Automaton Segmentation: A New Approach to Preserve Privacy in XML Information Brokering

Fengjun Li, Bo Luo, Peng Liu, Dongwon Lee, and Chao-Hsien Chu
The Pennsylvania State University
University Park, PA 16802, USA
{fengjun, bluo, pxl20, dongwon, chc4}@psu.edu

ABSTRACT

A *Distributed Information Brokering System (DIBS)* is a peer-to-peer overlay network that comprises diverse data servers and brokering components helping client queries locate the data server(s). Many existing information brokering systems adopt server side access control deployment and honest assumptions on brokers. However, little attention has been drawn on privacy of data and metadata stored and exchanged within DIBS. In this paper, we address privacy-preserving information sharing via on-demand information access. We propose a flexible and scalable system using a *broker-coordinator overlay network*. Through an innovative automaton segmentation scheme, distributed access control enforcement, and query segment encryption, our system integrates security enforcement and query forwarding while preserving system-wide privacy. We present the automaton segmentation approach, analyze privacy preservation in details, and finally examine the end-to-end performance and scalability through experiments and analysis.

Categories and Subject Descriptors

K.4.1 [**COMPUTERS AND SOCIETY**]: Public Policy Issues—*privacy*; K.6.5 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection

General Terms

Security

Keywords

Privacy, XML, Access Control

1. INTRODUCTION

In a federated information system with diverse participants (from different organizations) such as data producers, data consumers, or both, the need of cross-organizational information sharing naturally arises. However, different types of applications often need different forms of information sharing. In particular, while some applications (e.g., stock price updating) would need a publish-subscribe framework [3, 6], the on-demand information access is more suitable for other applications. Examples include cases like querying for products (parts) from manufactures and contractors network or providing emergency health care services to visitors (or tourists) whose medical records are not in local hospitals. Consider the following motivating example.

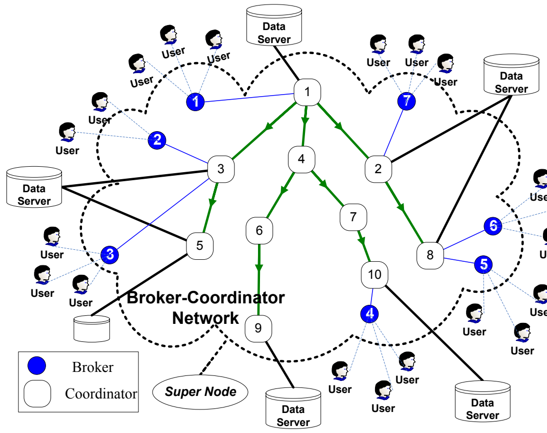


Figure 1: System architecture of a *distributed information brokering system*

Example 1. Let us consider a medicare network scenario. Each organization (e.g., hospital) participates as a data source that holds its own patient database. Since the records are highly sensitive and private, intensive privacy and security enforcement is desired. Diverse users (e.g., doctors, assistants, pharmacists, and administrators) are to access local or remote patient data according to certain access control policies. Furthermore, users who ask queries from their own terminals do not have to have prior knowledge of data distribution. For instance, when a doctor wants to retrieve all the historical records of a patient, her query may be forwarded to all data sources that hold related information. However, the user does (and should) not need to known where the data comes from. □

When N parties need to share data, as shown in Example 1, "pouring" all data into a centralized repository managed by a third party may lead to legal/political hurdles and trust/privacy concerns. In such scenarios, a peer-to-peer information sharing framework can be desirable. In its simplest form, we may establish two symmetric client-server

CCS'07, October 29–November 2, 2007, Alexandria, Virginia, USA.
Copyright 2007 ACM 978-1-59593-703-2/07/0010 ...$5.00.

relationships between every pair of parties, but having 2^N relationships is not scalable. To achieve better scalability, peer-to-peer overlay networks have been proposed to include not only the data servers of N parties but also a set of information brokering components helping client queries locate the right data server(s) [11, 12, 13]. In this paper, such a distributed on-demand information access system is referred to as *Distributed Information Brokering System (DIBS)*. Figure 1 shows an example DIBS (to be elaborated in Section 3.4). When data are owned, scattered, and managed by multiple parties in DIBS, various privacy concerns arise. Consider the following example.

Example 2. Continuing from Example 1, suppose that Anne is in ER and all patient data are stored and managed in XML format (as opposed to in relational records). If a doctor's XML query, "`/provider/.../patient[name()='Anne'] /symptom[cancer()='blood']//*`", is disclosed, then people may guess that Anne has a cancer. Similarly, Anne may not wish to reveal that she is now in Los Angeles under emergency health care but her health records are stored in Mt. Sinai Hospital of New York, since people may guess that she has cancer related problem if they know that her records are from the hospital renowned for its blood cancer treatment. That is, a medicare DIBS needs to protect not only confidentiality of patient data, but also privacy of such sensitive information as "who asks what queries" or "where data comes from". □

Despite its importance, to our best knowledge, none of existing DIBS work is designed with *user* and *data* privacy in mind. To satisfy such privacy protection requirements, therefore, we propose a novel DIBS, named as *Privacy Preserving Information Brokering system* (*PPIB*). As shown in Figure 1, PPIB contains a *broker-coordinator overlay network*, in which the brokers are responsible for forwarding user queries to coordinators concatenated in tree structure while preserving privacy. The coordinators, each holding a *segment* of access control automaton and routing guidelines, are mainly responsible for access control and query routing.

PPIB takes an innovative automaton segmentation approach to privacy protection. In particular, two critical forms of privacy, namely query content privacy and data object distribution privacy (or data location privacy), are enabled by a novel automaton segmentation scheme, with a "little" help from an assisting query segment encryption scheme. This scheme preserves privacy without sacrificing functionality. While providing "full" capability to do in-network access control and to route queries to the right data sources, this scheme ensures the information that a (curious, corrupted or broken) coordinator can gather is far from being enough to infer either "which data is being queried" or "where the data is located". Second, the automaton segmentation scheme can also provide high-quality privacy protection to metadata (e.g., access control policy). Third, user location privacy is protected by multilateral security, a design principle of *PPIB*.

To the best of our knowledge, (1) *PPIB* is the first system that uses automaton segmentation to do privacy-preserving in-network access control. (2) *PPIB* is the first system that integrates automaton segmentation, in-broker access control, and query routing. (3) *PPIB* provides the most comprehensive privacy protection for information brokering systems, and its performance degradation is insignificant compared with traditional DIBS systems (in a practical setting, the performance degradation of *PPIB* is at milliseconds level). (4) The evaluation results show that *PPIB* is a scalable privacy solution.

2. PROBLEM STATEMENT

2.1 Distributed Information Brokering Systems

Conceptually, a distributed information brokering system (DIBS) is a peer-to-peer overlay network consisting of data servers, brokering components, and end users. Applications atop DIBS always involve some sort of *consortium* among a set of data owners (or organizations). While expressing a strong need of cross-organizational information sharing, data owners in such a consortium still expect to remain as much autonomous as possible. As a result, data owners collect data independently, and manage it in their local data servers. Data is not poured into some center data warehouse or replicated in distributed databases. Instead, data servers send *metadata* about their data objects distribution as well as access control rules to the consortium, which will further assign them to brokers to help information brokering.

Traditional information sharing approaches always assume the use of trustable servers, such as the central data warehousing server or database servers. However, the honest or semi-honest assumptions (e.g., honest-but-curious assumption as adopted in [2]) may not hold for brokers. In practice, they may either be abused by insiders or compromised by outsiders. It is obvious that the brokers become the most vulnerable privacy breach of a DIBS, which leads to inevitable security and privacy risks. On one hand, the survival of information brokering depends on the trust of brokers to enforce authentication, access control as well as query forwarding, while on the other hand, failing to provide proper protection of information released in this process may create circumstances that harm the privacy of user, data and the system.

2.2 Privacy Vulnerabilities

In existing research of DIBS, relatively little attention has been drawn to privacy protection. To impose order into the multitude of privacy vulnerabilities in current DIBS approaches, we propose a taxonomy of privacy in **three** types: *User Privacy*, *Data Privacy*, and *Metadata Privacy*.

A. User Privacy

Generally speaking, we can summarize the user privacy as *"who, where, and what"*. "Who" refers to the identity of a user, "where" denotes his/her location at the moment of sending a query, and "what" represents the interest and purpose that can be inferred from his/her query.

User location could be easily retrieved by analyzing the IP packet of the query. User identity is a key concern of user privacy, which can be obtained either from authentication process or by associating user location information with other public data. Although the "what" privacy may not be reveled directly, one still can make reasonable inference from the content of the query. Sometimes, the inference is sensitive information of others as shown in Example 2. Sometimes, the inference is about user's own interest or purpose. For example, if Bob sends a query "`//departure[code=JFK, date=08/15]/arrival[code=SFO]//*`" to a distributed airline reservation system, the intermediate broker can easily infers that Bob needs a ticket from New York to San Francisco on Aug 15. In either circumstance, the "what" privacy is hurt.

Although *user identity*, *user location*, and *query content* are privacy-sensitive matters, one cannot apply popular privacy-preserving techniques directly in the DIBS. This is because a broker needs to learn these privacy-sensitive information to fulfill query brokering. For example, since data in an DIBS is only accessible by legitimate users, user identity cannot be represented by anonymity as other privacy-preserving applications do. In other words, the broker is responsible for authenticating user identity. As a result, to what extent user's privacy is preserved highly depends on how we minimize the disclosure of these privacy-sensitive information. This requires a new mechanism where the broker cannot infer the privacy of individuals while still fulfilling its designated functions.

B. Data Privacy

In a DIBS, data owners collect data independently and manage it with autonomous data servers. While providing data access to legitimate users, data servers have to release certain privacy-sensitive information that needs to be protected. In general, we can express privacy concerns of data with two questions, "where is the data?" and "who stores what?". The former concerns *data location* privacy, and the latter, denoted *data object distribution* privacy, inquires which type of data is contained in a particular data server. Unlike other large public databases or data warehouse, data owners in the proposed DIBS are highly conservative about their data privacy. They only share data and data distribution within the consortium.

C. Metadata Privacy

Two types of metadata are involved in the information brokering process in an DIBS, *query indexing guidelines* and *access control rules*. The former describes where the data objects are distributed among all the data servers, and the latter assigns accessibility to legitimate users according to access control policy provided by data owners. It is obvious that the metadata is highly relevant to both the privacy of data location and the privacy of data object distribution. However, to facilitate information brokering, these metadata have to be stored at the intermediate brokering components, which may be abused by the insider or compromised by the outsider according to our assumptions. As a result, the metadata becomes an obvious and easier target of attacks. Risk rises when unsecured or dishonest brokering components try to abuse or leak these privacy-sensitive information. In existing DIBS approaches, a compromised broker can obtain data location information from indexing guidelines or access control policy since these information are stored in brokers to facilitate routing and access control. Even if we can adopt some encryption schemes to hide these sensitive information from brokers, a compromised broker can probe the whole system by sending snooping queries. In this way, a compromised broker is more dangerous to the system than ordinary malicious users.

Remark. Note that different types of privacy may be intertwined with each other. For instance, query indexing guidelines may reveal data server locations; query content, access control rules and query indexing guidelines may reveal data objects distribution.

3. BACKGROUND

3.1 XML Preliminaries

This paper will focus on semantically rich applications such as health care. For those applications, keyword-based indexing and querying techniques (e.g., [26, 18, 8]) would not meet the expressiveness needs. To illustrate, health care providers need to declaratively express flexible constraints on the information to be retrieved. When tourist Anne is in ER, the doctor may query for medical records that match "last name Anne, 5 or 6 years old, has an 'open femur fracture with contamination', female", and simple filename-based, keyword-based, or range queries are not sufficient for this context.

To support rich semantics, we assume data are queried and exchanged in XML format. Data is assembled into XML documents, conforming to XML syntax and semantic rules. An *XML document* consists of elements, attributes, and text nodes. An element has a set of attributes, and may contain other XML elements and text nodes. Thus, these elements collectively form a tree-base data structure. The widely-adopted XML standard allows people to abstract naive data representations (e.g., patient records) into semi-structured XML data which can be retrieved by expressive yet simple XPath queries. *XPath* is a restricted variation of regular path expressions, which can refer to all or part of the nodes in an XML document using axes [17]. *Axes* represent the structural relationships between nodes. In particular, an axis defines a set of nodes relative to the current node. For example, "`/`" denotes the child node, "`//`" denotes the current node itself and all the descendant nodes, and "`@`" denotes the attribute. Although several query languages using different query algebras have emerged recently, most of them use XPath for locating nodes in XML documents. Thus, although our system is applicable to any regular path expression and any query language based on it, we focus on XPath in this paper.

3.2 Access Control Model

Access control is required in most if not all DIBS. We adopt the popular XML access control model proposed in [5, 20, 22]. In this model, users are members of appropriate *roles*; and an *access control policy* consists of a set of role-based 5-tuple access control rules (ACR): $R = \{subject, object, action, sign, type\}$, where (1) *subject* is a role to whom an authorization is granted; (2) *object* is a set of XML nodes specified by XPath; (3) *action* is one of "read," "write," and "update"; (4) $sign \in \{+, -\}$ refers to access "granted" or "denied," respectively; and (5) $type \in \{LC, RC\}$ refers to either "Local Check" (i.e., authorization is only applied to attributes or textual data of context nodes–"`self::text() | self::attribute()`"), or "Recursive Check" (i.e., authorization is applied to context nodes and propagated to all descendants–"`descendant-or-self::node()`"). When an XML node does not have either explicit (via LC rules) or implicit (via RC rules) authorization, it is considered to be "access denied." It is possible for an XML node to have more than one relevant access control rule. If conflict occurs between "+" and "−" rules, "−" rules take precedence. Five example access control rules under the 5-tuple model are shown in Figure 2

In our DIBS, each owner contributes a policy governing the access to her data objects, and the *system-wide* access control policy is simply the union of all the per-owner policies.

3.3 Automata-based Access Control Enforcement

View-based access control enforcement suffers from excessive storage requirement and expensive maintenance. Many view-free XML access control mechanisms are proposed to overcome the disadvantage. In our approach, we adopt and extend a view-free automaton-based access control mechanism proposed in [15]. It uses XPath expressions in access control rules (ACR) to build a Non-deterministic Finite Automaton (NFA). We call such a NFA an *access control automaton*. Each incoming query is checked against the NFA. As a result, each query could be (1) accepted: when user is allowed (by ACR) to access all the requested nodes, the query is kept as is. (2) rewritten: when user is allowed to access part of the requested content, the query is rewritten into a safe one, which asks for authorized content only. (3) denied: when user is not allowed to access any requested node, query is rejected.

Here, we use an example to illustrate how automaton-based access control enforcement works. In the examples throughout the paper, we adopt the well known XMark [23] schema, mimicking an online auction. As shown in Figure 2, we use 5 access control rules assigned to two roles. First, an automaton is built based on the XPath expressions from *object* part of the rules. For example, constructing NFA with rule R_1, we will have automaton states 0, 1, 2, 3, and 4 as shown in the figure. Especially, state 4 is an *accept state*, as shown in double-circle. Moreover, each state is attached with two binary arrays, namely *access list* (indicating which roles can access this state) and *accept list* (indicating this is accept state for particular role(s)), respectively. For instance, state 4 is accessible to users of *role 1* only, and is an accept state for this role.

At run time, user queries are checked against the automaton. Using the same example, suppose a user of *role 1* asks three XPath queries. (1) Q_1: `/site/categories/books/name` goes through states 1, 2, 3, and reach accept state 4. As a result, this query is accepted. (2) Q_2: `/site/regions/asia/* /name` goes through states 0, 1, 5, 6, 7 and 8. Based on the semantics of `*` in XML, the query is rewritten into into safe query: `/site/regions/asia/item/name`. (3) Q_3: `/site/regions/*/item/price` is denied since no accept state can be reached.

Due to space limit, we omit the details of automaton based access control. Please refer to [15, 13].

3.4 Automata-based In-network Query Broker

In traditional DIBS, access control mechanisms are implemented at data servers so as to check the accessibility right of a query (either by the database kernel or by a query filter [15] outside of the kernel) before answering it. However, [13] claims that, whenever access control is enforced at the data source-side, suspicious queries are allowed to traverse through the whole system until they get rejected at the far end. Thus, by sending snooping queries, attackers can probe the system to get data distribution and server location information, and do further inferences after successfully finding out the location of sensitive data. In addition, source-side access control wastes substantial network resources (e.g., bandwidth). To tackle the above problems, in-broker access control is proposed in [13]. The idea is to "push" the access control mechanism from the edge (i.e., data source side) to the "heart" of information brokering systems (i.e., information brokers). In this paper, we will embrace this idea and do access control at coordinators.

Access control rules:

R_1: *{ role 1, "/site/categories//name", read, +, RC}*

R_2: *{ role 1, "/site/regions/*/item/location", read, +, RC}*

R_3: *{ role 1, "/site/regions/*/item/quantity", read, +, RC}*

R_4: *{ role 2, "/site/regions/*/item/description", read, +, RC}*

R_5: *{ role 2, "/site /regions/*/item/name", read, +, RC}*

Indexing guidelines:

L_1: *{ "/site/categories/category/name", 192.168.0.5}*

L_2: *{ "/site//*/item/location", 192.168.0.1}*

L_3: *{ "/site/regions", 192.168.0.3}*

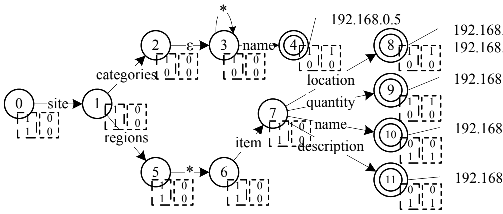


Figure 2: An example of automaton based access control and routing.

Content-based XML routing is applied in all DIBS [11, 12, 13]. In this paper, we adopt a rather simple content-based routing scheme, as the one used in [13]. In the scheme, each coordinator holds a set of indexing guidelines, and each *indexing guideline* consists of (1) an XPath expression indicating data objects and (2) an IP address indicating data location. It means that if a query matches the XPath expression, it will be forwarded to the IP address.

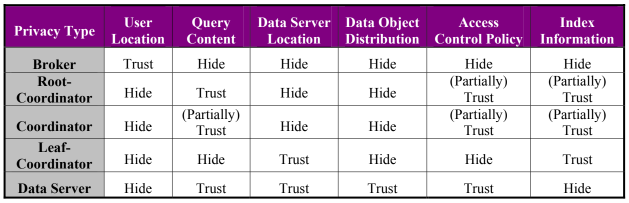
In [13], indexing guidelines are attached to accept states of the access control NFA, since only accepted/rewritten queries should be forwarded to the data servers. Let us use the indexing guidelines (L_1 to L_3) in Figure 2 as an example. Comparing with the five access control rules, L_1 is only relevant to R_1. As a result, IP of `192.168.0.5` is attached to state 4, and all queries accepted at this state will be forwarded to data server `192.168.0.5`. Please refer to [13] for details.

3.5 Assumptions

We assume that multiple *data owners* contribute XML data to DIBS. Therefore, data is stored in multiple *data servers* which are geographically distributed; and data may be replicated. In our model, each data location is an IP address identifying a unique data server; and each data object is indexed by an XPath expression. We assume data owners share data within some sort of *consortium*. Whenever a new data owner O joins the consortium, (1) O will let the consortium know which data objects she owns and where the data objects are stored, and (2) the consortium will "align" and merge the XML *schema* of O's data into the XML schema shared by the members of the consortium. We assume all XPath queries are crafted based on the shared XML schema.

4. PRIVACY PRESERVING INFORMATION BROKERING APPROACH (PPIB)

In this section, we propose an innovative *Privacy Preserving Information Brokering* (*PPIB*) framework to address the



Privacy Type	User Location	Query Content	Data Server Location	Data Object Distribution	Access Control Policy	Index Information
Broker	Trust	Hide	Hide	Hide	Hide	Hide
Root-Coordinator	Hide	Trust	Hide	Hide	(Partially) Trust	(Partially) Trust
Coordinator	Hide	(Partially) Trust	Hide	Hide	(Partially) Trust	(Partially) Trust
Leaf-Coordinator	Hide	Hide	Trust	Hide	Hide	Trust
Data Server	Hide	Trust	Trust	Trust	Trust	Hide

Figure 3: Brokering components have restricted trust on system privacy.

user/data/metadata privacy vulnerabilities associated with existing distributed information brokering systems, including the DIBS presented in [13].

4.1 Broker-Coordinator Overlay

As shown in Figure 1, we consider a broker-coordinator overlay consisting of N brokers and M coordinators, denoted by $B_1, B_2, ...B_N$ and $M_1, M_2, ...M_M$, respectively. Based on their functions, coordinators are classified into three types: a root-coordinator M_1, intermediate coordinators $M_2, ...M_i$, and leaf-coordinators $M_{i+1}, ...M_N$. All the coordinators form a *coordinator tree* of height h: (1) The root-coordinator is the root of the tree. It is an entrance for incoming queries. (2) Each intermediate coordinator holds a specific *segment* of an access control automaton (see Section 3). It also holds the location information of its child coordinator(s), which holds the "next segment" of the automaton. (3) Each leaf-coordinator holds an NFA-based query indexer, which was constructed from indexing rules. At runtime, it forwards safe queries to data servers, which has the data to answer it.

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 (to be elaborated in Section 4.2). In the broker-coordinator overlay, brokers and coordinators work separately and cooperatively. N brokers are distributed in the DIBS system so that each user is directly connected with at least one local broker. Each local broker also has direct link with at least one active root-coordinator. Coordinators are replicated to provide efficient and reliable service. We also introduce a centralized control point, the *super node*, into the *PPIB* approach. It is responsible for initiation setting and key management in the whole information brokering process (see section 5).

In PPIB, the *responsibility sharing principle* is implemented for two purposes. (1) To protect user privacy: a user query is divided into access-control-related part (e.g., the *role* of the user and *query content*) and user-privacy-related part(e.g., *authentication information*, *location*, etc). The first part is visible to coordinators for access control enforcement; while the second part is visible to local brokers for authentication. Encryption is adopted so that access-control-related information is not visible to brokers, although they route it to the root-coordinator. (2) To protect data privacy and access control policy privacy: ACRs and data object distribution information are divided and distributed to several coordinators (to be elaborated in 4.2). As a result, PPIB only requires minimal trust (or honesty) in each coordinator, as shown in Figure 3, where "Hide" means "no need to trust". It is clear that whenever the system's level of trust in each brokering component can be lowered without hurting privacy, the system's privacy protection capability will be enhanced.

4.2 Automaton Segmentation

In PPIB, we adopt the view-free automaton-based access control mechanism [13, 15], and extend it in a decentralized manner with our *Automaton Segmentation* scheme. The idea of automaton segmentation comes from the concept of *multilateral security*: split sensitive information to largely meaningless shares held by multiple parties who cooperate to share the privacy-preserving responsibility.

Our automaton segmentation scheme first divides the global access control automaton into several **segments**. Granularity of segmentation is controlled by a parameter *partition size*, which denotes how many XPath states in the global automaton are partitioned and put into one segment. By and large, the granularity is a choice of the system administrator. Higher granularity leads to better privacy preserving, but also more complex query processing. Each accept state of the global automaton is specially partitioned as a separate segment. Then we assign each segment to one independent *site*. As a result, a site in essence holds a small automaton. At run-time, it conducts NFA-based access control enforcement as a stand-alone component. However, in the state transition table of the last state of each segment, the "next state" points to a root state at a remote site, instead of a local state.

In PPIB, a site is actually a *logical* unit. So a *physical* coordinator (i.e., a machine) can in fact hold multiple sites. For convenience, we add *dummy accept states* to each automaton segment. The dummy accept states do not accept queries. Instead, they are used to store the location of actual "next states," i.e. the address(es) of the coordinators who hold the next segment of the global automaton. At runtime, they are used to forward the halfway processed query to the next coordinators. On the other hand, only the sites holding original accept states accept queries and forward them to the data servers. As a result, access control and query brokering are seamlessly integrated at coordinators, and the global automaton-based query brokering mechanism is decentralized and distributed among many coordinators.

```
Algorithm 1 Automaton Segmentation: $Deploy()$
Input: Automaton State $S$
Output: Site Address: $addr$
  for each symbol $k$ in $S.StateTransTable$ do
    $addr = Deploy(S.StateTransTable(k).NextState)$
    $DS = CreateDummyAcceptState()$
    $DS.NextState \leftarrow addr$
    $S.StateTransTable(k).NextState \leftarrow DS$
  end for
  $Site = CreateSite()$
  $Site.addSegment(S)$
  $Coordinator = GetCoordinator()$
  $Coordinator.AssignSite(Site)$
  return $Coordinator.address$
```

In its simplest (and inefficient) form, an access control automaton can be segmented to the finest granularity to best preserve privacy. In this case, each automaton state is divided into one segment and deployed at one site. Algorithm 1 demonstrates a recursive algorithm for finest-granularity automaton segmentation and deployment. As an example, the global automaton shown in Figure 2 is partitioned into 11 segments as shown in Figure 4. For instance, Site 0

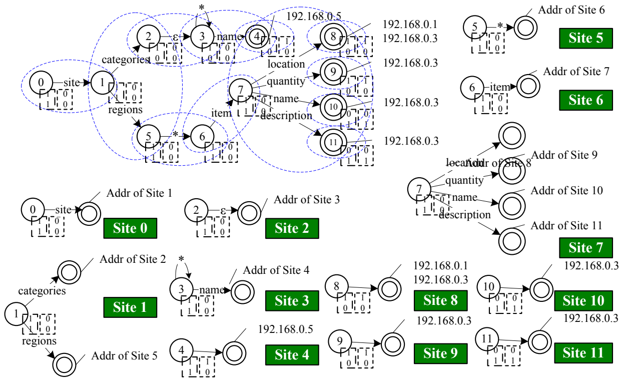


Figure 4: An example to illustrate automaton segmentation scheme.

holds state 0 of the global automaton (symbol "site"); and a dummy accept state which holds the address of Site 1.

Example 3. To illustrate how decentralized automata enforces access control, let us use the query Q: "`/site/regions /asia/item[name='Abacavir']/location`". When Q arrives at Site 0, the first XPath step "/site" is accepted. As the dummy accept state of Site 0 points to Site 1, Q is forwarded to Site 1. Then, the second XPath step "`/regions`" is accepted and the corresponding dummy accept state directs the remaining query to Site 5. There, Site 5 accepts "`/asia`" (wildcard "`*`" matches any input token) and forwards Q to Site 6. At Site 6, element name "`item`" is first accepted. Since the automaton segment does not carry any predicate states, the predicate from Q is kept as it is. Finally, "`/location`" is accepted at Site 7, and Site 10 forwards the query to data server at `192.168.0.3`. Note that, "Abacavir" is a medicine used in AIDS treatment. Therefore, the query Q, as well as related data and metadata, are all highly private and sensitive information. Under the automaton segmentation scheme, metadata privacy is preserved by dividing metadata into multiple sites. In Section 4.4, we will further analyze this example to show how we protect query and data privacy. □

If there are wildcard "`*`" or descendent "`//`" in the query, access control enforcement in the partitioned automaton becomes more complicated. The query may match multiple keywords at a particular site (e.g., the "`/*`" step in a query matches with all tokens in the automaton segment). Thus the query is split into several branches, each of which continues to be processed in the automaton independently. The process is similar to the un-partitioned global automaton. For details, refer to [13].

Distribution and Replication of Automaton Segments In our design, a site is a logical unit which hosts one segment of the global automaton. A physical network peer which holds one or more logical site is called a coordinator. Especially, the coordinator holding the root node of the global automaton is called the root-coordinator; the coordinators with the accept states of the global automaton are called the leaf-coordinators; and the others are called intermediate coordinators.

Sites could be flexibly replicated. For a site S_i, we first make a replication S_i'. Then, for all sites that forward queries to S_i (i.e., whose dummy accept states point to S_i), we change the pointers, re-route some of them to S_i'. For instance, we can create a replication of Site 0 of Figure 4 without changing any other site since it is the root. On the other hand, if we create a replication of Site 1, we need to change the dummy accept state of Site 0 to route a portion of the queries to this replicated site. Moreover, replicates of different logical sites could reside at one physical node. For instance, Site 2 and Site 6 in Figure 4 could be hosted at one coordinator and do not hurt the ACR privacy. This is because the two segments they are holding do not belong to the same rule, and combining them cannot provide any extra hint on ACR.

However, for simplicity, throughout the rest of the paper, we assume that each coordinator only host one site, and we do not consider replications of sites. Therefore, we do not specially distinguish logical sites and physical coordinators anymore, and we will mainly use the term coordinator.

We also need to clarify that our *PPIB* approach supports co-existence of multiple schemas. All access control rules based on the same XML schema are captured in one automaton; and independent automatons are constructed for rules from different XML schemas. Independent automatons could be merged by combining the root coordinators.

4.3 Query Segment Encryption Scheme

To protect user/data privacy that may be revealed by the queries, we propose a *query segment encryption scheme*, which is a good instance that combines data avoidance principle (i.e. encrypting sensitive data) with multilateral security principle (i.e. multiple parties cooperate to take one task, while each party only holds one share of sensitive information).

When an XPath query is being processed at a particular state in the NFA, the query content naturally splits into two parts: XPath steps that has been processed by NFA (accepted or rewritten), and XPath steps to be processed. Although the whole query will be forwarded to the coordinator who holds the next NFA state, NFA will only take the unprocessed steps as input. The idea of *query segment encryption* scheme is to encrypt the processed part of a query so that subsequent coordinators have only an incomplete view of the query content. For encryption, a trusted authority is needed for key distribution and management. In our scheme, this trustee is the *super node*.

The notions used for encryption are defined as follows: both the public and private keys of an XML query are denoted as Pub_Q and $Priv_Q$, respectively; then the corresponding encryption and decryption of string M are denoted as $Encrypt(M, Pub_Q)$ and $Decrypt(M, Priv_Q)$, respectively; for the symmetric encryption scheme, we denote the encryption and decryption applied to message M with secret key K as $E_K(M)$ and $D_K(M)$, respectively.

When an XPath query $Q = s_1 s_2 ... s_n$ first arrives at the root-coordinator, it becomes the input to the automata segment. Suppose automata segment takes s_1, generates s_1' and reaches the dummy accept state (when s_1 is accepted, $s_1' = s_1$, when s_1 is rewritten, $s_1' <> s_1$). The root-coordinator then requests a new Pub_Q from the *super node* and encrypts s_1' as $(E_{K_1}(s_1'), Encrypt(K_1, Pub_Q))$, where K_1 is the secret key of the *root-coordinator*. Both encrypted part and the remaining query $(s_2 ... s_n)$ are forwarded to the next coordinator. If the query passes all intermediate-coordinators and reaches the leaf-coordinator, the whole query will be encrypted as $E_{K_1}(s'_1), E_{K_2}(s'_2), ..., E_{K_n}(s'_n)$. Thus, the entire query content is hidden from the leaf-coordinator.

4.4 Query Brokering Process

Conceptually, the overall query brokering process can be described as four phases.

- In **Phase 1**, an end user sends his/her XML query to the local broker since the user has no idea about data and data server distribution. The broker checks user identity and replaces user authentication information with "role ID". Before sending the query to the broker, user encrypts it with root-coordinator's public key so that the broker cannot see the query content. User also generates a new public key and passes it with the query.
- In **Phase 2**, the broker forwards the encrypted query, user's role ID and public key to the root-coordinator, which is the entrance of the coordinator tree. The broker also encrypts *user location* with symmetric encryption, and attaches its own address to the query. In this way, the broker works in a similar way as an anonymizer in [27] – none of the coordinators is able to infer *who* created the query.
- In **Phase 3**, the encrypted query is recovered by the root-coordinator with its own private key, and then processed in the coordinator-tree, as described in Sections 4.2 and 4.3. Once it arrives at the accept state of any leaf-coordinator, the query is sent to destined data servers. In this phase, query content privacy is protected by the query segment encryption scheme in Section 4.3.
- Finally, in **Phase 4**, data server decrypts the secret keys $(K_1, ..., K_n)$ of the coordinators with the private key from the *super node*, and then decrypts all the encrypted segments $(s_1, ..., s_n)$ of the query with these secret keys. The XML answer is encrypted with user's public key (generated in phase 1) to protect data confidentiality, and returned to the user through local broker.

Example 4. Let us revisit Example 3. The user is asking for the location of item "Abacavir", a medicine typically used in AIDS treatment. Obviously, the user does not want anyone to know that she queried for this item. Moreover, it is a potential risk if others know that the particular data server `192.168.0.3` holds "Abacavir" information. In our *PPIB* framework, the broker only knows user identity, but not the query itself. The root coordinator knows the query, but not the user identity or query location. Other coordinators know only partial contents of the query, but not the user identity or query location. The leaf coordinator knows where the data is located, but it has no hint about the query, i.e., it knows "where", but not "what". We can further examine this example according to Figure 4. As a conclusion, note that all of user, data, and metadata privacies are protected in our proposed PPIB framework. □

5. PPIB MAINTENANCE

Besides routine key management, *PPIB* maintenance is evoked (1) when a brokering component joins/leaves the system, (2) when a data server joins/leaves, and (3) when an access control rule is added/removed. In this section, we describe of system maintenance procedures in these scenarios.

In previous sections, we implicitly assume that there exists a system administrator, who decides issues like automaton segmentation granularity and site distribution. Now we formally introduce it as *administrator node*, which has the highest trust and security level and oversees the whole overlay network. Administrator is needed only when initiating the overlay or when maintenance requests are received. Therefore, the administrator node is not always active.

A. When Brokering Component Joins/Leaves. Brokering components include brokers and coordinators. Before joining, it sends a request to the administrator node (through an existing peer) to wake it up. The administrator checks the security and trust level of the peer, observes the load of all brokers/coordinators in the network, and assigns a role to the new peer. If it is a coordinator, usually, existing sites are moved or replicated to the new host (refer to Section 4.2 for site replication). If the new peer is a broker, it simply replicates an existing broker. Finally, the administrator broadcasts the newcomer to the super node and other related peers. For instance, if the newcomer replicates a root-coordinator, its address is sent to related brokers.

When a peer wants to leave the network, it calls up the administrator with the request. Based on the load of this peer and availability of replications, the administrator could drop the hosted sites, or move some sites to another coordinator. The administrator also informs the super node as well as related peers (e.g. coordinators whose dummy accept states point to the leaving coordinator).

Moreover, to avoid unexpected failures of coordinators, the administrator routinely checks their status. If a coordinator fails, the administrator assigns its task to others and re-routes the related peers. The administrator also balances workload by managing site replications.

B. When Data Server or Object Joins/Leaves. Data location information tells how data is distributed over data servers. When a data server or a data object is added or removed, an update message is created and sent to the administrator for authentication. The message is in the form of $msg(DSAddr, XPath, +/-)$, where $DSAddr$ is the address of the data server, $XPath$ is an XPath expression that refers data objects, and $+/-$ denotes add or removal, respectively.

When a data server/object is removed from the network, administrator first processes the $msg.XPath$ through the automaton to locate related leaf-coordinators, and removes the corresponding indexing rules from them. If a leaf-coordinator does not carry any indexing rule after removal, the corresponding path (from the root-coordinator to the particular leaf) is examined and the sites who does not carry any other rules are suspended.

Example 5. For the coordinator network shown in Figure 4, if we remove data server `192.168.0.5` with the following message:

```
msg(192.168.0.5, /site/categories/category/name, -)
```

then, indexing rule at Site 4 is first removed. Moreover, sites 2, 3, and 4 are suspended since they are not leading to any data. □

When a data server/object is added to DIBS, the administrator locates related leaf-coordinators by processing

$msg.XPath$ through the automata, and assigns $msg.DSAddr$ to them. If the new data server/object affects a suspended branch of the coordinator network, the branch is then activated.

C. When ACR is Added/Removed. Whenever a data owner wants to change access control policy, he sends an updating request to the administrator: $msg\_ac(ACR, +/-)$. When a new access control rule is added to the system, it is sent to the root-coordinator. The XPath expression of the rule go through the automaton until no exact match is found at a certain state. The administrator creates new automaton states for the remaining segments of the new rule. The newly constructed automaton segments are then distributed to the coordinator network. Moreover, related indexing rules are identified and attached to the new leaf-coordinator. The removal of an access control rule also starts from the leaf-coordinator, and goes backward until it reaches a site which also holds keywords from other rules.

6. PRIVACY AND SECURITY ANALYSIS

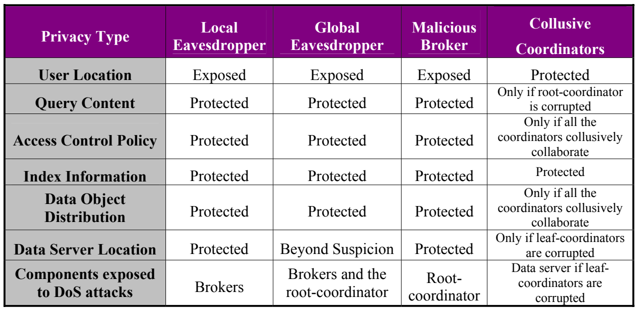
In this section, we consider four types of attackers in an DIBS, and estimate possible damage that the attackers can do to hurt user privacy, data privacy, or metadata privacy. In general, there are various types of attackers. Considering their roles, we can categorize them as malicious insiders and ambitious outsiders; considering their capabilities, as eavesdroppers and power attackers that can compromise any brokering component; considering their working mode, as single attackers or collusive attackers. In this work, we propose a taxonomy of four distinct types of attackers, which covers all aforementioned types of attackers.

A. Local Eavesdropper

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. As a conclusion, an outside attacker who is not powerful enough to compromise brokering components in the system is less harmful to system security and privacy.

B. Global Eavesdropper

A global eavesdropper is an attacker who observes the traffic in the entire network. It watches brokers and coordinators gossip, so it is capable to infer the locations of local brokers and root-coordinators. This is because the assurance of the connections between user and broker, and between broker and root-coordinator. However, from the later-on communication, the eavesdropper cannot distinguish the coordinators and the data servers. Therefore, the major threat from a global eavesdropper is the disclosure of broker and root-coordinator location, which makes them targets of further DoS attack.



Privacy Type	Local Eavesdropper	Global Eavesdropper	Malicious Broker	Collusive Coordinators
User Location	Exposed	Exposed	Exposed	Protected
Query Content	Protected	Protected	Protected	Only if root-coordinator is corrupted
Access Control Policy	Protected	Protected	Protected	Only if all the coordinators collusively collaborate
Index Information	Protected	Protected	Protected	Protected
Data Object Distribution	Protected	Protected	Protected	Only if all the coordinators collusively collaborate
Data Server Location	Protected	Beyond Suspicion	Protected	Only if leaf-coordinators are corrupted
Components exposed to DoS attacks	Brokers	Brokers and the root-coordinator	Root-coordinator	Data server if leaf-coordinators are corrupted

Figure 5: The possible privacy exposure caused by four types of attackers.

C. 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 attack.

D. 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.

Finally, we summarize the possible privacy exposure in Figure 5.

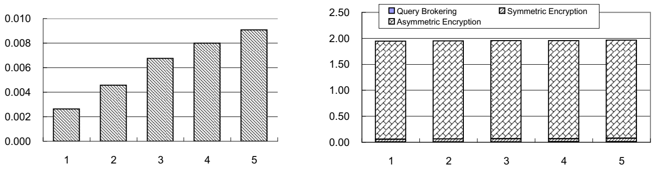
7. PERFORMANCE ANALYSIS

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 (3.4G CPU). We use the XMark [23] 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.

7.1 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

X: Number of keywords at a query broker; Y: Time (ms)

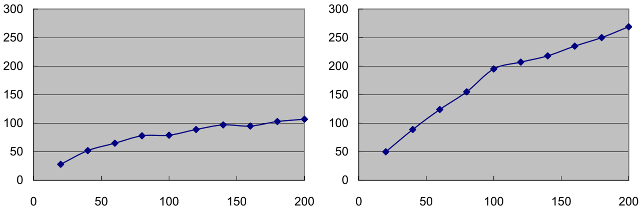


(a)Average query brokering time at a coordinator.

(b)Average symmetric and asymmetric encryption time.

Figure 6: Average query processing time at a coordinator (T_C).

X: number of access control rules; Y: number of coordinators.



(a)simple path rules; (b) rules with wildcards in the XPath.

Figure 7: System scalability against ACR complexity.

consider the following four components: (1) average query 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, T_E will remain the same (at seconds level [14]). 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, T_E and $T_{backward}$ are not affected by the broker-coordinator overlay network. We only need to calculate and compare the total forward query processing time ($T_{forward}$) as $T_{forward} = T_C \times N_{HOP} + T_N \times (N_{HOP} + 1)$. It is obvious that $T_{forward}$ is only affected by T_C, T_N, and the average number of hops in query brokering, N_{HOP}.

Average query processing time at a coordinator: Query processing time at each broker/coordinator (T_C) consists of: (1) access control enforcement and locating next coordinator (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).

To examine T_C, we manually generate five sets of access control rules. Access control rules of each set are partitioned into segments (keywords), which are assigned to coordinators. From set 1 to set 5, the number of keywords held by one coordinator increases from 1 to 5. We also generate 1000 synthetic XPath queries, and use Triple DES for symmetric encryption and RSA for asymmetric encryption. Figure 6(a) shows that query brokering time is at milliseconds level, and increases linearly with the number of keywords at a site. Figure 6(b) shows that symmetric and asymmetric encryption time is at seconds level, and asymmetric encryption time dominates the total query processing time at a coordinator. As a result, average (T_C) is about *1.9 ms*. Query processing time at brokers and leaf-coordinators are shorter but still in the same level. For simplicity, we adopt the same value (i.e. *1.9 ms*) for the average query processing time at brokers and coordinators.

Average network transmission latency: We adopt average Internet traffic latency *100 ms* as a reasonable estimation of T_N [1], instead of using data collected from our gigabyte Ethernet.

Average number of hops in query processing: We consider the case in which a query Q is accepted or rewritten by n ACRs $\{R_1, ..., R_n\}$ into the union of n safe sub-queries $\{Q'_1, ..., Q'_n\}$. When an accepted/rewritten sub-query Q'_i is processed by the rule R_i, the number of hops it experiences is determined by the number of segments of R_i. In the experiment, we generate a set of 200 synthetic access control rules and 1000 synthetic XPath queries.We choose the finest-granularity automaton segmentation (each XPath step of an ACR is partitioned as one segment and kept at one coordinator) for maximum privacy preserving. Our experiment result shows that N_{HOP} is 5.7, and the maximum number of hops of all queries is 8.

Total forward end-to-end query processing time: from above experiment results, the total forward query processing time is calculated as $T_{forward} \simeq 1.9 \times 5.7 + 100 \times (5.7 + 1) \simeq 681(ms)$. It is obvious that network latency $T_N * (N_{HOP} + 1)$ dominates total forward end-to-end query processing time, because the value of T_C is negligible compared with T_N. Moreover, since T_N remains the same (as an estimation from Internet traffic), 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.

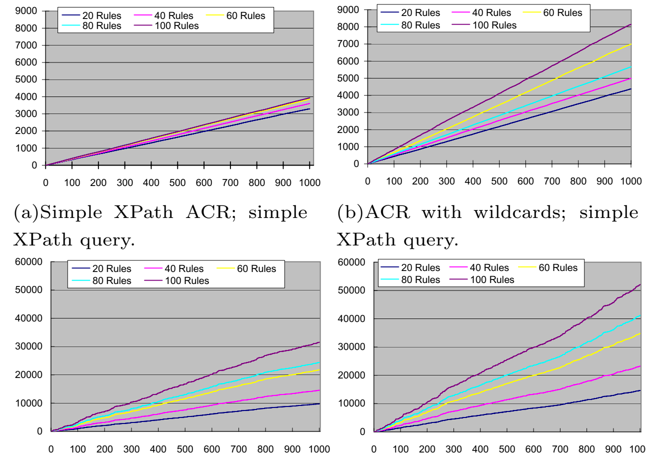
7.2 System Scalability

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).

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. As shown in Figure 7 (also adopting the finest granularity automaton segmentation), 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.

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 to measure the system load. When a query is accepted as multiple sub-queries, all sub-queries are counted towards system load.For a query that is rejected after i segments, the processed i segments are counted.

X: number of queries in a unit time; Y: number of total segments in the system.



(a)Simple XPath ACR; simple XPath query.

(b)ACR with wildcards; simple XPath query.

(c)ACR with wildcards; query with 5% wildcard probability at each XPath step.

(d)ACR with wildcards; query with 10% wildcard probability at each XPath step.

Figure 8: System scalability against queries.

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. Figure 8 shows system load vs. number of XPath queries in a unit time. More specifically, Figure 8 (a) only has simple path rules (without wildcard or predicate), and Figure 8(b) has rules with wildcards. In both cases, system load increases linearly and each query is processed less than 10 segments. Figure 8(c) and (d) use the same set of ACRs as in (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 sub-figure, we can see that larger ACR leads to higher system load, but the increase appears to be linear in all cases.

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 number of indexing entries at leaf-coordinators. However, in PPIB, query indexing is implemented through hash tables, which is scalable. Thus, the system is scalable when data size increases.

8. RELATED WORK

Research areas such as information integration, Web search, peer-to-peer file sharing systems, and publish-subscribe systems provide partial solutions to the problem of large scale data sharing. Information integration seeks to provide an integrated view over large numbers of heterogeneous data sources by exploiting the semantic relationship between schemas of different sources [7, 16, 9]. It turns out that the *PPIB* approach will facilitate but is orthogonal to the information integration technology. On the other hand, Web search focuses on locating data sources with high precision and coverage [26, 18]. However, it only supports keyword queries with limited expressiveness.

Peer-to-peer systems are designed to share files and data sets (e.g. in collaborative science applications). Distributed hash table technology [25, 8] is adopted to locate replicas based on keyword queries. However, although these technologies have recently been extended to support range queries [21], the coarse granularity (e.g. files and documents) still makes them short of our expressiveness needs. Further, P2P file-sharing systems may not provide complete set of answers to a request while we need to locate all relevant data.

Addressing a conceptually dual problem, the XML publish-subscribe systems (e.g. [3, 6]) is probably the closely related technology to our proposed research: while we locate relevant data sources for a given query and route the query to these data sources, the pub/sub systems locate relevant consumers for a given document and route the document to these consumers. However, due to this duality, we have different concerns: they focus on efficiently delivering the same piece of information to a large number of consumers, and we are trying to route large volume but small-size queries to many fewer sites. 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 on top of normal IP network. In particular, specialized data structures are maintained on nodes of the overlay networks to route path queries. In [24], a robust mesh has been built to effectively route XML packets by making the use of self-describing XML tags and the overlay networks. Kouds et al. [12] describes a decentralized architecture for ad hoc XPath query routing across a collection of XML databases using the open and agreement cooperation models. In [10], content-based routing of path queries in peer-to-peer systems is studied to serve the purpose as sharing data among a large number of autonomous nodes. The main difference between these approaches and ours is that they focus on distributed query routing, while we seamlessly integrate query routing and security checking (e.g. access control) so as to preserve relevant privacy information.

As long as privacy becomes an important information that should be protected from unauthorized entities, several approaches have been designed to preserve anonymity in communication. Crowds [19] is a distributed and chained Anonymizer (http://www.anonymizer.com), where users are grouped dynamically and issue request on behalf of a crowd member. In [27], sender anonymity is guaranteed by building up anonymous connections among *Onion Routers* using Chaum Mix. Since we integrate security checking enroute that involves more privacy concerns other than anonymity, our privacy addresses more challenge.

As for security check, many researches have been proposed on distributed access control (see [28] for a good overview on access control in collaborative systems). Earlier approaches implement access control mechanisms at the nodes of XML trees and filter out data nodes that users do not have authorizations to access [4, 17]. These approaches rely much on the XML engines. View-based access control approaches create and maintain a separate view (e.g. a specific portion of XML documents) for each user [20, 29]. However, supporting large number of views causes high maintenance and storage cost. Our *PPIB* approach adopts a recently proposed NFA-based query re-writing access control scheme [15, 13] and extends it to a decentralized manner. It has a better performance compared with [17], and any off-the-shelf XML databases can be used due to its query re-writing nature.

9. CONCLUSION AND FUTURE WORK

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, site distribution and replication. A main goal is to make *PPIB* self-reconfigurable.

Acknowledgements

The ideas in this paper were refined during discussions with Ting Yu, Wang-Chien Lee, Prasenjit Mitra, and Michael Rabinovich. The anonymous reviewers provided valuable feedback that helped improve the paper's quality. This research was supported in part by NSF CCR-0233324 and NSF/DHS 0335241.

10. REFERENCES

[1] Internet traffic report. http://www.internettrafficreport.com.

[2] R. Agrawal, A. V. Evfimievski, and R. Srikant. Information sharing across private databases. In *SIGMOD*, pages 86–97, 2003.

[3] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *Proc. of INFOCOM*, 2004.

[4] S. Cho, S. Amer-Yahia, L. V. S. Lakshmanan, and D. Srivastava. Optimizing the secure evaluation of twig queries. In *VLDB*, pages 490–501, China, 2002.

[5] E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *ACM Trans. Inf. Syst. Secur.*, 5(2):169–202, 2002.

[6] Y. Diao, S. Rizvi, and M. J. Franklin. Towards an Internet-scale XML dissemination service. In *VLDB*, Toronto, 2004.

[7] M. Genesereth, A. Keller, and O. Duschka. Informaster: An information integration system. In *SIGMOD*, Tucson, 1997.

[8] R. Huebsch, B. Chun, J. Hellerstein, B. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. Yumerefendi. The architecture of pier: an internet-scale query processor. In *CIDR*, pages 28–43, 2005.

[9] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *SIGMOD*, pages 205–216, 2003.

[10] G. Koloniari and E. Pitoura. Content-based routing of path queries in peer-to-peer systems. In *EDBT*, 2004.

[11] G. Koloniari and E. Pitoura. Peer-to-peer management of xml data: issues and research challenges. *SIGMOD Rec.*, 34(2):6–17, 2005.

[12] N. Koudas, M. Rabinovich, D. Srivastava, and T. Yu. Routing xml queries. In *IEEE ICDE*, page 844, 2004.

[13] 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.

[14] H. Lu, J. X. Yu, G. Wang, S. Zheng, H. Jiang, G. Yu, and A. Zhou. What makes the differences: benchmarking xml database implementations. *ACM Trans. Inter. Tech.*, 5(1):154–194, 2005.

[15] B. Luo, D. Lee, W.-C. Lee, and P. Liu. QFilter: Fine-grained run-time XML access control via NFA-based query rewriting. In *ACM CIKM*, Washington D.C., USA, nov 2004.

[16] I. Manolescu, D. Florescu, and D. Kossmann. Answering xml queries on heterogeneous data sources. In *VLDB*, pages 241–250, 2001.

[17] M. Murata, A. Tozawa, and M. Kudo. XML access control using static analysis. In *ACM CCS*, Washington D.C., 2003.

[18] S. Park, A. Khrabrov, D. M. Pennock, S. Lawrence, C. L. Giles, and L. H. Ungar. Static and dynamic analysis of the internet's susceptibility to faults and attacks. In *IEEE Infocom*, 2003.

[19] M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[20] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *SIGMOD*, pages 551–562, Paris, France, 2004.

[21] O. Sahin, A. Gupta, D. Agrawal, and A. E. Abbadi. A peer-to-peer framework for caching range queries. In *Proc. of the 20th Int. Conf. on Data Engineering*, 2004.

[22] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[23] A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, and R. Busse. "The XML Benchmark Project". Technical Report INS-R0103, CWI, April 2001.

[24] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh-based content routing using XML. In *Symposium on Operating Systems Principles*, pages 160–173, 2001.

[25] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *IEEE/ACM Trans. Networking*, volume 11 of *1*, 2003.

[26] A. Sugiura and O. Etzioni. Query Routing for Web Search Engines:Architecture and Experiments. *Computer Networks*, 33(1), 2000.

[27] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 4–7 1997.

[28] W. Tolone, G.-J. Ahn, T. Pai, and S.-P. Hong. Access control in collaborative systems. *ACM Comput. Surv.*, 37(1), 2005.

[29] T. Yu, D. Srivastava, L. V. S. Lakshmanan, and H. V. Jagadish. Compressed accessibility map: Efficient access control for XML. In *VLDB*, pages 478–489, China, 2002.