XℓPPX: A Lightweight Framework for Privacy Preserving P2P XML Databases in Very Large Publish-Subscribe Systems

Angela Bonifati[1] and Alfredo Cuzzocrea[2]

[1] ICAR Inst., Italian National Research Council, Italy
bonifati@icar.cnr.it
[2] DEIS Dept., University of Calabria, Italy
cuzzocrea@si.deis.unical.it

Abstract. The problem of supporting privacy preservation of XML databases within very large publish-subscribe systems is rapidly gaining interest for both academic and industrial research. It becomes even more challenging when XML data are managed and delivered according to the P2P paradigm, since malicious accesses and unpredictable attacks could take advantage from the totally-decentralized and untrusted nature of P2P networks. In this paper, we propose **XℓPPX**, a distributed framework for very large publish-subscribe systems which supports (*i*) privacy-preserving fragmentation of XML documents stored in P2P XML databases, and (*ii*) the creation of trusted groups of peers by means of "self-certifying" XPath links. Furthermore, we present algorithms for querying privacy-preserving XML fragments in both schema-aware and schema-less mode, which are common scenarios when P2P XML databases operate in very large publish-subscribe systems. Finally, we complete our analytical contributions with an experimental study showing the effectiveness of our proposed framework.

1 Introduction

Supporting the privacy preservation of XML databases is a novel and interesting research challenge. With similar attractiveness, very large publish-subscribe systems are rapidly gaining momentum as long as innovative knowledge processing and delivery paradigms like Web and Grid Services take place. In such scenarios, due to its well-understood features, XML is widely used as basic language for both representing and processing semi-structured data located in remote databases within distributed and heterogeneous settings. For the sake of simplicity, here and in the following, we model an XML database as a (large) collection of XML documents on top of which traditional DBMS-like indexing and query functionalities are implemented, thus adopting the so-called *native XML databases*.

It is widely-recognized that providing solutions to guarantee privacy preservation over sensitive XML data plays a critical role in next-generation distributed and *pervasive* applications, and, particularly, in the context of very large publish-subscribe systems. This issue is even more challenging when XML data are managed and delivered according to the popular P2P paradigm, since malicious accesses and unpredictable attacks could take advantage from the totally-decentralized and untrusted nature

G. Psaila and R. Wagner (Eds.): EC-Web 2007, LNCS 4655, pp. 21 – 34, 2007.

of P2P networks. This scenario gets much worse when these networks admit *mobile peers*, since mobile devices have limited power and resource capabilities thus they cannot process huge amounts of data, neither implement complex *security countermeasures*. On the contrary, in publish-subscribe systems, the information is often carried by *data providers* that tend to enclose it into very large XML databases.

Starting from these considerations, in this paper we address the issue of efficiently querying P2P XML databases in very large publish-subscribe systems while guaranteeing privacy preservation over sensitive XML data. The solutions we propose to this challenge are codified inside the core layer of a lightweight distributed framework, called **X*ℓ*PPX**, which effectively and efficiently supports the representation and management of privacy-preserving P2P XML databases in very large publish-subscribe systems. In more detail, **X*ℓ*PPX** supports (*i*) the secure fragmentation of XML documents in very large publish-subscribe systems by means of lightweight XPath-based identifiers, and (*ii*) trusted groups of peers by means of secure XPath links that exploit the benefits deriving from well-known *fingerprinting techniques* [22]. Due to the latter feature, we name as "self-certifying" our innovative XPath links. Also, **X*ℓ*PPX** embeds algorithms for querying secure XML fragments in both schema-aware and schema-less mode, which is very useful in P2P networks.

In the rest of the paper, for the sake of simplicity we refer to privacy-preserving XML documents in terms of "secure documents". However, privacy preservation is quite different from both *security*, which deals with cryptography-driven algorithmic solutions for information hiding, and *access control*, which concerns with the problem of limiting information access to particular classes of end-users according to a pre-fixed access scheme (e.g., like in multimedia databases). By contrast, privacy preservation is mainly related to the problem of guaranteeing the privacy of sensitive data during data management tasks (e.g., query evaluation). Note that this problem delineates a double-edged sword, since data providers would make available knowledge and intelligent tools for efficiently processing knowledge, but, at the same time, they would hide sensitive knowledge (e.g., this could be the case of *Business Intelligence* systems). Conversely, data consumers would acquire useful-for-their-goals knowledge, whereas malicious data consumers would also access sensitive knowledge (e.g., this could be the case of attackers stealing enterprise's secrets).

2 Application Scenario and Motivating Example

We assume a very large publish-subscribe system in which peers can freely roam and extract knowledge from distributed XML databases located at particular nodes in the system (see Figure 1). Precisely, the system consists of a set of data providers holding XML documents, and a set of *data replicators* maintaining *views* on top of those documents. XML data stored by providers are yielded according to popular publish-subscribe dynamics (e.g., corporate B2C systems). Moreover, data providers notify data replicators when new documents of interest become available. On the other hand, data replicators build their views on the basis of locality (e.g., frequent queries and downloads), and periodically refresh such views. Mobile peers of the network are under the scope of a given data replicator, thus letting define several *domains of*

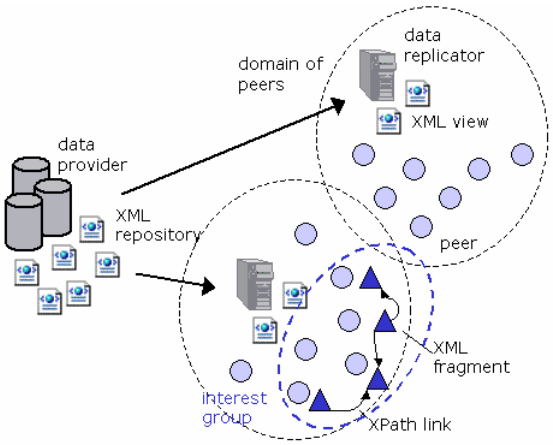


Fig. 1. Application scenario

peers. Domains of peers are freely built on the basis of (*i*) the interest of communities of peers in accessing views storing particular information (e.g., stock quotations), and (*ii*) locality issues (e.g., neighborhood criteria). Furthermore, without any loss of generality peers can also freely roam from one domain to another. It should be noted that the described setting covers a large set of modern systems and applications based on the novel Web and Grid Services paradigm, and, in particular, those adhering to the pioneering pervasive computing initiative.

Peers of such system are usually not interested to the entire view published by the local replicator, but to point some data contained into it, i.e. extracting *fragments* of the view. To avoid excessive computational overheads, peers prefer to access data replicators rather than data providers, while still wishing to access data providers whenever needed (e.g., due to load balancing issues). While peers fetch their fragments of interest, they also keep links to the related fragments stored on their neighbors. A link is represented as an absolute XPath expression within the original document, being such expression capable of uniquely determining the related fragment via prefix-matching. Peers linked to each other form an *interest group* (also called *acquaintance* in [3]), which can be viewed as a group of peers sharing *semantically-related fragments*, i.e. fragments related to concepts having a certain relationship with respect to the semantics of a specific application domain. Without any loss of generality, the same peer may belong to multiple interest groups.

In such a scenario, a pertinent problem is how to guarantee privacy preservation while accessing sensitive fragments, i.e. to avoid that untrusted peers can access secure fragments. It should be noted that while privacy preservation issues concerning a given data provider/replicator can be handled in a centralized manner by adopting specialized solutions, devising privacy-conscious P2P XML data processing solutions in environments like the one described above is still an open and unsolved problem, due the same nature of P2P networks. As a consequence, in this paper we focus on the latter research challenge.

In our reference application scenario, a peer has basically two options when it is looking for further information. In case it needs update its information, it has to access again the replicator, although consuming bandwidth and resources. In turn, when it needs *static data*, i.e. data with low variability over time, it is also offered to access its neighbor fragments via the above-mentioned links. Consider an example of this scenario in which a replicator publishes a view on public services available in a given urban area. Among the others, this view contains information about the local market stocks and the public transportation timetable. Notice that the former data are subject to a great variability over time, whereas the latter remain unchanged for a long time (e.g., a season). Imagine that a (mobile) user Bob is seeking for information about both kinds of data. For stock trends, Bob will directly access the local replicator, whereas for the departure times of trains, he can still access (trusted) neighbor's data (e.g., Alan, who had previously downloaded the same fragment of interest). In our framework, such a strategy is feasible thanks to our fragmentation model, which lets build links between related fragments.

A trusted peer can access the others' secure fragments by both directly browsing the links, and, alternatively, formulating an arbitrary XQuery query on (*schema-aware*) documents (note that fragments of documents are, in turn, documents). A further capability of our framework lets exploit the set of path expressions of an interest group to aid query formulation in the absence of a schema (i.e., against *schema-less* documents). It should be noted that schema-less query processing is a common situation in highly dynamic P2P networks, since, as shown in [3], a *global mediated schema* is not a reasonable assumption for such networks. For instance, Alan can retrieve the sets of path expressions of his interest group and use them in two ways: *(i)* grasp the fragments owned by other peers that may contain data he is interested to query later on, or *(ii)* use these path expressions to formulate his query, whenever the schema of the global document is not available.

As a side remark, note that handling updates in our framework (e.g., linked fragments that change in their structure or content), is outside the scope of this paper, and it is also an interesting and exciting research challenge we plan to address in the near future.

3 The XML Fragmentation Model in a Nutshell

For what concerns the P2P data layer, our framework relies on an *XML fragmentation model for P2P networks*, which is based on our preliminary work [4]. There, we assume to deal with P2P XML databases storing fragments of XML documents. In the present paper, we study the specific issues of hiding fragments to untrusted peers, thus protecting possibly sensitive data. We adopt a distributed, privacy-preserving mechanism according to which each (self-certifying) XPath link that points to a fragment is (*i*) encrypted by using a trusted key founding on the fingerprinting technique, and (*ii*) shared by all the other members of the same (interest) group. This solution also makes perfect sense when compared with *symbolic links* of *Self-certifying File Systems* (SFS) [12], since such links are encrypted by using a *Public Key Server* (PKS). In fact, in a P2P environment like the one we have just described in our application scenario (see Figure 1), we cannot rely on a central authority for public keys,

due to fault tolerance and scalability reasons. Nevertheless, it is reasonable to make groups of peers sharing the *responsibility* of a key (details are given in Section 4).

Although popular P2P networks allow the users share entire data files, the P2P paradigm is flexible enough to be applied to data at any granularity. In particular, in [4] we study the issue of sharing XML fragments scattered to multiple peers. In this Section, we revise the concepts presented in that paper, which serves as background to the novel privacy issues presented here. The key concept behind our model is the fact that, in a P2P setting, an XML document *does not entirely* reside on one peer for either space or relevance reasons. Therefore, we focus on how to represent the fragments of a document split on several peers, in order to be able to re-unify those at wish. Hence, a fragment is a sub-document of the original document maintaining XPath links to the latter, thus retaining the convenient side-effect of building a *decentralized catalog* over the P2P network. In our model, we do not expect the peers to agree on one schema and keep it updated with respect to every neighbor's changes to local data. On the contrary, every peer has one or more fragments and it may execute global queries without necessarily knowing the global schema. Thus, as we said above, we can handle both schema-less documents, which are very common in P2P networks as well as the Web, and schema-aware documents, more common in distributed database systems. Our query mechanism only relies on links for the first kind of documents, while it also looks at the local version of the schema for the second kind. Notice that this local version may disagree with other versions present in the network, since, as we said previously, we are not assuming a common mediated schema among peers.

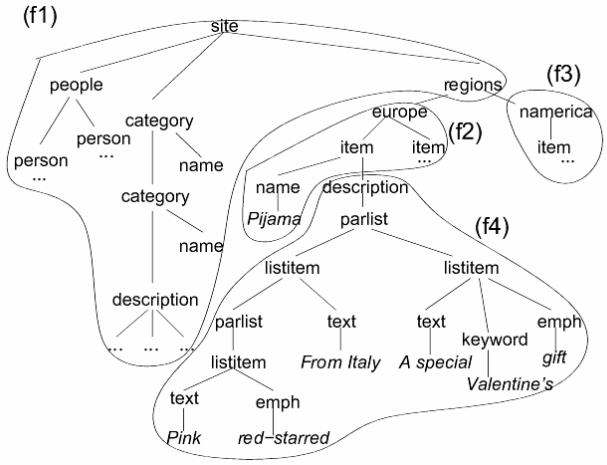


Fig. 2. A snippet of an XMark document

To enable fragmentation of XML documents, our model exploits a set of *lightweight path expressions*. A fragment *f* is a valid sub-tree of the original document having the following set of paths: (*i*) the *fragment identifier* f_I, i.e. the unique path expression identifying the root of the current fragment within the global document; (*ii*) the *super-fragment*

path expression p_s, i.e. the unique path expression linking the parent of the current fragment; (*iii*) one or more *child fragment path expressions* p_c, i.e. the path expressions linking the children of the current fragment. While f_l and p_s are stored separately from the fragment content, p_c paths are stored within special tags `sub` added as fragment leaves. Schema-aware documents also store on each peer the XML schema of the documents along with the local fragments and the above-described paths. Figure 2 shows a snippet of an XMark document [23], shredded according to our fragmentation strategy. Fragment f_2 = `/site/regions/europe` has `/site` as super-fragment and `/site/regions/europe/item[1]/description` as child fragment. Similarly, fragment f_1 = `/site` has no super-fragment and has both `/site/regions/ europe` and `/site/regions/namerica` as child fragments. Henceforth, paths sharing the same prefix, i.e. belonging to the same document, are named *related paths* in our fragmentation model.

Path expressions described above play a crucial role in identifying the fragments because they represent flat keys over the network. Moreover, they are more lightweight and scalable than B^+-trees used in [9] to answer range queries over relational data. In the following, we will describe how the fragments are allocated in a P2P network implementing **X***ℓ***PPX**, thus adhering to our XML fragmentation model. Here, we want to highlight that our strategy is reminiscent of vertical partitioning in distributed database systems, but customized for XML data. This distribution strategy uses a limited fragment of XPath, namely *light XPath*, consisting only of `/` and `[i]`, such that `[i]` indicates positional axes, which are useful to identify fragments. Note that the original document order is preserved in both assigning fragment identifiers and in encoding child fragments.

For the network configuration, in our setting we assume the structured P2P paradigm [13,15], which uses a *Distributed Hash Table* (DHT) to bias the search towards particular peers. Structured networks are quite different from super-peer networks used in [21]. In our model, the DHT leverages the uniqueness of fragment identifiers to identify and (al)locate the needed data over the network. For instance, a fragment such as `/site/regions/namerica` has a unique encoding within the DHT employed by the network. This path identification mechanism would work with any fragmentation technique (e.g., the one proposed in [5]), or even if fragmentation had not happened, such as, for instance, in a publish-subscribe system like the one described in [14] where entering machines publish their data by means of (Web/XML-enabled) path expressions. Implementation-wise, we rely on Chord [25] to realize the DHT. In Chord, each node keeps a logarithmic routing table to some external nodes, thus allowing a considerable network efficiency to be achieved. This is opposite to other systems, in which each node is connected to all other nodes, and similar to other DHT-based networks, such as *Pastry* [8] and *P-Grid* [1]. As in *DHash* [10], we admit a limited number of replicas of the fragments. Besides being stored on the current peer, the same fragment is replicated on every node of the peer routing table. This is done in order to guarantee the reliability of the network in the presence of peer failures.

Each fragment comes with the p_c and p_s path expressions, and with its fragment identifier f_l stored within the local peer. The fragment identifier is exposed to the outside under the form of a unique key. In such a way, any other peer that looks for a

particular path expression will search it through the DHT. We have extended the Chord DHT to support this behavior. However, in the original Chord, the hash function used is SHA-1, which is replaced in our model with the fingerprinting technique. Fingerprinting path expressions in a P2P network is similar to fingerprinting URLs [6], but different from an application point of view. In [4], we provide an experimental evaluation demonstrating that hashing and fingerprinting guarantee the same load balancing. We prefer fingerprints to any arbitrary hash function because of their software efficiency, their well-understood probability of collision (discussed in [4]) and their nice algebraic properties.

Now, we discuss how fingerprinting works. Let $A = \langle a_1 a_2 \ldots a_m \rangle$ be a binary string. We associate to A a polynomial $A(t)$ of degree $m - 1$ with coefficients in the algebraic field Z_2, $A(t) = a_1 \cdot t^{m-1} + a_2 \cdot t^{m-2} + \ldots + a_m$. Let $P(t)$ be an irreducible polynomial of degree k, over Z_2. Given $P(t)$, the fingerprint of A is the following: $f(A) = A(t) \ mod \ P(t)$. The irreducible polynomial can be easily found following the method in [22]. Therefore, in order for a peer to compute the path expression fingerprint, it suffices to store the irreducible polynomial $P(t)$. The latter has a fixed degree equal to $NF + 2 \cdot DM + Q$, being (*i*) 2^{NF} the number of fragments in the network, (*ii*) 2^{DM} the length of the longest path expression in the network, and (*iii*) 2^Q a threshold due to the probability of collision between two arbitrary distinct tokens [6]. It should be noted that polynomials are quite small structures to be replicated on each participating peer if compared to *replicated global indexes* used in [5]. Moreover, our set of lightweight path expressions and the accompanying polynomial are not directly comparable to probabilistic approaches based on *bloom filters* as in [14], which can handle XML data of relatively small depth.

4 XℓPPX: Overview and Privacy Preservation Features

In order to support the privacy preservation of XML fragments over P2P networks in very large publish-subscribe systems (and, as a consequence, that of database storing such fragments), in **XℓPPX,** XPath links pointing to fragments are encrypted using a trusted key encoded by means of a fingerprinting technique, and shared by peers of the same (interest) group. As we said above, such links are thus identified as "self-certifying" links, i.e. able to guarantee privacy preservation features.

To this end, besides fingerprinting path expressions, we also *fingerprint the actual XML content of the fragments*. This is novel with respect to previous initiatives, and not discussed at all in [6], where fingerprinting is not used for authentication purposes. Indeed, since fingerprinting, like hashing, reduces any arbitrary string to a fixed-length token, we can safely apply fingerprinting to the serialized content of an XML fragment. Since all we need to decode a fingerprinted item is the irreducible polynomial $P(t)$ (see Section 3), it is straightforward to create interest groups that share the same polynomial. Every peer within such groups can verify the authenticity of fragments in the community, and contribute to any issued query, which would be blind to the others. Thus, we enable privacy-preserving query processing with such verified peers. Of course, there will be as many groups of peers as the number of

polynomials we wish to allow in the network, ranging from the scenario with one distinct polynomial per-peer or per-group-of-peers to the scenario with one unique polynomial for all peers. Notice that this approach also guarantees that peers that answer queries are trustworthy (we describe our query algorithms in Section 5).

Figure 3 shows an overview of **X*ℓ*PPX**, where four peers and two interest groups are depicted. In more detail, the peers P_0 and P_3 form an interest group via sharing the polynomial $Poly_X$, whereas the peers P_1 and P_2 form another interest group via sharing the polynomial $Poly_Y$. The Figure also shows the logical architecture of the components that need to be implemented on each peer within **X*ℓ*PPX**. Consider a peer P_i among those depicted in Figure 3. Components implemented on P_i are the following: (*i*) *P2P Middleware*: it is the basic P2P middleware implementing traditional P2P communication protocols among peers of the network; (*ii*) *Privacy-Preserving Module*: it is the component of our framework implementing our fingerprint-based privacy-preserving scheme; (*iii*) *Fragment Manager*: it deals with the management of fragments according to our P2P XML fragmentation model; (*iv*) *Data Access Module*: it provides data access functionalities on the underlying P2P XML fragment database; (*v*) *Data Query Module*: it provides data query functionalities on the underlying P2P XML fragment database; (*vi*) *External Data Access API*: a collection of APIs used to access and query the local data replicator when up-to-date information is required (see Section 2).

Our proposal resembles the use of cryptography [17] (e.g., algorithms RSA and AES), and *self-certifying path names* [12], which are used in SFS directories to encrypt the host symbolic names. However, these cryptographic functions yielding large pieces of data introduce excessive spatio-temporal overheads in an overlay P2P network. Therefore, as claimed in Section 1, even if security issues obviously influence our work, our effort is in the privacy preservation of P2P data management.

Along with the basic idea of fingerprinting paths and fragments to support privacy preservation features, we also provide two extensions useful to handle privacy preservation among peers, which overcome more traditional schemes, and can be used to develop more specific privacy-preserving protocols for large and highly dynamic P2P networks. First, in traditional schemes (e.g., SFS), public key is decided by a centralized server, which would be not applicable in a P2P network. Contrarily to this approach, in order to decide the public key, we actually use an *authority group* policy as in [19]. More precisely, we require that *a public key is jointly decided by all the group members*. Secondly, traditional schemes focus on secure handling host names, whereas in our solution we need a broader level of security, i.e. extended to any (XPath) links either embedded in a fragment or lying out of it.

Handling new peers joining the network and wishing to be admitted in an interest group is another interesting issue, which becomes relevant with respect to the problem of avoiding that malicious peers attack the network. According to the authority group policy implemented in **X*ℓ*PPX**, the admission procedure requires that a new peer entering the network can be admitted in a group *if and only if* all the members agree on its admission. In more detail, in our model, this only applies if the new peer holds fragments related to those stored by other members. If this is not the case, then the peer can only be assigned to a new public key and create its own group.

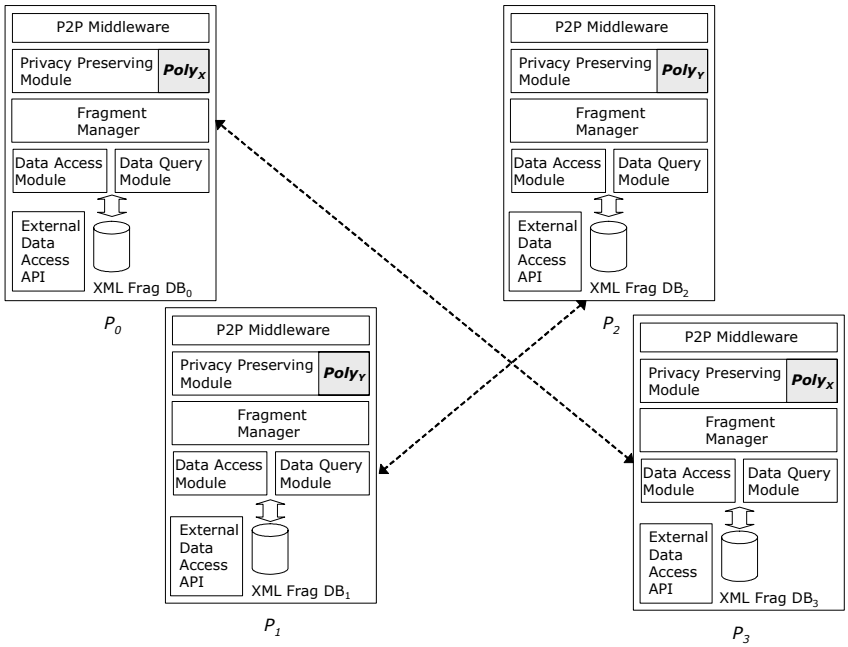


Fig. 3. XℓPPX Overview

There may be other kinds of attacks on a DHT-based P2P network, which go beyond the authenticity of data, such as threatens to the liveness of the system, and prevention of participants to find data (see [24] for a complete survey). These issues are orthogonal to the problem we investigate in this paper.

5 Schema-Less and Schema-Aware Evaluation of XPath Queries in XℓPPX

An arbitrary user can pose a query against the view of a replicator or, alternatively, against the fragments of the neighbor peers. We focus on the latter kinds of queries, as querying the replicator is quite straightforward and based on well-known solutions (e.g., query answering using views). To query linked fragments, our fragmentation model provides an effective strategy that inspects the information encoded into self-certifying path expressions. The latter represent a distributed catalog that can be fruitfully exploited during query evaluation. We base our description on examples that refer fragments shown in Figure 2. We discuss the evaluation of *descendant XPath queries*, since more complex XQuery queries use this module as a building block. While we use *light XPath* for fragment identifiers, the XML language we use for queries is complete XPath 1.0. Given an arbitrary XPath query, we divide it into several segments, either containing only / or containing // only. Note that, among all filters, only positional ones are kept in the path expression, as the others were not admitted as fragments identifiers. Thus, filters other than positional are initially

disregarded in the DHT lookup, and their evaluation postponed to the very-end, when the DHT-accessible results have been found.

When a schema of the split document is not available, the query is evaluated by looking at the local data of each peer, and by guessing the external data based on the related path expressions. Indeed, if we are evaluating `//description` and we originate the query on the peer storing the fragment f_2, then we already learn from f_2 child path expression that there is a tag `description` on the peer storing the fragment f_4. This gives us a priority in the order links are followed: this "promising" path expression is followed as first. When instead the tags contained in the child path expressions do not give this information, we follow all the links without giving priority to any.

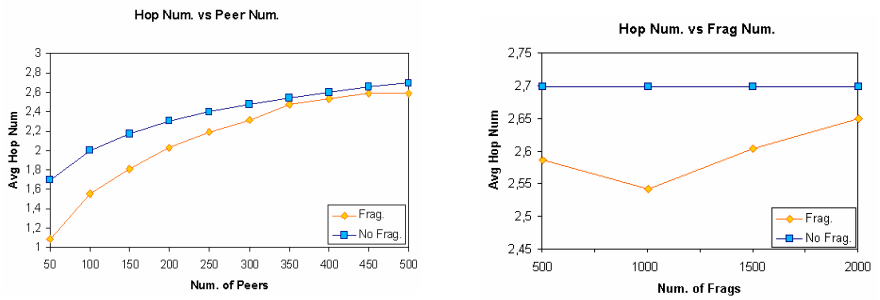
Consider now the query `/site/regions//item//description`, which is divided into `/site/regions` and `//item//description`. Hence, the two segments are evaluated according to two different algorithms. Single-slash path expressions can be directly fingerprinted and matched with the DHT content. In case no match is found, we know that no fragment is rooted in `/site/regions` (and this is indeed the case in Figure 2). Thus, we prune the path expression one step at a time and repeat the search in the DHT. Conversely, the path expression `/site`, which corresponds to the fragment f_1 is in the DHT. Starting from this fragment, our algorithm attempts to build the entire path expression `/site/regions`, with `regions` being a local element of f_1. Once the `regions` elements have been found, these become the context nodes of the double-slash path `//item//description`. Double-slash paths are evaluated one step at a time. To improve performances, the p_c paths of f_1 are followed in parallel to search item elements. The reader should note that here p_c paths do not let guess the presence of items on external fragments. The element `item` under `namerica`, as well as the rhs `item` under `europe`, is discarded as it has not neither local child nor external p_c paths to explore. Conversely, the lhs `item` under `europe` shows an external path which is promising in that it has `description` as final step. The latter path is then followed, and the element `description` is returned as result[1].

When a schema of the document is available, simple optimizations can be enabled. We emphasize here the use of the schema as a sort of *data summary*, similar to what provided by *data guides* and indexes for XML data [18]. In fact, the evaluation of both single-slash and double-slash queries can be improved by quickly determining and discarding ill-formed queries, i.e. queries that do not respect the available schema. For instance, a query like `/site//teacher` can be promptly declared as empty. The schema may also help in evaluation of queries containing wildcards, such as `/site//*/person`. In such a case, we avoid inspecting fragments f_2 and f_3 (and, as a consequence, f_4) if we can infer from the schema that `person` elements cannot lie underneath `regions` elements. The reader should notice that the two optimizations described above can be achieved by simply handling schemas of documents, thus without introducing excessive computational overheads. In any case, as highlighted above, schemas are not mandatory in our query strategy. Also, we expect that a peer modifies at will its local data by introducing new elements. We assume the modifications would not be communicated to other peers.

[1] We assume here the network only has these fragments, otherwise the search would exhaustively explore the other fragments till coverage of all the nodes.

6 Experimental Study

In order to test the effectiveness of our proposed framework, we conducted various experiments by using as input the popular XML benchmark data set XMark. As highlighted previously, we focus our attention on testing the capabilities of the P2P layer of **XℓPPX**, having the latter a more critical role than data-provider and replicator layers. In particular, we performed two sets of experiments. In the first one, we stressed the *query capabilities* of **XℓPPX**, i.e. the capability of **XℓPPX** in answering XPath queries over fragmented XML data, whereas in the second one we stressed the *privacy preservation capabilities* of **XℓPPX**, i.e. the capability of **XℓPPX** in supporting privacy preservation functionalities over fragmented XML data.

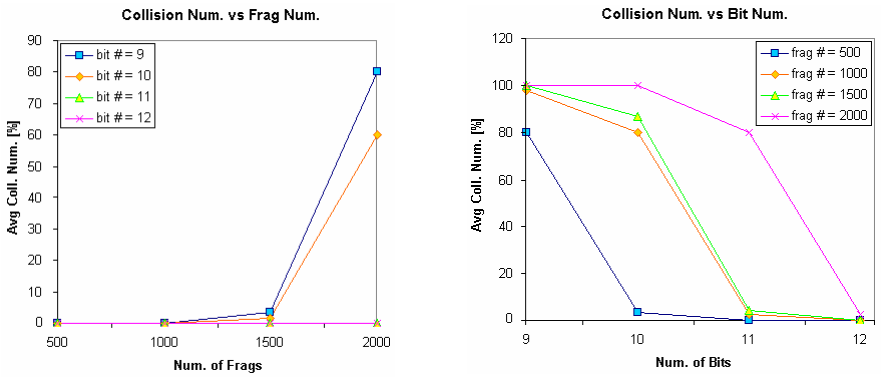


(*a*) (*b*)

Fig. 4. Query performance on the XMark benchmark Xm_{30}

In our experimental setting, we use polynomials with degree equal to 64, which leads to an acceptable probability of 2^{-10}, and allows us to exploit a maximum length for path expressions of 50 steps (averaged on a length of 10 symbols per step), and a maximum number of fragments equal to 2^{30}, which is huge enough for arbitrary networks. Figure 4 and Figure 5 show the experimental results of our analysis, which has been conducted over an XMark document that was 30 MB is size, denoted by Xm_{30}.

For what concerns the query capabilities of **XℓPPX**, Figure 4 (*a*) shows the average hop number with respect to the number of peers in the two different settings in which (*i*) Xm_{30} is split into 1000 fragments, and (*ii*) Xm_{30} is kept entire. Figure 4 (*b*) shows the same metrics with respect to the number of fragments (the number of peers is set to 500). For what concerns the privacy preservation capabilities of **XℓPPX**, Figure 5 (*a*) shows the average collision number in percentage among fingerprints with respect to the number of fragments, when varying the number of bits used to encode fingerprints. Finally, Figure 5 (*b*) shows the same metrics with respect to the number of bits used to represent fingerprints, when varying the number of fragments in the network. In both latter cases, the number of peers is set to 500, and the number of interest groups is set to 5. From the analysis of the experimental results, it clearly follows that the proposed framework allows us to efficiently query P2P XML databases while guaranteeing privacy preservation over sensitive XML documents.



(*a*) (*b*)

Fig. 5. Privacy preservation performance on the XMark benchmark Xm_{30}

7 Related Work

In this Section, we briefly outline some state-of-the-art solutions for supporting privacy preservation functionalities over P2P networks, which are close to our work. On the other hand, the literature on XML-based publish-subscribe systems is well established, and we omit it here, due to space reasons. A comprehensive tutorial can be found in [11].

Threshold distributed cryptography is used in [19] for the sake of membership control. [19] casts doubt on the viability of these approaches in unstructured P2P networks if they do not guarantee verifiability of group members compromised by adversaries. We acknowledge this issue, and focus on a simple yet effective solution for trusted P2P XML fragments. Security issues in an agent-based P2P network are examined in *BestPeer* [21]. However, security has a different meaning there, i.e. to protect the data carried by agents when they inspect fixed or unknown network paths. For this purpose, *signencryption* [17] is used. However, no experimental evidence is given to justify the scalability of cryptography in P2P networks. Our approach is different as it is customized for XML fragments over P2P networks. 128-bit encryption to protect peer communication is also used in agent-based *PeerDB* [20]. *PeerDB* differs from our framework in that it performs IR-based queries on non-distributed relational data. However, like ours, it realizes a P2P full-fledged database. Access-control on each peer is used in [2] to protect *active XML documents* containing service calls. Goals of [2] are different from ours, as [2] aims at providing a sort of "application-level" degree of security which could indeed used on top of our framework to exploit the available functionalities in the context of (secure) P2P Web Services. Security on semantic views of Web information are realized by means of *information mediators* in [16]. This architecture, although very interesting for a Web context, is not applicable to a P2P network. Handling uncooperative or selfish nodes in a large-scale P2P network is done in [7]. Both problems are very interesting but orthogonal to ours.

8 Conclusions and Future Work

In this paper, we have presented **XℓPPX**, a framework for supporting privacy preservation of XML fragments among peers embedded in very large publish-subscribe systems, where XML views are published by both data providers and replicators. Efficiently supporting query processing on secure P2P XML fragments is another goal of **XℓPPX**. In **XℓPPX**, privacy preservation features across peers are provided by means of well-established fingerprinting techniques, whose reliability and efficiency have been proved via a comprehensive experimental study over XMark documents. Schema-less and schema-aware query algorithms for the evaluation of descendant XPath queries over secure XML fragments have also been presented and discussed. The experimental evaluation of **XℓPPX** has further confirmed the advantages deriving from our fragmentation technique, and the effectiveness of the proposed privacy-preserving scheme. Future work is mainly focused on extending the query functionalities implemented in **XℓPPX**, in order to include more advanced IR capabilities. Moreover, we plan to test the scalability of privacy preservation features of **XℓPPX** in real-life systems.

References

[1] Aberer, K., Cudre-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R.: P-Grid: A Self-Organizing Structured P2P System. SIGMOD Record 32(3), 29–33 (2003)

[2] Abiteboul, S., Alexe, B., Benjelloun, O., Cautis, B., Fundulaki, I., Milo, T., Sahuguet, A., An Electronic Patient Record "on Steroids": Distributed, Peer-to-Peer, Secure and Privacy-conscious. In: Proc. of VLDB, pp. 1273–1276 (2004)

[3] Bernstein, P.A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I.: Data Management for Peer-to-Peer Computing: A Vision. In: Proc. of ACM WebDB, pp. 89–94 (2002)

[4] Bonifati, A., Cuzzocrea, A.: Storing and Retrieving XPath Fragments in Structured P2P Networks. Data & Knowledge Engineering 59(2), 247–269 (2006)

[5] Bremer, J.-M., Gertz, M.: On Distributing XML Repositories. In: Proc. of ACM WebDB, pp. 73–78 (2003)

[6] Broder, A.Z.: Some Applications of Rabin's Fingerprinting Method. Springer, Heidelberg (1993)

[7] Buchmann, E., Bohm, K.: FairNet - How to Counter Free Riding in Peer-to-Peer Data Structures. In: Proc. of CoopIS/DOA/ODBASE, vol. 1, pp. 337–354 (2004)

[8] Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Proximity Neighbor Selection in Tree-based Structured Peer-to-Peer Overlays. Technical Report MSR-TR-2003-52, Microsoft Research (2003)

[9] Crainiceanu, A., Linga, P., Gehrke, J., Shanmugasundaram, J.: Querying Peer-to-Peer Networks Using P-Trees. In: Proc. of ACM WebDB, pp. 25–30 (2004)

[10] Dabek, F., Brunskill, E., Frans Kaashoek, M., Karger, D., Morris, R., Stoica, I., Balakrishnan H.: Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service. In: Proc. of IEEE HotOS, pp. 81–86 (2001)

[11] Jacobsen, H.-A., Llirbat, F.: Publish-Subscribe Systems. Tutorial at IEEE ICDE (2001)

[12] Fu, K., Kaashoek, M.F., Mazieres, D.: Fast and Secure Distributed Read-Only File System. Computer Systems 20(1), 1–24 (2002)
[13] Galanis, L., Wang, Y., Jeffery, S.R., DeWitt, D.J.: Locating Data Sources in Large Distributed Systems. In: Proc. of VLDB, pp. 874–885 (2003)
[14] Gong, X., Yan, Y., Qian, W., Zhou, A.: Bloom Filter-based XML Packets Filtering for Millions of Path Queries. In: Proc. of IEEE ICDE, pp. 890–901 (2005)
[15] Huebsch, R., Hellerstein, J.M., Lanham, N., Loo, B.T., Shenker, S., Stoica, I., Querying the Internet with PIER. In: Proc. of VLDB, pp. 321–332 (2003)
[16] King, R.: Security Maintenance Mediation: a Technology for Preventing Unintended Security Breaches. Concurrency and Computation: Practice and Experience 16(1), 49–60 (2004)
[17] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
[18] Milo, T., Suciu, D.: Index Structures for Path Expressions. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 277–295. Springer, Heidelberg (1998)
[19] Narasimha, M., Tsudik, G., Yi, J.H.: On the Utility of Distributed Cryptography in P2P and MANETs: The Case of Membership Control. In: Proc. of IEEE ICNP, pp. 336–345 (2003)
[20] Ooi, B.C., Tan, K.L., Zhou, A.Y., Goh, C.H., Li, Y.G., Liau, C.Y., Ling, B., Ng, W.S., Shu, Y.F., Wang, X.Y., Zhang, M.: PeerDB: Peering into Personal Databases. In: Proc. of ACM SIGMOD, p. 659 (2003)
[21] Pang, X., Catania, B., Tan, K.: Securing Your Data in P2P Systems. In: Proc. of IEEE DASFAA, pp. 55–61 (2003)
[22] Rabin, M.O.: Fingerprinting by Random Polynomials. Technical Report CRCT TR-15-81, Harvard University (1981)
[23] Schmidt, A., Waas, F., Kersten, M., Carey, M., Manolescu, I., Busse, R.: XMark: A Benchmark for XML Data Management. In: Proc. of VLDB, pp. 974–985 (2002)
[24] Sit, E., Morris, R.: Security Considerations for Peer-to-Peer Distributed Hash Tables. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 261–269. Springer, Heidelberg (2002)
[25] Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. of ACM SIGCOMM, pp. 149–160 (2001)