An Electronic Patient Record "on Steroids": Distributed, Peer-to-Peer, Secure and Privacy-conscious*

Serge Abiteboul[1] Bogdan Alexe[2] Omar Benjelloun[1] Bogdan Cautis[1]
Irini Fundulaki[2] Tova Milo[1,3] Arnaud Sahuguet[2]

[1]:INRIA Futurs, [2]: Bell Labs, Lucent Technologies, [3]: Tel-Aviv University

1 Introduction

Getting sick or injured is never a good idea. You never know when it's going to happen or where. In such situations, it is crucial to be able to gather all the relevant information to make the diagnosis and treatment as effective as possible.

By nature, an electronic patient record (EPR) [9] consists of many pieces owned and managed by different entities: yourself as a patient, your referring doctor, the various specialists you are dealing with (e.g. gynecologist, optometrist, physical therapist), the pharmacist(s) you shop from, the various hospitals you go to for surgery or special examination, the insurance company (private or state-owned) that handles the billing and reimbursement, some wearable devices that monitor your heartbeat or glucose level, etc.

Besides the distribution of the data, one of the main challenges in the management of EPR information is its sensitive nature. Clearly, a patient does not want unauthorized parties to access confidential parts of her medical record. One should note that the issue of managing information that is both highly distributed and partly confidential does not arise only in the EPR context, but is also typical of many other distributed applications, such as the management of user profiles, shared agendas, collaborative workspaces, etc. *The goals of this demo are (1) to propose a unified, peer-to-peer, privacy conscious solution to the management of distributed sensitive information, and (2) illustrate it through an EPR management example scenario.*

So far, most approaches to the management of EPRs considered a setting where the information is highly centralized (e.g. in hospitals), for which centralized approaches – like the one advocated by Hippocratic databases [5] – make perfect sense. As EPR relevant information is more and more distributed, we argue that a centralized approach is not always satisfactory. To handle the inherent distribution of data, we advocate for a peer-to-peer architecture, where EPR data can be seen as a large virtual XML document [12] – one per user – that is being accessed and modified by the numerous players involved. As far as access control on XML data is concerned, existing approaches [6, 8] offer great flexibility in terms of the definition and the enforcement of access rules, but do not provide means to handle highly distributed data. In a peer-to-peer setting, each peer may want to enforce particular access control rules for the data that it owns and for the data that is accessible through it, and possibly delegate to other peers the task of defining/enforcing these rules. These functionalities are precisely the ones provided by the system demonstrated here.

Contribution: We demonstrate a novel solution for the privacy-conscious integration of distributed data, that relies on the combination of two key technologies: Active XML [4], that provides a highly flexible peer-to-peer paradigm for data integration, and GUP$^{\text{ster}}$ [14], that unifies the enforcement of access control and source descriptions. While each of these two technologies has been demonstrated separately before [4, 1, 2, 11], we show here that their synergy yields a powerful generic platform that seamlessly handles data integration and privacy enforcement tasks in a highly distributed setting.

Active XML (AXML for short) is a framework to manage XML documents where some of the data is given explicitly, whereas other parts are calls to Web services [16] that generate the "missing" data. Such documents can be viewed as a partially virtual. The AXML platform makes it possible to manage and query these documents, offering rich features such as lazy/distributed query evaluation and typing [13, 3].

GUP$^{\text{ster}}$ is a framework for the privacy-conscious management of distributed XML data. Source descrip-

* This work was partially supported by EU IST project DBGlobe (IST 2001-32645), and by the French government grant ACI MDP2P.

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

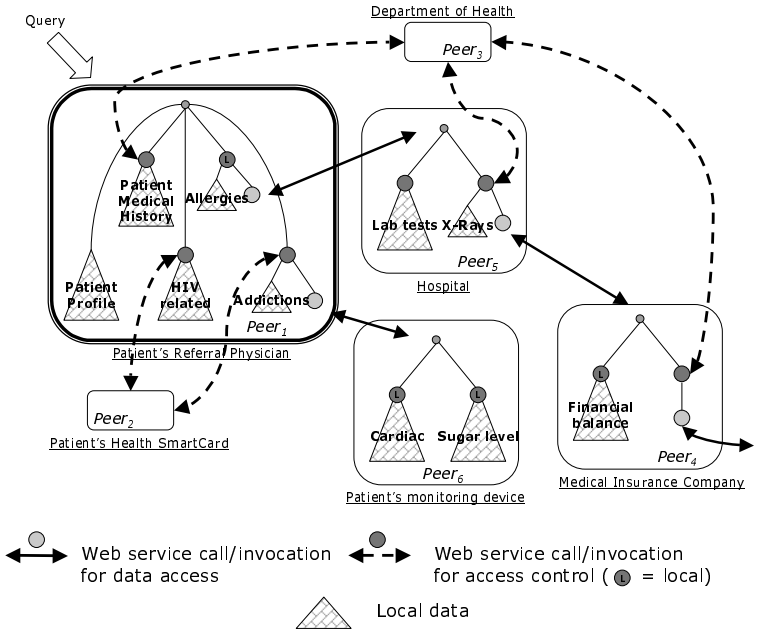

Figure 1: Global architecture

tions (i.e. the queries supported by each source) and user access rights are specified *in a uniform way* using XSquirrel, a specialized XML query language. The nice closure properties of this language for operators such as union and composition allow user queries to be statically *rewritten* into queries that are both *authorized* (from the access control rules) and *compatible* (with the source descriptions).

Together, the two technologies are combined in the architecture of Fig. 1, where peers can act as data sources, integrators, and provide/enforce access control/source descriptions. In this demo, this enables us to (i) represent a user's EPR as a virtual document, distributed among different peers on the network, (ii) enforce fine-grained access control policies over this document, also in a distributed way, and (iii) process queries while enforcing the access control, in a distributed, privacy conscious manner. In Fig. 1, an EPR at a physician's peer has some part on a monitoring device, and some part at the hospital and (recursively) at the insurance company. Access control on this distributed data is partly determined by the patient's preferences (through her SmartCard), partly by the general regulations of the Department of Health, and partly locally at each participating peer.

The rest of this paper is organized as follows. Section 2 overviews AXML and GUP^{ster}. Section 3 explains how their combination supports privacy-conscious data integration. Section 4 outlines a demonstration scenario for the management of electronic patient records.

2 Underlying technologies

In this Section, we briefly introduce Active XML documents and the GUP^{ster} query rewriting mechanism.

```
<patientData>
  <identity>
    <name>
      <first>Mary</first>
      <last>Smith</last>
    </last>
    <dateOfBirth>19710402</dateOfBirth>
    <ssn>
      <call svc="filterSSN@ssa.org">
        <call svc="getMySSN@local" />
      </call>
    </ssn>
  </identity>
  <address>2200 Broadway, NY, 10023 NY</address>
  <medicalRecord>
    <call svc="filterMR@DeptOfHealth">
      <visit>
        <date>20030102</date>
        <MD>
          <name>John Jones</name>
          <specialty>eyeMD</specialty>
        </MD>
        <prescription>Eye drops</prescription>
        <diagnosis>
          <call svc="diagDescription@DoctorPepper">
            <call svc="getDiagnosis@DoctorPepper">
              <visitID>627692876693</visitID>
            </call>
          </call>
        </diagnosis>
      </visit>
      ...
      <call svc="getVisits@MDVisits">
        <patient>Mary F. Smith</patient>
      </call>
    </call>
  </medicalRecord>
  ...
</patientData>
```

Figure 2: An EPR as an AXML document

2.1 Active XML: P2P data integration

AXML is a declarative framework that harnesses Web services for data integration, and is put to work in a peer-to-peer architecture. An AXML document is an XML document where some of the data is given extensionally, as regular XML elements, while some data is given *intensionally*, by means of calls to Web services [16]. These service calls are represented by special `call` elements, that are embedded in the document. The data inside the `call` element is the *parameter* of the service call. When a service is invoked, the call parameter is passed to it, and its result replaces the service call in the document.

For example, consider the document in Fig. 2, that represents an electronic patient record (ignore for now the boxed parts). The document contains the patient's identity information and medical record. Some of the data is given explicitly, e.g. the patient name, and some can be obtained by calling Web services, e.g. her social security number (`ssn`)that is available from `getMySSN@local`, and additional visits information that can be obtained via the `getVisits` service hosted by the `MDVisits` peer.

Lazy query evaluation When a query is evaluated on a document, the service calls whose answer is relevant for the query are invoked [1]. To optimize the computation, when possible, (sub)queries are "pushed" to the service providers, thus reducing the materialization and transfer of data (see [3] for details). When a query is pushed to a service, it is passed to it as an extra parameter. The service, depending on its particular implementation, can either naively perform its normal computation, and then evaluate the pushed query on the result, or have a more optimized strategy.

Note that, since the called services may themselves be defined as queries over AXML documents, the called peers may run a similar computation, thus achieving a P2P ad-hoc style of data mediation.

2.2 GUP^{ster}: Declarative access control

To ensure the privacy of data, one needs a fine grained control over the requests initiated by users. We adopt here the GUP^{ster} approach [11], that unifies access control (AC) and source descriptions, by relying on a single query language to specify both, and a single query rewriting mechanism to enforce them.

The query language used by GUP^{ster} to describe AC rules and source descriptions is XSquirrel, a simple XML *projection* query language. The language uses a syntax similar to that of XPath [7], but has different semantics: rather than returning sets of nodes, like XPath, it returns a projection of the queried document. Intuitively, the result of an XSquirrel query on a document is the sub-document that contains (a) all descendant leaf nodes of the requested nodes and (b) their ancestors up to the root of the initial document.

As a simple example, suppose that we want to restrict the access to the `visit` elements in Mary's EPR, allowing Dr. Hull to access only the `prescription`, `diagnosis` and `MD`'s `specialty` elements. The corresponding XSquirrel access rule would be:

`AC1`: `Hull`, `/visit/(MD/specialty` $\cup$ `prescription` $\cup$ `diagnosis)`

Similar rules can be used to describe the access rights of other users, or the queries accepted by data sources (in our case, the embedded Web service calls).

To process a query, GUP^{ster} fetches the relevant AC/source description rules and composes them with the query, yielding a *restricted* query that accesses/returns only allowed information. For instance, suppose that Dr. Hull attempts to retrieve all the data regarding her visits to ophthalmologist by issuing the query: `/visit[MD/specialty="eyeMD"]`

The restricted query would be:

```
/visit[MD/specialty="eyeMD"] /(prescription
                               ∪ diagnosis ∪ MD/specialty)
```

While the result of the composition in this case is very intuitive, the general rewriting technique for more complex queries and rules is more intricate, and relies on XSquirrel's closure properties. See [10] for details.

[1]Service calls (and their parameters) represent *virtual* data and are not queriable.

It should be noted that, in the original GUP^{ster} system, AC rules as well as source descriptions are specified with respect to a global schema, and are managed by a centralized mediator, in charge of applying them for all the queries asked on the global schema. The main contribution of the present work is in leveraging GUP^{ster}'s access control mechanism for P2P, schema-free integration of decentralized data.

3 Privacy-conscious P2P integration

We first present filtering services, our basic construct for distributed access control, then explain how they are used to enforce access control, and finally consider some important security aspects.

3.1 Filtering services

GUP^{ster}'s access control functionality is naturally incorporated into AXML documents, by being provided as Web services. We define *filtering services*, that enforce GUP^{ster}-like rules on AXML data. Each of these services can protect some AXML data by filtering the queries that can be evaluated on them, according to a set of access control rules. Note that the protected data does not have to be sent extensionally as a parameter, but can be represented intensionally by a service call, and thus hidden from the filtering service.

In our EPR document, calls to such services are the boxed ones. For instance, `filterMR@DeptOfHealth` is a filtering service that restricts Dr. Hull's access to Mary's visit information, by using the access control rule `AC1` given in Section 2.2.

It should be noted that, like for other Web services, filtering services can be freely used and combined inside AXML documents. Therefore, in the same way that service calls are used to integrate data from various sources, they can also be used to combine access control/source descriptions that are enforced/provided by various parties. For instance, queries targeted at the diagnosis information, that is retrieved from `getDiagnosis@DoctorPepper` will be both protected by the AC enforced by the call to `filterMR@DeptOfHealth` and limited by the source description provided by `diagDescription@DoctorPepper`.

We next explain how access control is enforced while queries are evaluated.

3.2 Query evaluation and access control

To enable the enforcement of access control, we extend Active XML's lazy evaluation and query pushing mechanisms to apply also to filtering services. When a query is evaluated on an AXML document, and a call to a filtering service is met, the corresponding subquery is pushed to the filtering service, which combines it with the access control rules defined for the user asking the query, using GUP^{ster}'s rewriting algorithms.

Then, the filtering service can either (lazily) evaluate the resulting rewritten query on the guarded data

and return the result of this evaluation, or let the requester evaluate this query. This choice is controlled by the input/output types specified for the filtering services, using techniques introduced in [13].

Consider our previous GUP[ster] example. Suppose the peer of Mary's referring MD (RMD) receives a request from Dr. Hull to evaluate, on the document of Fig. 2, the following query:

```
/patientData/medicalRecord/visit[MD/specialty="eyeMD"]
```

The RMD peer pushes a query about visits to `filterMR@DeptOfHealth`, which composes it with the access control rule `AC1`, as discussed in Section 2.2. Then, the composed query is evaluated (either at the RMD peer or at the filtering peer `DeptOfHealth`). This will also entail pushing a subquery to `getVisits@MDVisits`, which, recursively, may also invoke calls to filtering services guarding its own data (hence further restricting the pushed query).

3.3 Security aspects

Classical transport-level security mechanisms, based on public-key encryption and digital signatures will be used to enforce properties such as confidentiality, authentication and non-repudiation, at the message level. When a finer level of control is required, XML Encryption and XML Signature standards [15, 17] will be used to encrypt/sign only parts of the exchanged messages.

Basic Web services for handling encryption and signature will be present locally on every peer, and calls to these local services will be used in AXML documents, e.g. to provide means to decrypt some encrypted data, or to verify a signature. The typing mechanisms mentioned above will also be used to enforce constraints such as: queries pushed to `getDiagnosis@DoctorPepper` must be signed by `filterMR@DeptOfHealth`.

4 Demonstration scenario

We will demonstrate how an EPR document can be managed by a number of AXML peers representing hospitals, MD's, insurance companies, Department of Health. These peers will be living on remote servers, laptops and mobile devices. Each of them will provide/integrate information/filtering services, or a combination of these. E.g., a hospital provides information about visits, while an insurance company gives reimbursement reports, and access control on both of them is enforced by both the regulations of the Dept. of Health, and their own respective privacy policies.

We will illustrate how this distributed data can be queried by different users (referring doctor, nurse, insurance company), and how the specified access control rules are enforced along the way. We will in particular show how:

- a patient manages her EPR, by adding or removing sources, and protecting this integrated data, by using external filtering services, or defining her own, using access control rules.
- a doctor can access/query this EPR with some restrictions,
- a nurse can access the EPR in an even more restricted way,
- the insurance company can check the latest prescriptions for future reimbursement.

For all these scenarios, we will demonstrate how the queries are executed efficiently:

- only the relevant/permissible parts of the AXML document are exchanged between peers,
- an AXML peer can fully or partially evaluate a query (similar to LDAP referral mode), by delegating some of the computation to filtering peers or information sources.

References

[1] S. Abiteboul, B. Amann, J. Baumgarten, O. Benjelloun, F. Dang Ngoc, and T. Milo. Schema-driven Customization of Web Services (demo). In *Proc. of VLDB*, Berlin, 2003.

[2] S. Abiteboul, J. Baumgarten, A. Bonifati, G. Cobena, C. Cremarenco, F. .Dragan, I. Manolescu, T. Milo, and N. Preda. Managing Distributed Workspaces with Active XML (demo). In *Proc. of VLDB*, Berlin, 2003.

[3] S. Abiteboul, O. Benjelloun, B. Cautis, I. Manolescu, T. Milo, and N. Preda. Lazy Evaluation for Active XML. In *Proc. of ACM-SIGMOD*, Paris, June 2004.

[4] S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber. Active XML: Peer-to-Peer Data and Web Services Integration (demo). In *Proc. of VLDB*, Hong Kong, 2002.

[5] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proc. of VLDB*, Hong Kong, 2002.

[6] E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents. *ACM Trans. on Information and System Security*, 5(3):290–331, 2002.

[7] J. Clark and S. DeRose (eds.). XML Path Language (XPath) Version 1.0. W3C Recommendation, November 1999. http://www.w3c.org/TR/xpath.

[8] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. A Fine-Grained Access Control System for XML Documents. *ACM Trans. on Information and System Security*, 5(2):169–202, May 2002.

[9] Health Level Seven. http://www.hl7.org/.

[10] I. Fundulaki and A. Sahuguet. A language-based approach for distributed user profile data management. Technical Report, November 2003.

[11] I. Fundulaki, A. Sahuguet, D. Lieuwen, N. Onose, G. Giraud, and N. Pombourcq. Share your data, keep your secrets (demo). In *Proc. SIGMOD*, Paris, 2004.

[12] E. Kuikka, A. Eerola, and J. Komulainen. Structuring the electronic patient record. Technical report, University of Kuopio, 2001.

[13] T. Milo, S. Abiteboul, B. Amann, O. Benjelloun, and F. Dang Ngoc. Exchanging Intensional XML Data. In *Proc. of ACM SIGMOD*, San Diego, 2003.

[14] A. Sahuguet, R. Hull, D. Lieuwen, and M. Xiong. Enter Once, Share Everywhere: User Profile Management in Converged Networks. In *Proc. of CIDR*, 2003.

[15] W3C. XML Encryption WG, 2001. http://www.w3.org/Encryption.

[16] W3C. Web Services Activity, 2002. http://www.w3.org/2002/ws.

[17] W3C. XML Signature WG, 2002. http://www.w3.org/Signature.