordinator sends a join request to the CA.

Metadata Update

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 **DataUpdate(object, address, action)**, to the CA,

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 **ACRUpdate(role, object, type)**message to reflect the change for a particular *role* and send it to the CA. The CA forwards the message to the root coordinator, from which the XPath expression in *object* is processed by each coordinator according to its state transition table, in the same way as constructing an automaton with a new ACR: if the message stops at a particular NFA state, the state will be changed to an accept state for that role.

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 in Table I.

VII. Performance Analysis

In this section, we analyze the performance of proposed PPIB system using end-to-end query processing time and system scalability. In our experiments, coordinators are coded in Java (JDK 5.0) and results are collected from coordinators running on a Windows desktop (3.4 G CPU). We use the XMark [56] XML document and DTD, which is wildly used in the research community. As a good imitation of real world applications, the XMark simulates an online auction scenario.

End-to-End Query Processing Time

End-to-end query processing time is defined as the time elapsed from the point when query arrives at the broker until to the point when safe answers are returned to the user. We consider the following four components: (1) average query

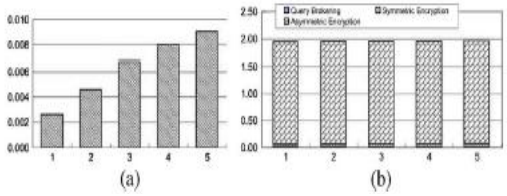


Fig. Estimate the overall processing time at each coordinator. (a) Averagequery brokering time at a coordinator. X: Number of keywords at a query broker. Y: Time (s). (b) Average symmetric and asymmetric encryption time. X: Number of keywords at a query broker. Y: Time (ms).brokering time at each broker/coordinator **(T_C)**; (2) average network transmission latency between broker/coordinators **(T_N)**; (3) average query evaluation time at data server(s) **(T_E)**; and (4) average backward data transmission latency **($T_{backward}$)**. Query evaluation time highly depends on XML databases system, size of XML documents, and types of XML queries. Once these parameters are set in the experiments $\mathbf{T_E}$, will remain the same (at seconds level [57]). Similarly, the same query set and ACR set will create the same safe query set, and the same data result will be generated by data servers. As a result $\mathbf{T_E}$, $\mathbf{T_{backward}}$ and are not affected by the broker-coordinator overlay network. We only need to calculate and compare the total forward query processing time ($\mathbf{T_{backward}}$)as$\mathbf{T_{backward} = Tc\ X\ N_{HOP} + T_N\ X\ (N_{HOP} + 1)}$. It is obvious that $\mathbf{T_{backward}}$ is only affected by T_C, T_N and the average number of hops in query brokering, $\mathbf{N_{HOP}}$.

1) Average Query Processing Time at the Coordinator:

Query processing time at each broker/coordinator **(T_C)**consists of: (1) access control enforcement and locating nextcoordinator (Query brokering); (2) generating a key and encrypting the processed query segment (Symmetric encryption); and (3) encrypting the symmetric key with the public key created by super node (Asymmetric encryption).

2) Average Network Transmission Latency*:* We adopt average Internet traffic latency 100 ms as a reasonable estimation of TN (from Internet traffic report) instead of using data collected from our gigabyte Ethernet.

3) Average Number of Hops*:* We consider the case in which a query **Q**is accepted or rewritten by **n** ACRs**{R1,…,Rn}** into the union of **n** safe subqueries **{Q'_1,…,Q'_n}.** When an accepted/rewritten subquery$\mathbf{Q'_i}$is processed by the rule $\mathbf{R_i}$, the number of hops is determined by the number of segments of $\mathbf{R_i}$. In the experiment, we generate a set of 200 synthetic access control rules and 1000 synthetic XPath queries.

4) End-to-End Query Processing Time:From above experiment results, the total forward query processing time is calculated as $\mathbf{T_{forward}}$ **~1.9 x 5.7 + 100 x (5.7 + 1) ~ 681** (ms). It is obvious that network latency $\mathbf{T_N*(N_{HOP}+1)}$dominates total forward end-to-end query processing time, because the value of $\mathbf{T_C}$is negligible compared with $\mathbf{T_N}$. Moreover, since $\mathbf{T_N}$ remains the same (as an estimation from Internet traffic), $\mathbf{N_{HOP}}$ becomes the deterministic factor that affects end-to-end query processing time. Note that for other information brokering systems, although they use different query routing scheme, network latency is not avoidable. As a conclusion, the proposed PPIB approach achieves privacy-preserving query brokering and access control with limited computation.

B. System Scalability

We evaluate the scalability of the PPIB system against complicity of ACR, the number of user queries, and data size(number of data objects and data servers).

1) Complicity of XML Schema and ACR*:* When the segmentation scheme is determined, the demand of coordinators is determined by the number of ACR segments, which is linear with the number of access control rules. Assume finest granularity automaton segmentation is adopted, we can see that the increase of demanded number of coordinators is linear or even better. This is because similar access control rules with same prefix may share XPath steps, and save the number of coordinators.Moreover, different ACR segments (or, logical coordinators) may reside at the same physical site, thus reduce the actual demand of physical sites. In this framework, the number of coordinators, **m**, and the height of the coordinator tree, **h** , are highly dependent on how access control policies are segmented.

2) Number of Queries:Considering **n** queries submitted into the system in a unit time, we use the total number of query segments being processed in the system tomeasure the system load. When a query is accepted as multiple subqueries, all subqueries are counted towards system load. For a query that is rejected after **i** segments, the processed **i** segments are counted.

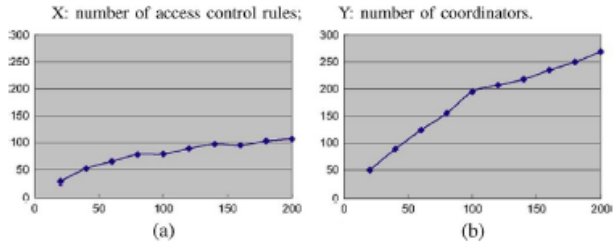


Fig. . System scalability: number of coordinators. (a) Using simple path rules.
(b) Using XPath rules with wildcards.

We generate 5 sets of synthetic ACRs and 10 sets of synthetic XML queries with different numbers and wildcard (i.e., "**/*** " and " //") probabilities at each XPath step in each experiment

3) Data Size:When data volume increases (e.g., adding more data items into the online auction database), the number of indexing rules also increases. This results in increasing of the number of leaf-coordinators. However, in PPIB, query indexing is implemented through hash tables, which is scalable. Thus, the system is scalable when data size increases.

VIII. Conclusion

With little attention drawn on 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. Through an innovative automaton segmentation scheme, in-network access control, and query segment encryption, PPIB integrates security enforcement and query forwarding while providing comprehensive privacy protection. Our analysis shows that it is very resistant to privacy attacks. End-to-end query processing performance and system scalability are also evaluated and the results show that PPIB is efficient and scalable.

Many directions are ahead for future research. 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.

References

[1]. W. Bartschat, J. Burrington-Brown, S. Carey, J. Chen, S. Deming, and S. Durkin, "Surveying the RHIO landscape: A description of current

[2]. RHIO models, with a focus on patient identification," Journal of AHIMA 77, pp. 64A–D, January 2006.

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

[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

[5]. Proceedings of IEEE INFOCOM, 2005.A. C. Snoeren, K. Conley, and D. K. Gifford, "Mesh-based content routing using XML," in SOSP, pp. 160–173, 2001.

[6]. G. Koloniari and E. Pitoura, "Peer-to-peer management of XML data: issues and research challenges," SIGMOD Rec., vol. 34, no. 2, 2005.

[7]. M. Franklin, A. Halevy, and D. Maier, "From databases to dataspaces: a new abstraction for information management," SIGMOD Rec., vol.34, no. 4, pp. 27–33, 2005.

[8]. F. Li, B. Luo, P. Liu, D. Lee, P. Mitra, W. Lee, and C. Chu, "In-broker access control: Towards efficient end-to-end performance of information brokerage systems," in Proc. IEEE SUTC, 2006.

[9]. F. Li, B. Luo, P. Liu, D. Lee, and C.-H. Chu, "Automaton segmentation: A new approach to preserve privacy in XML information brokering," in ACM CCS '07, pp. 508–518, 2007.

[10]. R. Agrawal, A. Evfimivski, and R. Srikant, "Information sharing across private databases," in Proceedings of the 2003 ACM SIGMOD,2003.

[11]. S. Mohan, A. Sengupta, and Y. Wu, "Access control for XML: a dynamic query rewriting approach," in Proc. IKM, pp. 251–252, 2005.

[12]. G. Skobeltsyn, Query-driven indexing in large-scale distributed systems. PhD thesis, EPFL, 2009.