

# Defending against Attribute-Correlation Attacks in Privacy-Aware Information Brokering

Fengjun Li<sup>1</sup>, Bo Luo<sup>2</sup>, Peng Liu<sup>1</sup>, Anna C. Squicciarini<sup>1</sup>, Dongwon Lee<sup>1</sup>,  
and Chao-Hsien Chu<sup>1</sup>

<sup>1</sup> College of Information Science and Technology, The Pennsylvania State University

<sup>2</sup> Dept. of Electrical Engineering and Computer Science, The University of Kansas

**Abstract.** Nowadays, increasing needs for information sharing arise due to extensive collaborations among organizations. Organizations desire to provide data access to their collaborators while preserving full control over the data and comprehensive privacy of their users. A number of information systems have been developed to provide efficient and secure information sharing. However, most of the solutions proposed so far are built atop of conventional data warehousing or distributed database technologies.

Recently, information brokering systems have been proposed to provide privacy-preserving information sharing among loosely federated data sources. However, they are still vulnerable to attribute-correlation attacks during query routing, due to the lack of protection of the routed queries. In this paper, we investigate the problems caused by such an attack, and propose a countermeasure by limiting the view of query content at each intermediate broker. We show that the proposed content-based XPath query routing scheme with level-based encryption and commutative encryption can effectively prevent an attribute-correlation attack originated by compromised brokers, with reasonable overhead.

**Keywords:** information brokering, attribute correlation attack, privacy, XML.

## 1 Introduction

Today's organizations often operate across organizational boundaries. They raise strong needs for efficient and secure information sharing to facilitate extensive collaborations. However, early approaches on information sharing, which mainly focus on providing transparency and interoperability among heterogeneous system, fall short of satisfying new requirements of these inter-organizational collaborations.

To better understand such requirements, we overview the unique needs of such interorganization collaboration by considering an example in the healthcare domain. Large-scale health information infrastructures, such as Regional Health Information Organization (RHIO), are being developed to share medical information (e.g., patient records) collected by collaborative health providers (e.g., hospitals) via protected "channels". First, there is no centralized authority to

coordinate the data in different hospitals. Each health provider is authorized by its patients to collect medical information independency, and stores it across multiple local data servers. Since the data is private and sensitive, the health providers are responsible for not leaking patient records to irrelevant parties. The health providers desire to share their data to fulfill collaboration, however, they prefer to do it in a restricted and controlled fashion. Data requestors, such as doctors, need to be able to retrieve the medical records with precision and not be distracted by “noisy” data. Finally, the RHIO should be able to maintain a large number of data servers, considering the participant population. In general, such interorganization collaboration application requires an information sharing system that offers full autonomy to underlying databases, preserve data security and privacy comprehensively, and provides good scalability.

Recently, information brokering systems (IBS) [8,13] have been proposed to meet the above requirements. In an IBS, geographically distributed data sources within a consortium are linked through a set of brokers to provide unified data access to all the users in the consortium. No third party is required to keep a centralized copy of data from local databases. Each data source maintains full control over its data and has great flexibility to grant, restrict, and revoke access of a particular user to certain data. Moreover, in order to achieve the desired query expressiveness, XML data model is widely adopted. Brokers are linked in a peer-to-peer fashion that makes IBS a scalable system.

Although IBS systems meet most of the requirements in current interorganization information sharing, they suffer from many attacks to privacy. In [13], we gave an insightful analysis of types of privacy involved in on-demand distributed information access as well as the risks. We proposed an automaton based approach for comprehensive privacy protection, however, it still suffered from a major privacy threat. In databases, data is represented as a set of records and each record can be viewed as a set of *attributes* with distinct values. Thus, the content of a query may be simply viewed as a sequence of query conditions (or predicates): each condition/predicate involves a specific attribute. Although the queries are sent through secure tunnels, an intermediate broker can view every XML query routed through it. If no proper privacy control is enforced, an en-route broker, when compromised or turned into a malicious insider, could easily extract query conditions and *correlate* the attributes to infer sensitive information about the data owner. The attack is known as *attribute-correlation attack*. Moreover, the results from attribute-correlation attacks may facilitate further inferences such as the re-identification attack.

In this paper, we define the attribute-correlation attack in information brokering systems, and provide solutions. More specifically, we design a content-based query routing scheme which uses encryption scheme to protect query content from attribute-correlation attacks by limiting broker’s view of query content. Through insightful analysis, we show that no private information leaks even if some brokers collude. To the best of our knowledge, this is the first solution that protects an IBS against attribute-correlation attacks launched by compromised brokers.

The rest of the paper is organized as follows. We first summarize the related work in Section 2. Then, in Section 3, we briefly introduce IBS and the attribute-correlation attack problem in IBS. We propose the details of our scheme in section 4, including broker tree construction, content-based routing, and level-based and commutative encryptions. Privacy and performance analysis is given in section 5. And we conclude in Sections 6.

## 2 Related Work

A number of information integration approaches had been developed to support business applications since 1980's. However, most of the early approaches focus on providing transparency and interoperability among heterogeneous system but neglect the needs for autonomy, scalability and privacy-preserving.

In early 90's, Wiederhold proposed the well-known mediator architecture [16,17], in which an additional layer of mediators was added between clients and back-end databases. The mediators, exploiting encoded knowledge about data, execute and enforce regulations over both query and data bidirectionally. However, the mediator approach was not designed for large scale federations, so it is only practical for systems with a small number of components.

The data warehousing and data store approaches [10,5] provide another way for information integration and sharing. However, they all request to pour the data into a centralized repository so as to provide a global view to the user. Therefore, these approaches fall short to meet the autonomy needs of individual organizations.

Federated database systems have the requested capabilities of autonomy and supporting large-scale data distribution. However, the current designs neglect comprehensive privacy considerations.

Other approaches, such as pub/sub approaches in [3,18], mesh-based overlay network proposed in [15], and peer-to-peer (P2P) systems that support content-based routing of XPath queries [6,11], all implement the *information push* model, thus they are not suitable for the applications focused in this paper, in which information access is typically in the *information pull* model.

## 3 Information Brokering Overview

An IBS system is a peer-to-peer overlay network consisting of data owners, brokers and end users (i.e. data requestors). This type of architecture is employed in an interorganization scenario, where multiple organizations have strong needs of cross-organizational information sharing.

More specifically, we define IBS as agent-based systems across loosely federated XML or XML-supported databases. XML data model and XML query languages are adopted to achieve the desired query expressiveness. Local databases are autonomous systems geographically distributed across multiple domains. Certain agents, namely brokers, are employed to link data sources (i.e. databases) and data requestors, where data sources and data requestors of each domain are

connected to a local broker. Then, brokers are linked in a peer-to-peer fashion to provide transparent data access to data requestors. The flexible topology of broker overlay may ease the on-and-off maintenance as well as reduce the cost when the number of databases scales.

In this work, we focus on read-only querying access to various data sources, which enables a data requestor with no priori knowledge about the requested data distribution to send a query to a local broker and to receive answers from data sources sitting in different organization domains.

To guide a query to data sources with the requested data, a broker(s) needs certain knowledge about data distribution. Such knowledge is represented as *routing rules*, describing which data source (i.e. `location`) holds which data objects.

In a distributed setting, regardless of the network topology and routing protocol adopted by the IBS [12], one broker only holds a partial set of all routing rules. Thus, multiple brokers should collaborate in routing an XML query to its destined data source(s).

### 3.1 The Attribute-Correlation Attack

A major function of an IBS is to route XML queries from data requestors to relevant data sources by means of routing rules. Routing rules are metadata of the form  $R = \{\text{subject}, \text{location}\}$ , where `subject` is an XPath [4] expression denoting a set of data objects and `location` is a list of IP addresses<sup>1</sup>. Routing rules could be expressed at different level of specificity. Two example routing rules are shown as follows.

```
Rule1: {/recordTarget//patient//*, {206.132.1.18, 206.132.1.19}},
Rule2: {/ClinicalDocument//Date[@value='041207']//*, 206.132.1.110}.
```

To route a query, a broker extracts the XPath expression from the `subject` field and compares with its routing rules. If the XPath expression matches one of its routing rules, the broker will forward the query to the address(es) in the `location` field of the routing rule; otherwise, the broker will deny the routing request and drop the query. Obviously, in order to fulfill the routing task, the involved brokers should be allowed to view the query in clear-text.

The main body of a query is an XPath expression consisting of a sequence of steps. Each step is an axis specification followed by a node test and a predicate (optional). Two steps are separated by a “/”. A query always contains one or several predicates in some of its steps, which act as query conditions. Each predicate involves a specific attribute that may represent sensitive and private data of its owner (e.g. name, SSN or credit card number, etc.). If there are more than one predicate in a query, one can associate the corresponding attributes to infer sensitive information about the data owner. This attack is known as *attribute correlation attack*.

<sup>1</sup> In this work, we adopt XPath as a format to express XML queries and routing rules. XPath is a restricted variation of regular path expressions, which can refer to all or part of the nodes in an XML document. However, our system is applicable to any regular path expression and any query language based on it.

Next, we elaborate on the vulnerability with an example. Assume a data requestor, doctor Bob, requests all medical records that are relevant to blood cancer of a patient named Alice. Bob creates an XML query:

```
q=/report[//patient/@name='Alice']//code[@displayname='BloodCancer']/*.
```

This query has two query conditions (i.e. `name = 'Alice'` and `displayname = 'BloodCancer'`) on attributes of XML nodes `patient` and `code`, respectively. Without further protection, one could easily extract the two query conditions and correlate the two attributes to infer “Alice has blood cancer”, a highly sensitive information of Alice.

The problem of correlated attributes in XML query was not considered as severe in early information brokering systems, due to the fact that query content is only viewable to intermediate brokers that are typically assumed trusted. However, as large amount of local data sources join the system, this assumption becomes more and more weak. When joining the system, a member organization either contributes with its own servers to be used as local databases or local brokers, or it delegates it to a third party. It is unrealistic to assume full trust to the brokers, especially if the IBS is used for privacy-sensitive applications.

To defend against the problem caused by attribute correlation, our goal is to limit or at least minimize the capability of any intermediate broker’s view of non-empty statement in the sub-queries.

## 4 New Privacy-Preserving IBS Design

The most intuitive approach to address the problem of attribute-correlation problem is to rely on trustworthiness of the brokers. It is believed that the chance of the attacker will be greatly reduced if we limit the interactions only among trusted brokers, which are less likely to be malicious or compromised. However, fully trusting a broker either exposes to unexpected risks in case it gets compromised or it introduces expensive costs on continuously monitoring its status. Moreover, the set of brokers trusted by any given organization needs to capture a complete copy of routing rules in order to fulfill the routing task. In the worst case, when every single organization only trusts its own broker, each broker needs to store all routing rules. As a conclusion, the trust-based solution is neither practical nor scalable.

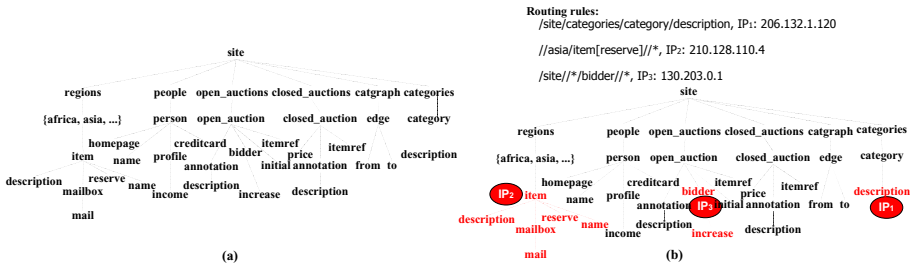
A more effective solution is to let multiple entities share responsibility for query routing, so that the trust assumption on each entity could be lowered to a more reasonable level. We assume the brokers are semi-honest, i.e. brokers will faithfully obey the rule but curiously infer the private data as much as they can.

Our approach is to split the routing responsibility into multiple brokers so that they can fulfill the routing task by cooperating together, but a single broker is no longer capable of launching the attribute-correlation attack. More specifically, our approach is characterized by two main components: (1) constructing a broker-tree overlay atop of current topology by decomposing each routing rule

into segments and assigning each routing segment to one broker; and (2) splitting each query into segments accordingly and wrapping each segment with a special key. We present the details in the following subsections.

#### 4.1 Broker Tree Construction

Since our work is orthogonal to information integration, we simply assume a global integrated schema (e.g. HL7) is shared among all organizations in a consortium. The schema is a structural description of the syntax of XML documents. If we model the schema as a tree, every node denotes an element or attribute of the schema and the edge between two nodes represents the parent-child relationship. Taking XMark<sup>2</sup> as example, its schema graph is shown in Figure 1(a).



**Fig. 1.** (a) The schema graph of XMark DTD; (b) Filtering routing rules against the schema

Routing rules are collected from distributed data sources (by means of query searching) based on the global schema. Considering the XPath expression in the **subject** field of a routing rule, it denotes a set of nodes in XML documents. We can locate a subtree by filtering the **subject** field of a routing rule against the global schema. Then, we attach the address(es) in the **location** field to the root of the subtree. In this way, we create a centralized *routing schema* for query routing, similar as in [13]. An example in Figure 1(b) shows that the addresses (i.e.  $IP_1$ ,  $IP_2$  and  $IP_3$ ) are attached to the roots of three subtrees (i.e. nodes **description**, **item** and **bidder**) when three routing rules are filtered against the XMark schema.

Next, we divide the routing schema into multiple sub-trees (namely *routing segments*) in a way that every leaf-node of a subtree points to the root of its child subtree. We allow flexible granularity in schema dividing : a more fine-grained schema dividing results in less nodes in a routing segment, which indicates well-protected privacy but increased maintenance cost. For each broker, we assign a unique routing segment to it, and attach its address to the leaf-nodes of the parent routing segments. In this way, the brokers are connected in a tree structure

<sup>2</sup> XMark is an XML benchmark modeling data from Internet auction applications with a single DTD. We use it as an example in the following discussions.

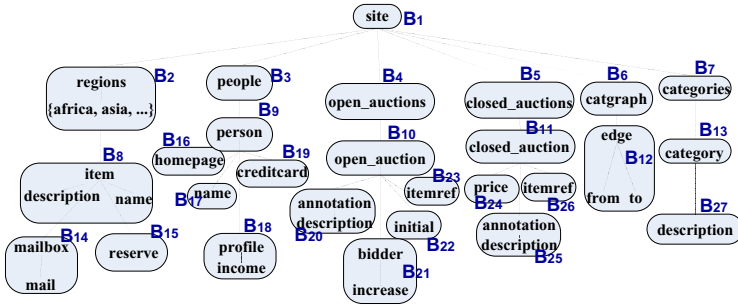


Fig. 2. A broker-tree is constructed with 27 subtrees

according to the relationship of their routing segments. Figure 2 shows a broker tree consists of 27 routing segments based on the routing schema in Figure 1(b).

Since routing rules are split into segments in a broker tree, several brokers need to cooperate to complete the routing task. Let us denote by  $q=\{q_{seg_1}||\dots||q_{seg_n}\}$  an XML query with  $n$  segments. The content based routing process can be simply described as follows. The query  $q$  is first sent to the root broker  $B_1$ , where  $q_{seg_1}$  is processed against the routing segment at  $B_1$ . If the routing segment matches with  $q_{seg_1}$ , the query will be forwarded to the broker whose address is attached at  $B_1$ ; otherwise, the query will be dropped. The same process follows until the query is dropped by some broker or reaches the final data source.

### 4.2 Query Segment Encryption

As introduced in the previous section, the first step of our solution consists of sharing the routing responsibility among multiple brokers by constructing a broker tree. Next, we design query segment encryption schemes to guarantee each broker can only decrypt one query segment of a encrypted query. More specifically, for any query segment  $q_{seg_i}$  of  $q=\{q_{seg_1}||\dots||q_{seg_n}\}$ , the responsible broker  $B_i$  is only allowed to view  $q_{seg_i}$  in clear-text, while both the processed segments  $\{q_{seg_1}||\dots||q_{seg_{i-1}}\}$  and the unprocessed segments  $\{q_{seg_{i+1}}||\dots||q_{seg_n}\}$  are still encrypted beyond the capability of  $B_i$ .

We propose a *level-based encryption* scheme for general cases. As we return later in the paper, this solution is not yet satisfactory, since the wildcard “/” introduces mismatch among the level keys. As such, a new *commutative encryption* scheme is proposed to solve this problem.

**Level-based Encryption Scheme.** A simple solution to meet the encryption requirement is to encrypt each query segment with the public key of the responsible broker. For example, assume an input query  $q=/site/regions/asia/item[id=10028]/name$ . If  $q$  is processed through the broker tree in Figure 2, the query segments  $/site$ ,  $/regions/asia$ , and  $/item[id=‘10028’]/name$  will be processed by brokers  $B_1, B_2$ , and  $B_8$ , respectively. So, we can encrypt the three query segments, each with the related public key.

However, a fundamental problem in the proposed PKI-based solution occurs since we assume no centralized routing authority exists after offline broker tree construction process. For any query, the route through the broker tree is unpredictable since neither the brokers enroute nor their public keys to decrypt portions of the route are known.

Therefore, we propose a more effective solution, which encrypts query segments while mitigating the dependency of routing, namely *level-based encryption scheme*. Instead of assigning a pair of public and private keys to each broker, we assign them to all the brokers in the same *level*<sup>3</sup>. The concept of level is defined as the distance from a node to the root of the tree. Taking the broker tree in Figure 2 as an example, brokers  $B_2, B_3, B_4, B_5, B_6$ , and  $B_7$  are all belonging to level 1. Moreover, if a routing segment contains nodes belonging to more than one levels, all the relevant private level key will be assigned to that broker. Since an XPath expression in an XML query is a location path consisting of a sequence of steps, we will encrypt the XPath steps with corresponding public level keys, e.g. encrypting the  $i$ th step with the public key of level  $i$ .

**Commutative Encryption Scheme.** The level-based encryption scheme works well unless the input query contains the *descendant-or-self* axis in its XPath expression. According to the level-based encryption scheme, a broker of level  $i$  can view and only view the  $i$ th XPath step of any given query. However, if a *descendant-or-self* axis (denoted as “//”) shows at the  $i$ th XPath step, the brokers behind level  $i$  may have chance to view some steps after the ones they are authorized to view. We refer to this issue as the *mismatching problem*.

We employ a simple example (as shown in Figure 3(a)) to further elaborate on the mismatching problem. We denote by  $E_i$  the encryption process with public level key  $Pu_i$  and  $D_i$  the decryption process with private level key  $Pr_i$ . The input query is  $Q_2 = /site//item/name$ . Three XPath steps of  $Q_2$  are encrypted with the public keys of level 1, level 2, and level 3, respectively. If we send  $Q_2$  into the global schema tree of Figure 1, the XPath step “/name” is an XPath node of level 5, which should be processed by broker  $B_8$  along the path “/site/regions/{africa|asia|...}/item/name” in Figure 2. However, following the level-based encryption scheme, broker  $B_2$ , with the private level keys of level 2 and 3, will first decrypt the XPath step “//item” at the node “regions” and add a step “regions” to the original query. Then, it will decrypt the XPath step “/name” at the node “/{Africa|Asia|...}” and add another step “Asia” to the query. As a result,  $B_2$  will uncover the XPath step (i.e. /name) that it is not authorized to access.

To tackle this problem, we propose a new encryption scheme based on well-known *commutative encryption algorithms* [2,7,14]. Commutative encryption is a collection of algorithms that have the property of being *commutative*. In short, an encryption algorithm  $E(\cdot)$  is commutative if for any two keys  $e_1$  and  $e_2$ ,  $E_{e_1}[E_{e_2}[m]] = E_{e_2}[E_{e_1}[m]]$ , where  $m$  is the message to be encrypted. We adopt

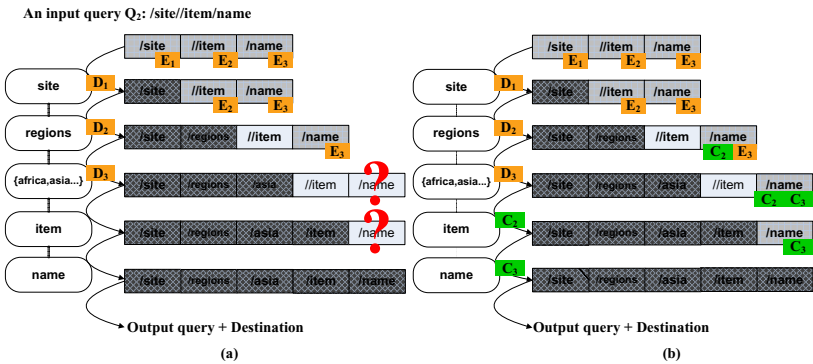
<sup>3</sup> We adopt the broadcast encryption scheme [9,1] to create public and private level keys so that the public key of any given level is generated based on the private keys of all the brokers at this level.



Pohlig-Hellman exponentiation cipher with modulus  $p$  as our commutative encryption function.

We employ the commutative encryption algorithm in order to make flexible switching of decryption sequence possible. The commutative-based encryption scheme introduces a commutative symmetric key for each level, namely *commutative level key*  $C_i$ . Besides being issued to brokers of level  $i$ ,  $C_i$  is also issued to all the brokers at level  $C_{i+2}$ . The public and private level keys  $Pr_i$  and  $Pu_i$ , as defined in level-based encryption scheme, are also defined and assumed to be commutative. Moreover, a pointer  $p$  is introduced to indicate the XPath step to be processed by the current broker. A broker will always decrypt the XPath step marked by the pointer with its private level key, and move the pointer to the next XPath step.

*Commutative encryption scheme* is an improver for the level-based encryption scheme. In general, the brokers process XPath steps of a query in the same way as they do following the level-based scheme. When encountering a step with wildcard “//”, and the token in wildcard step does not match with the one in its routing segment, the broker will launch the special commutative encryption process, which is summarized as follows: first, the broker starts the set wildcard processing stage by setting a flag  $f = 1$ . Then, it encrypts all the following unprocessed XPath steps with its commutative level key. With respect to the example in Figure 3(a), the XPath step “/name” is encrypted with key  $C_2$  at the node “regions” of  $B_2$ . The following brokers compare the wildcard token with the one in its routing segment. If they do not match, the broker, says  $B_j$ , will apply two symmetric commutative keys onto all the XPath steps in the query. This additional encryption guarantees every unprocessed XPath segments are protected by the commutative keys of the current level and its upper level. If they match with each other,  $B_j$  will set  $f = 2$  to indicate post-wildcard processing. In this stage, a broker encrypts all the unprocessed steps with  $C_j$  and decrypts them with  $C_{j-2}$ . As shown in Figure 3(b),  $B_2$  wraps unprocessed query step /name with the commutative level key  $C_2$  and  $C_3$ . Then, when  $B_8$  finds “//item”



**Fig. 3.** Example of (a) the mismatching problem caused by the descendent-or-self axis in a query; (b) the solution based on commutative encryption scheme

matches with the token in its routing segment, it decrypts `/name` with  $C_2$  and  $C_3$ , accordingly.

## 5 Analysis

The main purpose of this work is to protect the privacy of the data owners while authorized organizations collect the data from them and share with other collaborators. More specifically, we protect the content of the query from the malicious or compromised intermediate servers during information brokering process. Therefore, we evaluate our system with two metrics: the *privacy* preserved with the proposed schemes, and the overhead introduced in the querying access of distributed data.

### 5.1 Privacy Analysis

If the honest-but-curious assumption about the brokers holds, the proposed level-based and commutative encryption schemes can guarantee that no intermediate broker views the complete query content while the brokers fulfilling the query routing function. Risks exist only when one or multiple brokers are abused by the insiders or compromised by the outside adversaries.

The threats caused by malicious or compromised brokers depend on the number of hostile brokers as well as their relative positions in the schema tree. Here, we classify possible cases as the threat under *one hostile broker* and *collaborate brokers*, and briefly discuss as follows.

**One hostile broker.** In the proposed IBS, a broker is assigned with a routing schema, which is part of the global schema tree, and with level keys (i.e. one private level key and two commutative level keys). With the level keys, a hostile broker can always uncover the corresponding XPath step(s) of an input query. However, the XPath step may at most contain one attribute. Thus, a single hostile broker cannot conduct attribute correlation attack with the restricted view of the query content.

However, a hostile broker may locate the relative position of the routing schema it holds in the global schema tree by looking it up in the global schema tree. From the relative location, it can further guess the routing schemas of the brokers right before and next to it. This information itself is not sensitive, although it may help colluded brokers to determine the next-step target to be compromised.

**Collusive brokers.** Hostile collusive brokers at different level are capable to uncover different XPath steps of an input query. Whenever a hostile broker receives a query, it can intercept the query and forward it to colluded brokers for decryption. In this case, the compromised broker will view multiple segments of the query instead of one, so its chance to successfully launch the attribute correlation attack increases. In our approach, the risk is however still restricted due to two reasons: first, when a query is intercepted by a broker, only its unprocessed part is at risk since processed XPath steps of the query are encrypted

using preassigned public keys of the data servers. So the attacker’s chance to succeed depends on the position of the hostile broker that first intercepts the query. If the hostile brokers are near the leaf brokers, the unprocessed query segments are very limited. Therefore, possible countermeasure is to strategically assign the routing schemas of higher levels to more trusted brokers. Secondly, the risk is also related to the number of levels that collusive brokers can cover. Since an attacker may not compromise all the brokers in a limited time interval, his view of the query content is still incomplete.

## 5.2 Performance Analysis

The overhead introduced by our routing scheme mainly includes computational cost and communication overhead. The former denotes the cost introduced by cryptography operations and query segment matching, and the latter represents the overhead in the end-to-end query brokering time due to distributed routing.

**Computational Cost.** Assume the cost of asymmetric encryption and decryption as  $C_{ae}$  and  $C_{ad}$ , respectively, and the cost of commutative encryption and decryption as  $C_{ce}$  and  $C_{cd}$ , respectively. Since Pohlig-Hellman method requires almost the same number of exponentiation operations and modulus operations as RSA method, we adopt  $C_e$  and  $C_d$  to denote the cost of encryption and decryption in general, where  $C_e$  and  $C_d$  are at millisecond level.

For each XML query with  $m$  XPath steps, the computational costs of our scheme include asymmetric encryption cost  $m \cdot C_e$  at the user-side, asymmetric decryption cost of  $C_d$  at each intermediate broker, and additional commutative encryption and decryption cost  $C_e + C_d$  at each broker when encountering query segment with wildcard. At each broker, the query segment is matched against the routing schema carried by the broker. We adopt a similar approach as in [13], which implemented query segment matching and routing table look-up using hash table. It results in an average query segment matching time as  $1\text{ms}^4$ .

**Communication Cost.** Since the computational cost at each intermediate broker is at millisecond level, the overhead in end-to-end query brokering is mainly caused by the direct IP latency in overlay routing. The latency is measured using round trip time (RTT), where data from Internet traffic report <sup>5</sup> shows average IP latency is about 200ms.

An important parameter that determines the RTT of a particular query is the *number of hops* experienced by the query. It is related to how we split the global schema tree. If the finest granularity splitting is taken, the number of hops is exactly the depth of the query, providing that the query has no wildcard. When there exists a wildcard, the number of hops cannot be measured accurately, but we know it should be smaller than the depth of the path, a branch of the global routing schema, along which it is routed. As a result, we can estimate the *average*

<sup>4</sup> The result is based on simulations with synthetical XML queries and XPath routing rules that are generated atop of the XML benchmark [4].

<sup>5</sup> <http://www.internettrafficreport.com>

*number of hops* with the average depth of the global routing schema. Considering the setting as in [13], this value is 5.7.

Therefore, the average overall overhead is around  $5.7 \times 100ms$ , which is reasonable considering the preserved privacy.

## 6 Conclusion

We have described an application of content-based information brokering to defend against attribute-correlation attacks. A commutative encryption based scheme is further designed to protect query content from irrelevant brokers. Performance overhead is evaluated and results show that the performance degradation is insignificant.

The privacy of query content is enhanced with the proposed scheme, however, it is not without limitation - the privacy of query content is at risk in some extreme cases when a specific set of brokers collude. We plan to further explore this vulnerability in the future work and to devise possible mitigation techniques.

## References

1. Abdalla, M., Kiltz, E., Neven, G.: Generalized key delegation for hierarchical identity-based encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 139–154. Springer, Heidelberg (2007)
2. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: SIGMOD 2003: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pp. 86–97. ACM, New York (2003)
3. Altinel, M., Franklin, M.J.: Efficient filtering of XML documents for selective dissemination of information. *The VLDB Journal*, 53–64 (2000)
4. Berglund, A., Boag, S., Chamberlin, D., Fernandez, M.F., Kay, M., Robie, J., Simon, J.: XML path language (XPath) version 2.0 (2003), <http://www.w3.org/TR/xpath20/>
5. Calvanese, D., Giacomo, G.D., Lenzerini, M., Nardi, D., Rosati, R.: Source integration in data warehousing. In: DEXA Workshop, pp. 192–197 (1998)
6. Chan, C.-Y., Felber, P., Garofalakis, M., Rastogi, R.: Efficient filtering of XML documents with XPath expressions. In: ICDE, San Jose, pp. 235–244 (2002)
7. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.: Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations* 4(2) (2003)
8. De Capitani, S., Samarati, P.: Authorization specification and enforcement in federated database systems. *Journal of Computer Security* 5(2), 155–188 (1997)
9. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
10. Hammer, J., Garcia-Molina, H., Widom, J., Labio, W., Zhuge, Y.: The stanford data warehousing project. *IEEE Data Engineering Bulletin* 18(2), 41–48 (1995)
11. Koloniari, G., Pitoura, E.: Content-based routing of path queries in peer-to-peer systems. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 29–47. Springer, Heidelberg (2004)

12. Koudas, N., Rabinovich, M., Srivastava, D., Yu, T.: Routing XML queries. In: Proceedings of 20th International Conference on Data Engineering, p. 844 (2004)
13. Li, F., Luo, B., Liu, P., Lee, D., Chu, C.-H.: Automaton segmentation: A new approach to preserve privacy in XML information brokering. In: ACM CCS 2007, pp. 508–518 (2007)
14. Lu, H.Y.S.: Commutative cipher based en-route filtering in wireless sensor networks. In: Vehicular Technology Conference, vol. 2, pp. 1223–1227 (September 2004)
15. Snoeren, A.C., Conley, K., Gifford, D.K.: Mesh-based content routing using XML. In: Symposium on Operating Systems Principles, pp. 160–173 (2001)
16. Wiederhold, G.: Mediators in the architecture of future information systems. *Computer* 25(3), 38–49 (1992)
17. Wiederhold, G.: Value-added mediation in large-scale information systems. In: DS-6: Proceedings of the Sixth IFIP TC-2 Working Conference on Data Semantics, London, UK, pp. 34–56 (1995)
18. Yan, T.W., Garcia-Molina, H.: The SIFT information dissemination system. *ACM TODS* 24(4), 529–565 (1999)