SP^{AC}: A Distributed, Peer-to-Peer, Secure and Privacy-aware Social Space

Angela Bonifati  
Italian National Research Council, Italy  
obonifati@icar.cnr.it

Hui (Wendy) Wang  
Stevens Institute of Technology, USA  
hwang@cs.stevens.edu

Ruilin Liu  
Stevens Institute of Technology, USA  
ruil3@cs.stevens.edu

ABSTRACT

To support privacy-aware management of data in social spaces, the user personal data needs to be stored at each user device, and shared only with a trusted subset of other users. To date, social spaces only have fairly limited access control capabilities, that do not protect the possibly sensitive data of the users.

In this demonstration, we showcase our SP^{AC} system, a distributed, peer-to-peer, secure and privacy-aware social space system. SP^{AC} is equipped with: (i) an SQL-based declarative distributed query language to specify which data to share and whom to share with. Such a language guarantees the fine-grained access to the data, (ii) a fully-decentralized authorization that relies on classic cryptographic protocols to provide robust and resilient key-based encryption for access control enforcement, and (iii) an update-friendly access control mechanism, that also addresses the updates on both the network and the access control policies.

Categories and Subject Descriptors: H.2 [Database Management]: Systems


Keywords: Access Control, P2P, Social Networks.

1. INTRODUCTION

Everyone has experienced the great potential of social spaces, as fostered by current software tools, e.g. Facebook, Twitter and GWave, to name a few. By default, such tools enable the users to share their own data with a selected subset of other users, by allowing to set up privacy parameters. It is important to argue that the personal data of the users should be kept on their own repositories, and not uploaded to centralized social servers. As more and more information is shared among social platforms for various reasons, e.g. collaborative work, shared calendar and multimedia content, we claim that protection of such data should be guaranteed in an inherently distributed fashion. We advocate a peer-to-peer architecture, where personal data is kept on the individual peers and selectively shared. Towards the goal of managing information in a highly distributed and partly confidential and secure environment, many important issues arise. Clearly, any user would like to prevent unauthorized parties to access her own personal space, her user profile and collaborative workspace.

So far, most approaches have only focused on one facet of the problem, i.e. exporting social networking to different platforms, such as ad-hoc networks of mobile phones [5], deploying personal data management applications in a trusted peer-to-peer architecture [3], or enriching online client-server social models with additional privacy benefits [1]. As far as access control is concerned, existing approaches offer great flexibility in terms of the definition and enforcement of access control and encryption, but do not provide means to handle highly distributed data. In a peer-to-peer setting, each peer may want to enforce access control rules on part of the data it owns, and possibly allow a subset of other peers to access its data.

In this demonstration, we showcase our SP^{AC} system, which is, to the best of our knowledge, the first full-fledged solution for protecting and sharing secure distributed data in distributed social spaces. SP^{AC} is equipped with the following capabilities: (i) an SQL-based declarative distributed query language that guarantees the fine-grained query-based access to the data; (ii) a fully-decentralized authorization, that relies on efficient key-sharing protocols [6] and ensures that an individual peer is not a single point of failure; (iii) an update-friendly access control mechanism, that addresses the updates due to both the network churn and the changes on the access control policies. To better illustrate the problem, we show in Figure 1 an application scenario that includes a social network of 8 peers, ranging from Peer_1 to Peer_8. Peer_1 wants to share the information on name and gender in her user profile with all the peers, but would like to restrict the access to her address to only those peers living in her state. Moreover, she is willing to share her private pictures with her family only. Access control is enforced by encryption via the decryption keys K_1, K_2 and K_3 in the three cases. A group of key pieces are generated from K_1, K_2 and K_3 that are necessary to decrypt the data for access; individual key pieces are of no use on their own to decrypt the data, according to the key-sharing protocol. As an example, assuming MyPictures has been encrypted by K_3, the latter will be split into six key pieces, out of which three are stored locally on Peer_1, and one each is distributed to Peer_2, Peer_3, and Peer_4 (key pieces are depicted as tiny shaded boxes in the figure). If her cousin Alice on Peer_2 wants to access MyPictures, she has to collect exactly 3 key pieces, necessary for key reconstruction. Therefore, she has to ask Peer_1 or look for at least other two key pieces, besides her own, at Peer_3 and Peer_4, respectively.

2. SYSTEM OVERVIEW

Data and Query Model. Each user in the social space acts as a data source peer. He/she stores the private data
Access Control Model. To specify which data is accessible to which peers, we define a declarative distributed access control language. The access control policy is expressed as an SQL query extended with a Peer clause, that might contain the peer_list as a list of peers $P_1, \ldots, P_n$ that have access to the data specified by the AC rules. Alternatively, the Peer clause might be expressed by means of an arbitrary SQL query on the relation peer_list, with an arbitrary conjunctive predicate $P \_WhereExpr$ on peer_list. Examples of policies are illustrated in Figure 1 on Peer_1 space.

Support for multiple AC policies and updates. In [2], we have addressed the challenges of key management when multiple access control rules access overlapping data values. Common data values are encrypted in the same encryption block by using the same key. We have also identified a monotone property for key shares on common values, that can significantly reduce the number of keys needed for the enforcement of access control rules. We have further investigated how to manage keys when there exist updates on access control policies, by proposing an effective scheme that preserves the monotone property of key shares in such cases, and shown the scalability and efficiency of our approach.

System architecture. Figure 2 illustrates the main components of our system. The user interface provides both command line (CLI) and graphical UI to users. The configurator allows users to configure their access control rules and queries via the user interface. SQL statements can be written manually by means of a text editor or composed through a PBE (Policy-By-Example) interface. Both the encryptor and the decryptor of the access control enforcer enforce the access control rules via encryption. The encryptor consists of key generation and distribution, while the decryptor consists of key reconstruction. The query evaluator interacts with the configurator and the access control enforcer to perform query evaluation on encrypted data.

Implementation. We have built our system $SP^{AC}$ in Java on top of Pastry [4], a DHT P2P network. We have measured the overhead of key management, the robustness with respect to network churn, and the query evaluation performance of the system. More details of the assessment results can be found in [2] and on our project website.

4. REFERENCES