Social Networking on top of the WebdamExchange System [1]

Émilien Antoine †, Alban Galland †, Kristian Lyngbaek †, Amélie Marian ‡, Neokis Polyzotis ♯

†*INRIA Saclay & ENS Cachan,*
61 av du Président Wilson, 94325 Cachan, France
firstname.lastname@inria.fr

‡*Rutgers University,*
110 Frelinghuysen Road Piscataway, NJ 08854-8019, US
amelie@cs.rutgers.edu

♯*U. of California Santa Cruz*
1156 High St., Santa Cruz, CA 95064, US
alkis@cs.ucsc.edu

***Abstract*—The demonstration presents the WebdamExchange system, a *distributed knowledge base management system with access rights, localization and provenance*. This system is based on the exchange of logical statements that describe documents, collections, access rights, keys and localization information and updates of this data.**

We illustrate how the model can be used in a social-network context to help users keep control on their data on the web. In particular, we show how users within very different schemes of data-distribution (centralized, dht, unstructured P2P,etc) can still transparently collaborate while keeping a good control over their own data.

I. Context and model

General usage of the web to publish and share information, in particular in the context of social networks, stresses the need to control access to private data in a distributed environment. Indeed, personal data on the web is typically spread across different centralized systems (e.g. Facebook, Twitter, Doodle), on various personal machines (e.g. laptop, smartphones), on friends machines or on untrusted peers (e.g. Weave servers of Mozilla Labs). Consequently, the users suffer from the silo-storage of the data while editing, sharing and navigating their data. Moreover, personal data is often very sensitive and one would like to enforce a strong access control on it. The goal of this demonstration is to introduce a WebdamExchange model [3] implementation, namely the WebdamExchange system, and to explain how it is used in a social-networking scenario where users interact transparently on private data hosted by very different kinds of data-distribution systems. There have already been some recent works on distributed social networks, e.g., [4], and also on privacy in P2P networks, e.g., [5], which mostly focus on trust and which are somewhat complementary to ours. In contrast, we use social networking as a motivating example for a more general system, since the distribution and access control on the web is not restricted to this setting.

To overcome the problem that the users of today's web are facing, a general model to describe data-management on the web is needed. This model has to simplify the distribution of the data of the users, the access-control on the data, and its integration while allowing for larger flexibility for reasoning on this knowledge. Some systems have already been presented in this direction, e.g., [6,7], which are focused on special kinds of distributed systems and enforce access control using an external key management system. In contrast with these approaches, we believe a general model has to cover both data and key management, and a larger spectrum of distribution techniques.

The WebdamExchange model [3] is a model of a *distributed knowledge base with access rights, localization and provenance*. The main originality of the approach is that logical statements are used to describe data, access rights, secret keys and localization information. Hence, all data and meta-data are represented at the same level and can be used for formal reasoning. In particular, it can be represented in an extension of datalog and used to model distribution and security properties, as shown in [2]. The statements carry most of their access control, by using cryptographic signatures to enforce write access and encryption to enforce read access, naturally leading to robust distributed systems. This knowledge can be communicated, replicated, queried, updated and monitored, while preserving access control properties. We demonstrate a system implementing the WebdamExchange model, namely the WebdamExchange system, and illustrate how it is used with a social network scenario.

The rest of the document is organized as follows. The next section describes a motivating example, and the corresponding desired functionality and challenges. Section III discusses how its features allow our system to overcome these challenges. Section IV presents the actual setting of our demonstration.

[1] This work has been partially funded by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant Webdam, agreement 226513. http://webdam.inria.fr/

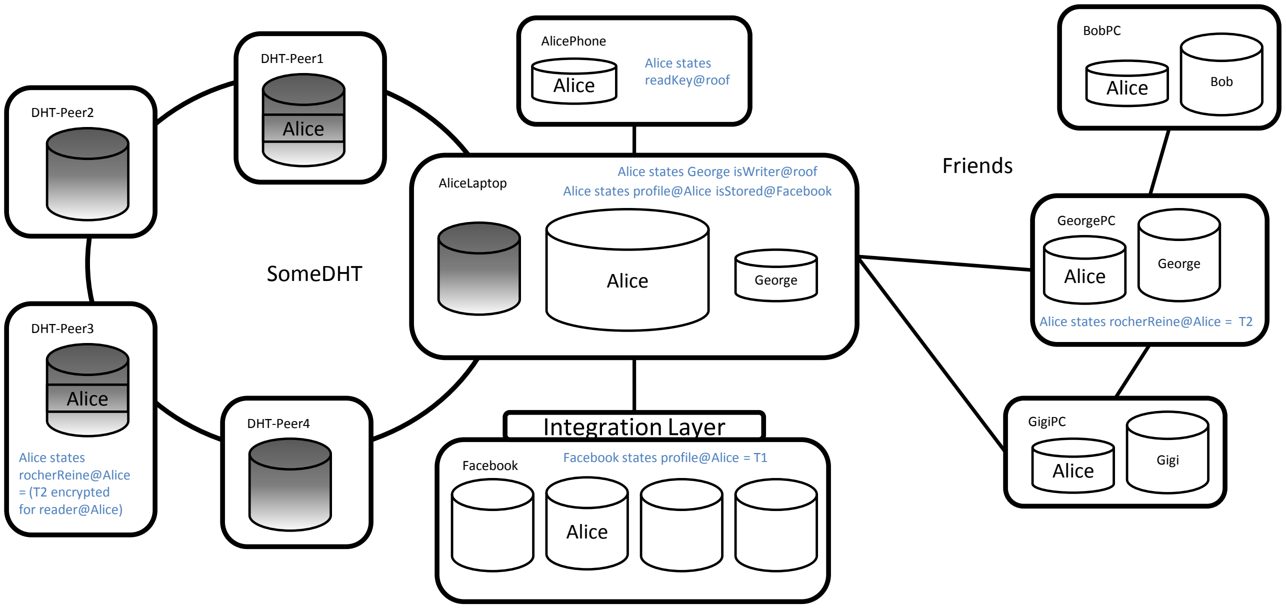
AlicePhone
Alice
Alice states readKey@roof
BobPC
Alice
Bob
DHT-Peer1
Alice
DHT-Peer2
AliceLaptop
Alice states George isWriter@roof
Alice states profile@Alice isStored@Facebook
Friends
Alice
George
GeorgePC
Alice
George
Alice states rocherReine@Alice = T2
SomeDHT
DHT-Peer3
Alice
Alice states rocherReine@Alice = (T2 encrypted for reader@Alice)
DHT-Peer4
Integration Layer
Facebook
Facebook states profile@Alice = T1
Alice
GigiPC
Alice
Gigi

Fig. 1. The distribution of Alice's data

II. Motivating example and challenges

The general setting of our motivating example is a group of rock-climbers who want to organize regular outings in the Fontainebleau forest. The users of the group use different schemes to host their data: their own computers, a centralized trusted peer (Facebook), an untrusted DHT and an unstructured P2P system of friends. They use the WebdamExchange system to transparently navigate and integrate their information while keeping total control on their data.

We now describe the motivating example in more detail. Alice wants to organize a regular rock-climbing outing in the Fontainebleau forest. Alice is part of a community of rock-climbers who use the WebdamExchange model (using directly the WebdamExchange system itself, or using a system bound to the model via an integration layer). Each user has a profile, with personal information (name, birth-date, address, email, phone number), a list of favorite spots (with location, difficulty and pictures) and a list of friends. Everyone in the list of friends can read this information but the user is the only one who can edit it.

Alice first creates a group "Rock-climbing Organization for Outing in Fontainebleau" (roof in short). She adds to this group her friends who are living close to Fontainebleau and her rock-climbing group of Paris 14th district, Roc14. She next creates a twitter-like discussion feed for the group, where users may post and read. She also creates a collaborative calendar where users may append, remove and read calendar entries (e.g. using XML representation of iCalendar standard format), with their availability. Finally, she creates a document to summarize the organization. She inserts in this document two XQuery services, one that summarizes the planning and one that aggregates the different rocks of Fontainebleau from the profile of the roof members and ranks them by popularity. She and her friend George are the only ones allowed to edit this document, but all members of the roof group can read it.

The data of Alice is distributed using different schemes (See Figure 1). Alice hosts some data on her laptop and on her smartphone. Of course, the data on her smartphone are usually not accessible from the web and her laptop is not running all the time, so she also replicates her data on an untrusted DHT, SomeDHT. She also has a favorite social network website, Facebook, and stores her profile information on this trusted peer. Finally, her friends are interested in some of her data, that they frequently use, and replicate it locally. Indeed, this unstructured P2P distribution is natural since people prefer to store data they care about and to interact only with friends.

The scenario described above raises several difficulties. We are particularly interested in the following.

Access control The management of access control is the most important requirement. In order to protect her data, Alice specifies different kinds of access rights, in particular read, write, append, remove, grant and revoke rights. The system should not permit illegal operations on the data. Unfortunately, it is not possible to prevent people from misbehaving outside of the system, e.g., a user may illegally send confidential data to another user. But we want to detect illegal operations resulting from such kinds of behavior whenever it is possible. To do so, the system needs to keep a full trace of the provenance, and support distributed monitoring.

Distribution As already discussed, an important problem on the web is the wide dispersion of the data. In particular, the data is spread between different peers, using different distribution schemes. In our example, Alice uses three different schemes at the same time. In the first one, Alice stores her data on a trusted peer. This peer can be her own personal computer or a trusted server like Facebook. She lets this peer manage her data on her behalf. In particular this peer has full control over her data. In the second scheme, Alice stores her data on an untrusted network of peers, SomeDHT. She does not want to delegate any right to these peers (nor read neither write), so her clients (for example her iPhone application) have to manage the updates themselves, by signing and encrypting all her information. In this context, granting rights amount to publishing keys encrypted for other users. Redundancy of the network is used to avoid denial of services (because of failure or out of malevolence). Finally, in the third scheme, Alice stores her data on a network of friends peers. She does not want to delegate more rights to her friends than what they already have. The data is replicated in clear on trusted friends, who are then in charge of enforcing access control.

A real system may even be more complex. Alice may have data on several social network websites (Facebook, Linked-In, MySpace). She may have several personal computers (at home, at work) and several mobile devices (smartphone, tablet). In such a highly distributed setting, the system has to provide an unified view of the access control and the localization of the data.

Data manipulation In this setting, providing a way to edit, navigate and query data is clearly necessary. The user should be able to localize all the data she has access to and to ask global structured queries to the system. Most of these queries are presented to the users as applications. For example, an application corresponding to a polling like doodle helps the user to deploy the corresponding data structures and to set the access control, and provides widgets to display the results. In our example, Alice installs typical applications like a collaborative calendar, a discussion feed and a domain-specific application to aggregate rocks preferences.

We next describe the main features of our system and explain how it solve these problems.

III. Main System features

Representation of data and meta-data The system manipulates a large variety of data, actual data (documents, lists) and meta-data (keys, localization information, access control), all as first class citizens. This is the subject of [3] and we will only briefly introduce it here. The actual data, e.g., the profile of Alice, a rock description or a discussion entry in the motivating example, is represented by XML documents. An unstructured document such as a picture is represented as a byte stream document.The result of a data update request is represented as a data statement of the form:

Alice *states* rocherReine@Alice = T

meaning that Alice replaced the content of her document *rocherReine*, describing her favorite rock, by the XML tree *T*. This statement is signed with the "write" signature of the roof group in order to enforce edit access control and by the signature of Alice to keep a trusted trace of provenance.

A list, e.g., the lists of favorites rocks, the discussion feed and the collaborative calendar in our example, is represented as a collection, that is a set of references to documents of the form:

Alice *states* rocks@roof += rocherReine@Alice

meaning that Alice added her document *rocherReine* to the *rocks* collection of the group *roof*.

The meta-data is similarly represented by special logical statements that can be manipulated like other data. These statements are used to define access control, to support key distribution used to enforce access control and to publish where information is localized. For example,

Alice *states* George isWriter@roof
Alice *states* readKey@roof
Alice *states* profile@Alice isStored@Facebook

meaning that Alice added George has a writer of the group *roof*, created a read key for the group and stored her profile on Facebook.

Control and monitoring As previously explained, all the statements and messages are authenticated by signature. The content of the statement may be encrypted if necessary, e.g. if it will transit through a participant that does not have the right to read it. Signature and encryption use the asymmetric keys distributed via the key statements. Statements and messages also keep traces of provenance, in order to avoid illegal operations and detect misuses. Provenance is recorded by the messages themselves when they are exchanged by piling-up signatures while statements record when and why they have been created.

The system guarantees that it prevents illegal operations and that every participant can successfully perform (directly or indirectly) any operation they are entitled to do. This is enforced by imposing restrictions on the manipulation of data. In particular, a participant of the system should check access rights before creating a statement on behalf of another user or sending an information in clear. It is discussed more precisely in [3].

Of course, no system can prevent illegal operations outside of the system. So our system has to enable a participant, who behaves legally, to detect illegal behaviors as soon as an illegal piece of information reaches her. Our system allows her to check authenticity and provenance of any fact and, if a fact is not valid, to determine when the misuses happened. More generally, the information of the knowledge base can be queried to solve interesting questions such as "How did Bob get access to the data of the roof group?".

Distribution schemes Section II introduced three basic schemes of distribution. The important notions are that the system provides localization statements to index which peers store the data. It also provides authentication and encryption to store data on untrusted peers. Replication and some special techniques like time to live and mutual certification are also used to avoid denial of services by malevolent peers. Thanks to the underlying model, these three schemes can be described using the same data model, which enables interoperability without losing the access control properties.

IV. Demonstration

Setting The demonstration will be based on the RockClimbing example of Section II. We will show how Alice can control and publish her data from different devices (iPhone, laptop) and how Alice's friends can access and manipulate the data based on their access rights. The demonstration relies on four components: a server-peer, deployed as an untrusted host, which stores encrypted data of any user in a persistent database, a client-peer and an iPhone-client, deployed as trusted hosts, which stores data of one user in memory and a Facebook-peer which binds the Facebook Graph API to our system.

For this demonstration, a web application has been designed to support a rock-climbing social network in a user-friendly manner on top of the WebdamExchange system. In particular, the user is able to create accounts, add friends, create groups, documents and collections and manage access rights easily. Building from the WebdamExchange peer, these kinds of web applications mostly consist in some interface to display and edit the documents. It would be similarly simple to build a personal information manager (such as Lotus Notes or Microsoft Exchange) or a synchronization tool with remote storage (such as Apple MobileMe or Microsoft Live Mesh). Of course, to be used as a commercial product, the system would need to be optimized to fit a huge volume of exchanges and to check security issues linked to web application deployment. Nevertheless, the core of the system and in particular the underlying model has already been designed with heavy usage in mind.

Peer Architecture The system is implemented in Java (or objective C for the iPhone client) and supports the fundamental parts of our model (notably authenticated and encrypted statements, query and update requests). A WebdamExchange peer (server-peer, client-peer, iPhone-client or Facebook-peer) is composed of four interacting classes, as illustrated in Figure 2. The *Store* class is in charge of local data management. The clients use an in-memory statement representation. On the server peer, storage and query processing are supported by an embedded native XML database, namely *eXist*. Statements are represented as objects with authenticated XML serialization. On the Facebook peer, Facebook is used as a trusted storage server. The *Security* class manages the key knowledge of the peer and all the encryption and signature operations. We use standard security libraries for that. This class isolates all the security operations from the rest of the code, making it easier to control this most sensitive part of the system (e.g., by supporting it in a separate smartcard). The *Communication* class handles communications between peers around standard SSL-secured Web services. It allows interactions between the different peers. Finally the *Manager* class is in charge of driving the system strategies. Different versions of the *Manager* class are used for the different peer roles in the scenario (client, server...).

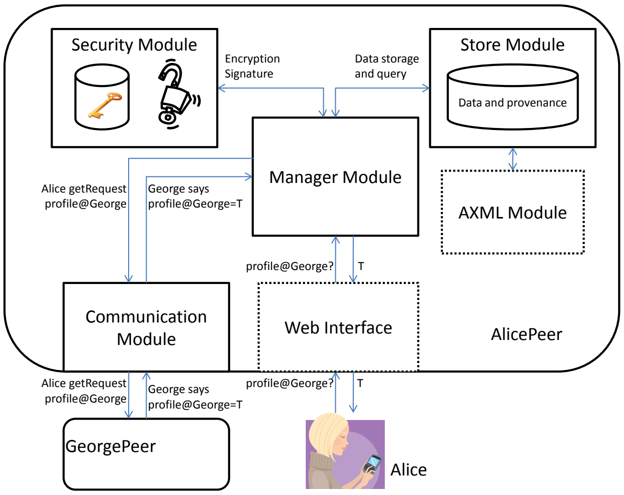

Fig. 2. WebdamExchange Peer Architecture

A WebdamExchange peer (with the exception of the iPhone client) is deployed as part of a Tomcat Apache server. It is integrated to an AXML peer, in order to support intensional data [1]. The precise description of the interaction with AXML is beyond the scope of this demonstration proposal. The iPhone client is deployed as an iPhone application on the App Store.

V. Conclusion

We demonstrate the use of a distributed knowledge base with access rights, localization and provenance, towards managing the sharing of information in a distributed context. The system supports transparent interaction of users with distributed data while preserving access control.

References

[1] S. Abiteboul, O. Benjelloun, and T. Milo. The active xml project: an overview. *VLDB J.*, 17(5):1019–1040, 2008.

[2] S. Abiteboul, M. Bienvenu, A. Galland, and M.-C. Rousset. Distributed datalog revisited. *Datalog 2.0 Workshop*, 2010 (To appear).

[3] S. Abiteboul, A. Galland, A. Marian, and A. Polyzotis. A model for web information management with access control. In preparation, draft on http://webdam.inria.fr/drafts/WebdamExchange.pdf.

[4] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta. PeerSoN: P2P social networking - early experiences and insights. In *2nd ACM Workshop on Social Network Systems*, 2009.

[5] M. Jawad, P. Serrano-Alvarado, and P. Valduriez. Protecting Data Privacy in Structured P2P Networks. In *Globe*, page 98, 2009.

[6] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the OceanStore prototype. In *2nd USENIX Conference on File and Storage Technologies*, pages 1–14, 2003.

[7] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. *ACM SIGOPS Operating Systems Review*, 35(5):188–201, 2001.