Secure and Privacy-Preserving Information Brokering using broker-coordinator overlay in Distributed Information Sharing

Ramesha N C
4Th Sem M.Tech,
APSCE, Bangalore

Mr. Somashekar T
Senior Lecturer,
APSCE, Banglore

Abstract—Today's organizations raise an increasing need for in formation sharing via on-demand access. Information brokering systems (IBSs) have been proposed to connect large-scale loosely federated data sources via a brokering overlay, in which the brokers make routing decisions to direct client queries to the requested data servers. Many existing IBSs assume that brokers are trusted and thus only adopt server-side access control for data confidentiality. However, privacy of data location and data consumer can still be inferred from metadata (such as query and access control rules) exchanged within the IBS, but little attention has been put on its protection. In this paper, we propose a novel approach to preserve privacy of multiple stakeholders involved in the information brokering process. We are among the first to formally define two privacy attacks, namely attribute-correlation attack and inference attack, and propose two countermeasure schemes automaton segmentation and query segment encryption to securely share the routing decision-making responsibility among a selected set of brokerin g servers. With comprehensive security analysis and experimental results, we show that our approach seamlessly integrates security enforcement with query routing to provide system-wide security with insignificant overhead.

Index Terms—Access control, information sharing, privacy.

I. INTRODUCTION

A LONG with the explosion of information collected by or- ganizations in many realms ranging from business to gov-
ernment 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 t establish 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.

Unfortunately, neither model is suitable for many newly emerged applications, such as healthcare or law enforcement information sharing, in which organizations share information in a conservative and controlled manner due to business considerations or legal reasons. Take healthcare information systems as example. Regional Health Information Organization (RHIO) [1] 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. As a data provider, a participating organization would not assume free or complete sharing with others, since its data is legally private or commercially proprietary, or both. Instead, it requires to retain full control over the *data* and the *access to the data*. Meanwhile, as a consumer, a healthcare provider requesting data from other providers expects to preserve her privacy (e.g., identity or interests) in the querying process. In such a scenario, sharing a complete copy of the data with others or "pouring" data into a centralized repository becomes impractical. To address the need for autonomy, federated database technology has been proposed [2], [3] to manage locallystored data with a federated DBMS and provide unified data. In the context of sensitive data and autonomous data providers, a more practical and adaptable solution is to construct a data-centric overlay (e.g., [4], [5]) consisting of data sources and a set of brokers that make routing decisions based on the content of the queries [6]–[9]. 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 [9], [10], such a distributed system providing data access through a set of brokers is referred to as *Information Brokering System* (IBS). As shown in Fig. 1, 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

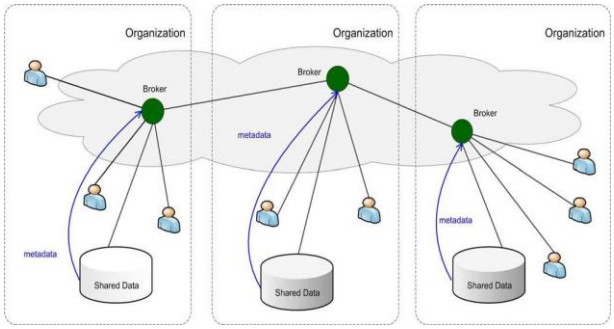

Fig. 1. Overview of the IBS infrastructure.

While the IBS approach provides scalability and server autonomy, privacy concerns arise, a 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 [11], 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. Experimental results show that PPIB provides comprehensive privacy protection for on-demand information brokering, with insignificant overhead and very good scalability.

II PROBLEM THEORM

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.

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 in- tercepts 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*.

Example 1: A tourist Anne is sent to ER at California Hospital. Doctor Bob queries for her medical records through a medicare IBS. Since Anne has the symptom of leukemia, the query contains two predicates: [pName="Anne"], and [symptom="leukemia"]. Any malicious broker that has helped routing the query could guess "Anne has a blood cancer" by correlating the two predicates in the query

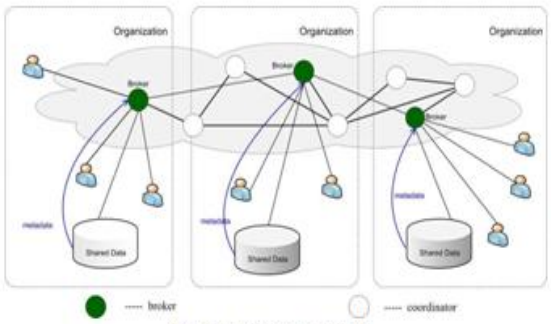

Fig. 2. Architecture of PPIB.

the fact by "guessing". For example, an attacker can guess the identity of a requestor from her query location (e.g., IP address). Meanwhile, the identity of the data owner could be explicitly learned from query content (e.g., name or SSN). 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".

B. Solution Overview

To address the privacy vulnerabilities in current information brokering infrastructure, we propose a new model, namely *Privacy Preservin 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.

Fig. 2 shows the architecture of PPIB. Data servers and requestors from different organizations connect to the system through local brokers (i.e., the green nodes in Fig. 2). 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.

The proposed for searchable encryption [22], [24], [23], however, to the best of our knowledge, all the schemes presented so far only support keyword search based on exact matching. While there are approaches proposed for multidimensional keyword

search [25] and range queries [26], supporting queries with complex predicates (e.g., regular expressions) or structures (e.g., XPath queries) is still a difficult open problem. In terms of privacy-preserving brokering, another related technique is secure computation [27] that allows one party to evaluate various functions on encrypted data without being able to decrypt.

Originally designed for privacy information retrieval (PIR) indatabase systems [28], such schemes have the same limitation that only keyword-based search is supported.

B. Preliminaries

1) XML Data Model and Access Control:

The eXtensible Markup Language (XML) has emerged as the*de facto*s tandard 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 [37] 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 "[]". In our study, we mainly focus on *value-based* predicates To specify the authorization at the node level,fine-grained access control models are desired. We adopt the 5-tuple access control policy that is widely used in the literature [38], [34], [36]. The policy consists of a set of access control

Existing access control enforcement approaches can be classified as *engine-based* [39], [32], [35], [40], [41], *view-based*[42], [43], [38], [44], [45], *preprocessing* [33], [34], [46]–[48], and *postprocessing* [49] approaches. In particular, we adopt the *Nondeterministic Finite Automaton* (NFA) based approach as presented in [36], which allows access control to be en- forced outside data servers, and independent from the data. The NFA-based approach constructs NFA elements for four building blocks of common XPath axes.

2) Content-Based Query Brokering: Indexing schemes have been proposed for content-based XML retrieval [50]–[53]. The index describes the address of the data server that stores a partic- ular data item requested by an user query. Therefore, a content- based index rule should contain the *content description* and the *address*. In [9], we presented a content-based indexing model with index rules in the form of, where

(1) *object* is an XPath expression that selects a set of nodes; and

(2) *location* is a list of IP addresses of data servers that hold thecontent.

Example 3. Example Index Rules:

I_1: {/site/people/person/name, 130.203.189.2}
I_2: {/site/regions//item[@id > ''100''], 135.176.4.56}
I_3: {/site/regions/samerica/item[@id > ''200''], 195.228.155.9}
I_4: {/site//namerica/item/name, 135.207.5.126}
I_5: {/site/regions/namerica/item/location, 74.128.5.91} ■

Example 2. Example ACRs:

R_1: {role$_1$, /site//person/name, read, +, RC}
R_2: {role$_1$, /site/regions/asia/item, read, +, RC}
R_3: {role$_2$, /site/regions/asia/item, read, +, RC}
R_4: {role$_2$, /site/regions/ * /item/name, read, +, RC}
R_5: {role$_2$, /site/regions/ * /item[location = ``USA"]/description, read, +, RC} ■

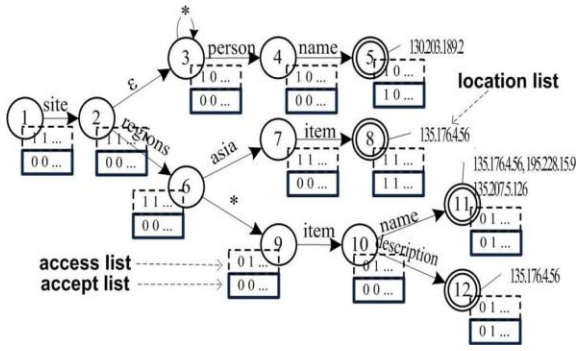

Fig. 4. State transition graph of the QBroker that integrates index rules with ACRs

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 (e.g., bloom filter [7], [6])

can be used to implement content-based indexing, we adopt the model in [9] in our study since it can be directly integrated with the NFA-based access control enforcement scheme. We call the integrated NFA that captures access control rules and index rules *content-based query broker* (QBroker). QBroker is constructed in a similar way as QFilter [36]. Fig. 3 shows the data structureof each NFA state in QBroker, where the state transition table stores the child nodes specified by the XPath expression as the child states in . The binary flag indicates that the state is a "double-slash" state. "double-slash" state, whose child state is an -transition state that directly transits to the next state without consuming any input symbol, will recursively accept input symbols. Fig. 4 shows a QBroker constructed from Example 2 and 3. Unlike QFilter that captures ACRs for only one role, QBroker adds two binary arrays to each state to capture

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 segmentencryption* 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 ofautomaton segmentation scheme is to *logically* divide the global automaton into multiple independent yet connected segments, and *physically* distribute the segments onto different brokering components

Algorithm 1 The automaton segmentation algorithm: $deploySegment()$

Input: Automaton State S

Output: Segment Address: $addr$

1: **for each** symbol k in $S.StateTransTable$ **do**
2: $addr = deploySegment(S.StateTransTable(k).nextState)$
3: $DS = createDummyAcceptState()$
4: $DS.nextState \leftarrow addr$
5: $S.StateTransTable(k).nextState \leftarrow DS$
6: **end for**
7: $Seg = createSegment()$
8: $Seg.addSegment(S)$
9: $Coordinator = getCoordinator()$
10: $Coordinator.assignSegment(Seg)$
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 coordi-

nator 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. Queries are processed along the paths of the coordinator tree in a similar way as they are processed by the global automaton: starting from the root coordinator, the first XPath step (token) of the query is compared with the tokens in the root coordinator.

If matched, the query will be sent to the next

coordinator, and so on so forth, until it is accepted by a leaf coordinator and then forwarded to the data server specified by the outpointing link of the leaf coordinator. At any coordinator, if the input XPath step does not match the stored tokens, the query will be denied and dropped immediately.

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.

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 means no predicate.

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,

a special *commutative encryption* module for processing the double-slash("//") XPath step in the query. to the root node. Since both the ACR and index rules are constructed following the global schema, an XPath step (token) in the XPath expression of a rule is associated with level

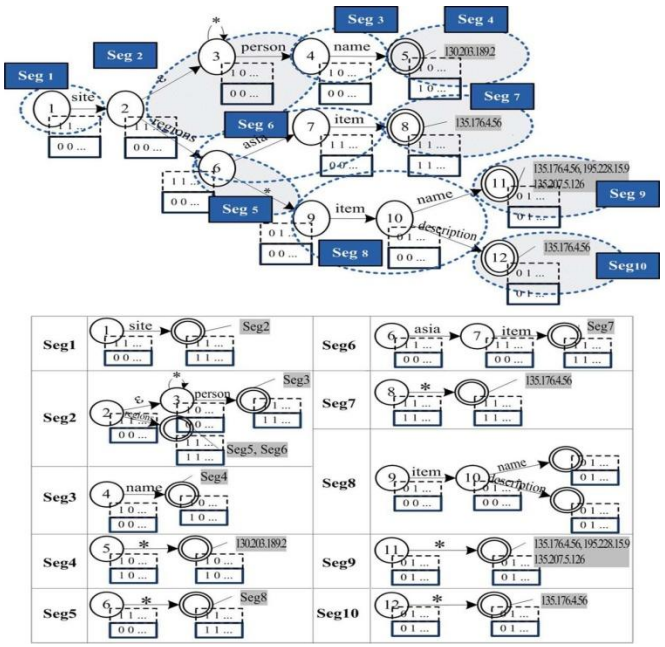

Fig. 5. Example to illustrate the automaton segmentation scheme: (a) divide the global automaton with granularity level of 1; (b) the segments are linked to form a tree structure.

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.

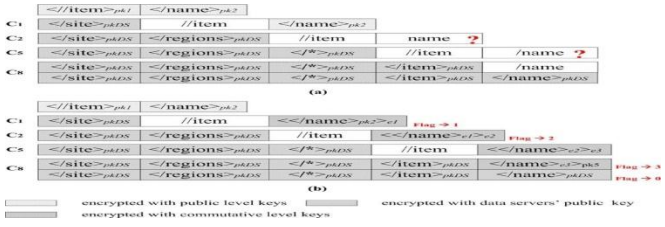

Fig 6. (a) Example of the mismatching problem; the mismatching problem caused by the descendant-or-self axis in a query. (b) Example of commutativeencryption; solve the mismatching problem with level-based commutative encryption.

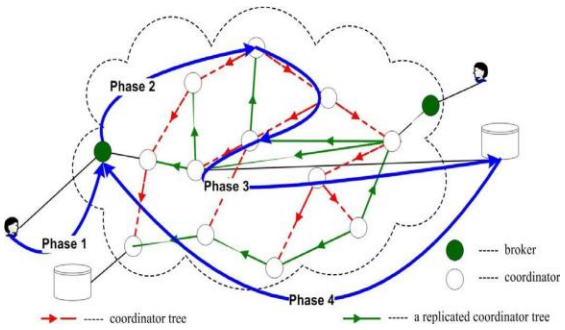

Fig. 7. We explain the query brokering process in four phases.

V. Maintenance

A. Key Management

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 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.

TABLE I
POSSIBLE PRIVACY EXPOSURE CAUSED BY FOUR TYPES OF ATTACKERS: LOCALEAVESDROPPER(LE), GLOBAL EAVESDROPPER (GE), MALICIOUS BROKER (MB), AND COLLUSIVE COORDINATORS (CC)

Privacy type	local eaves-dropper	global eavesdropper	malicious broker	collusive coordinators
User Location	Exposed	Exposed	Exposed	Protected
Query Content	Protected	Exposed	Exposed	Exposed only with compromised root coordinator
AC Policy	Protected	Protected	Protected	Exposed if path coordinators collude
Index Rules	Protected	Protected	Protected	Exposed if path coordinators collude
Data Distribution	Protected	Protected	Protected	Exposed if path coordinators collude
Data Location	Protected	Beyond suspicion	Protected	Exposed with malicious leaf coordinators

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.

A. 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

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, as shown in Fig. 9(a) and (b). 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.

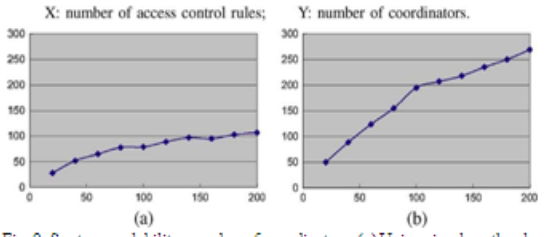

Fig. 9. System scalability: number of coordinators. (a) Using simple path rules. (b) Using XPath rules with wildcards.

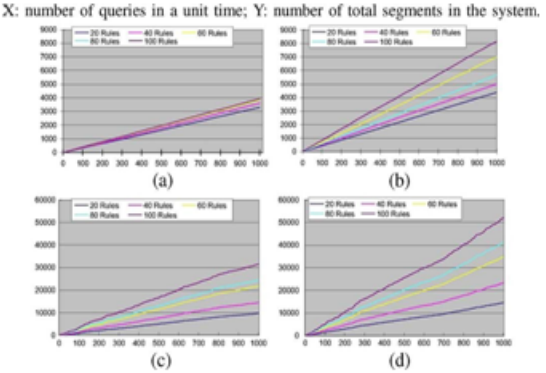

Fig. 10. System scalability: number of query segments. (a) Using simple XPath rules and simple XPath queries. (b) Using simple XPath queries and rules with wildcards. (c) Using queries and rules with 5% wildcards probability at each XPath step. (d) Using query and ACR with 10% wildcards probability at each XPath step.

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. Fig. 10 shows system load versus. number of XPath queries in a unit time. More specifically, Fig. 10(a) only has simple path rules (without wildcard or predicate), and Fig. 10(b) has rules with wildcards. In both cases, system load increases linearly and each query is processed less than 10 segments. Fig. 10(c) and (d) use the same set of ACRs as in Fig. 10(b), but add wildcards into queries with probability 5% and 10% at each step, respectively. In the worst case, each query is processed no more than 50 segments. Moreover, if we compare curves in each subfigure, we can see that larger ACR leads to higher system load, but the increase appears to be linear in all cases.

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{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, and M. 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.

[8] 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.

[9] 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 infor-mation brokerage systems," in *Proc. IEEE SUTC*, Taichung, Taiwan,2006, pp. 252–259.

[10] 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 *Proc. ACM CCS'07*, 2007, pp. 508–518.

[11] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[12] R. Agrawal, A. Evfimivski, and R. Srikant, "Information sharing across private databases," in *Proc. 2003 ACM SIGMOD*, San Diego, CA,USA, 2003, pp. 86–97.