INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Log Auditing Approach for Trust Management in Peer-to-Peer Collaboration

Hien Thi Thu Truong — Claudia-Lavinia Ignat

N° 7472

Decembre 2010

Domaine 2

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7472--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche NANCY - GRAND EST

A Log Auditing Approach for Trust Management in Peer-to-Peer Collaboration

Hien Thi Thu Truong , Claudia-Lavinia Ignat

Domaine : Algorithmique, programmation, logiciels et architectures
Équipes-Projets SCORE

Rapport de recherche n° 7472 — Decembre 2010 — 13 pages

Abstract: Nowadays we are faced with an increasing popularity of social software including wikis, blogs, micro-blogs and online social networks such as Facebook and MySpace. Unfortunately, the mostly used social services are centralized and personal information is stored at a single vendor. This results in potential privacy problems as users do not have much control over how their private data is disseminated. To overcome this limitation, some recent approaches envisioned replacing the single authority centralization of services by a peer-to-peer trust-based approach where users can decide with whom they want to share their private data. In this peer-to-peer collaboration it is very difficult to ensure that after data is shared with other peers, these peers will not misbehave and violate data privacy. In this paper we propose a mechanism that addresses the issue of data privacy violation due to data disclosure to malicious peers. In our approach trust values between users are adjusted according to their previous activities on the shared data. Users share their private data by specifying some obligations the receivers must follow. We log modifications done by users on the shared data as well as the obligations that must be followed when data is shared. By a log-auditing mechanism we detect users that misbehaved and we adjust their associated trust values by using any existing decentralized trust model.

Key-words: trust, privacy, peer-to-peer collaboration, log-auditing

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Une approche reposant sur l'audit pour la gestion de la confiance dans la collaboration pair-à-pair

Résumé : Depuis quelques années nous assistons à une explosion de la popularité des logiciels sociaux comme les wikis, les blogs, les micro-blogs ou encore les réseaux sociaux tels que Facebook et MySpace. Malheureusement, les plus usités de ces services reposent tous sur un contrôle centralisé ; les données d'un utilisateur se retrouvant centralisées chez un seul fournisseur de service. Cela engendre indéniablement un problème de confidentialité puisque les utilisateurs n'ont plus le contrôle sur la manière dont leurs propres données sont diffusées. Afin de surpasser ces limitations, plusieurs approches récentes proposent de remplacer le contrôle centralisé de ces services par des approches pair-à-pair reposant sur des mécanismes de confiance où chaque usager décide avec qui il souhaite partager ces données personnelles. Toutefois, dans ce genre de collaboration, il est très difficile de s'assurer qu'une fois une donnée partagée avec des pairs, celle-ci ne soit pas divulguée à d'autres pairs non autorisés. Dans cet article, nous proposons un mécanisme qui permet de circonscrire les problèmes de violation de confidentialité liés à la divulgation de données par des pairs malicieux. Dans notre approche, les indices de confiance entre les usagers sont ajustés selon le comportement passé des usagers vis-à-vis des données partagées. Lorsqu'un usager partage des données, il impose aux autres usagers des obligations qu'ils doivent respecter. Chaque modification sur une donnée partagée effectuée par un usager et chaque obligation est répertoriée dans un journal. En réalisant un audit de ce journal, notre approche détecte les usagers qui se sont mal-comportés et ajuste en conséquence leurs indices de confiance.

Mots-clés : confiance, confidentialité des données, collaboration pair-à-pair,audit

1 INTRODUCTION

Social software including wikis, blogs, micro-blogs and social networks has emerged as a new interpersonal communication form. Existing micro-blogging services such as Twitter and social networks such as Facebook or MySpace have millions of users using them everyday. While these social services offer many attractive functionalities, they require storing personal information in the hands of a single large corporation which is a perceived privacy threat. Users are obliged to provide and to store their data to vendors of these services and to trust that they will preserve privacy of their data, but they have little control over the usage of their data after sharing it with other users. These corporations could produce a profile based on the individual behavior and therefore detrimental decisions to an individual may be taken. Moreover, due to large amounts of information these social services sites process every day, a single flaw in the system could permit retrieval of large parts of personal data. For instance on Facebook features such as messages, invitations and photos help users gain access to private information. Moreover, flaws in the Facebook's third-party API have been found which allow for easy theft of private information.

Some recent approaches such as [2] proposed moving away from centralized authority-based collaboration towards a peer-to-peer trust network where users have full control over their personal data that they store locally and can decide with whom to share their data. Users define their network of trust containing people that they trust and with whom they wish to collaborate. These peer-to-peer networks of trust overcome the disadvantages of centralized architectures by offering a good scalability and fault-tolerance and the possibility of sharing costs of administration. In a peer-to-peer collaboration model rather than having a central authority which has access to all users personal data, control over data is given to users. Therefore, the risk of privacy breaches is decreased as well as only a part of the protected data in the peer-to-peer network may be exposed at any time. However, in this peer-to-peer collaboration it is very difficult to ensure that after data is shared with other peers, these peers will not misbehave and violate data privacy. To prevent data misuse, trust management mechanisms are used where peers are assigned trust values and a peer collaborates only with high trusted peers. However, to our knowledge, there exists no approach that automatically updates trust values according to peers misbehavior.

In this paper we propose an approach of log auditing for computing trust in a peer-to-peer environment according to respecting obligations peers receive from other peers concerning their private data. We also propose a novel audit-based compliance control approach suited for distributed collaborative environments where obligations are checked a-posterior and not enforced a-prior. This approach in which usage policies are checked posteriorly is different from prior-checked access control mechanisms. Rather than requiring a hard security mechanism, our solution uses a trust-based approach that is more flexible for users.

The rest of this paper is organized as follows. Section 2 is an overview of Peer-to-Peer trust. Section 3 presents our log auditing approach in decentralized systems. Then we describe the formal structure of log in Section 4. In Section 5 we present a discussion on obligations that are associated to logs. Section 6 describes the mechanism for local trust assessment with algorithm analysis.

Section 7 compares our approach with related works and Section 8 presents concluding remarks and directions of future work.

2 PEER TO PEER TRUST OVERVIEW

Peer-to-peer underlying architectures reflect society better than other types of computer architectures [3], being better adapted to the way people think and to user's needs for knowledge sharing and providing users more freedom to interact with each other. In this peer-to-peer collaboration model, it is very difficult to ensure data privacy. According to [20], data privacy is the right of individuals to determine for themselves *when*, *how*, and *to what* extent information about them is communicated to others. A peer shares his private data only with peers that he trusts, so, privacy of data is preserved for a direct connection with a peer. But the main issue refers to what happens to data released to authorized persons, i.e how the user may, must and must not use it. This issue is called usage control [13, 4] and is modeled by means of certain obligations that users receive together with data.

Trust is a belief or confidence in the honesty, goodness of a person or organization. In [19] a classification of trust models is given. A trust level is an assessment of the probability that a peer will not cheat. An honest peer will be assigned a high trust level while a malicious peer will be assigned a low trust level. These trust levels are updated according to the peer's behavior. If a peer misbehaves, its trust level is decremented. The solution that we propose in this paper for adjusting trust levels of peers according to their behavior is general and could be combined with any existing reputation mechanism.

In order to present an overview of our approach let us consider an example in the domain of data sharing in a social network. Suppose Alice creates a document and she wants to share it with different friends, say Bob and Carol. She shares it to Bob with a certain right to *modify* it. It is very difficult to enforce Bob to follow that policy in a decentralized environment. Bob can do any action on the document once received it. There is no way for Alice to guarantee Bob will not misbehave on the document after it has been shared. In our approach we propose a mechanism logging past actions of users concerning shared data. Bob's actions will be logged by the system. Alice will never know what Bob has done with her data if Bob only keeps the log locally. But his log of local edit actions will be disclosed to whom he continues to re-distribute data. If Carol receives the document, she can check the log to know what actions that Alice and Bob did.

Our system does not aim to prevent fraud. Rather, the log mechanism provides audit capabilities in order to detect attempts at fraud after the data has been shared and used. The local actions on data and the communications between peers are assumed to leave some evidence and hence are observable. The owner attaches a usage policy to the data in order to specify what actions are allowed and under which conditions. According to this a-posterior checking, user trust values are updated with a decrement. For checking compliance to obligations, we audit the log containing modifications done by users on the shared data as well as the obligations that must be followed. Each user evaluates trust on other users and keeps trust values locally instead of storing them at a central authority. Trust values help users decide to collaborate or not with

other users. These values are updated after each posterior checking of the log for detection of misbehaved users. We can use any trust model for updating trust values. To our knowledge, our approach is the only one that addresses data privacy violation by discovering malicious users and updating user trust values.

3 LOG AUDITING APPROACH

Our system consists of a group of communicating peers. Each peer has its own workspace. These peers collaborate together in creating and sharing data in a decentralized environment where no central administration point exists. Users are administrators themselves.

The local edit actions and communication actions among peers are logged by the system in *edit log* and *communication log*. Each user keeps locally one edit log and one communication log. When a user shares the document with others, logs and usage policies will be associated with the document. The policies are specified in communication log. Initially, the log is empty, but after certain iterations, as observations are made, the log will grow up. The logs are created under the following assumptions:

- ◇ Logs are created automatically by system and they are unalterable. This assumption is practical. In reality, logs could be changed but there are some techniques as in [9] to detect or avoid log modification.
- ◇ It is required that at least one obligation is given in a sharing action. In collaboration, when the document is sent back to previous sender, it is not required to re-define new obligations.
- ◇ The occurring order of events stored in logs is maintained by logical clock [11].

In a collaboration-based system, users are expected to behave correctly, but they might be suspected of incorrect behaviors. A user violates an obligation if he performs actions which are not permitted in usage policies. We update decrementy their trust values each time violations are detected. The trust value for well-behaved peers is higher than the trust value for malicious peers. A peer has initial trust values associated with other peers. They are calculated and adjusted after each log analysis.

Log auditing consists in the analysis of both edit and communication logs. Each time the user receives different versions of a same document, the system automatically analyzes the logs in order to judge the past behaviors of other users. An important point in checking past behaviors is to detect mismatching of actions and obligations them. Next, the received logs that include past actions and obligations are checked to be merged with current local logs. If the logs should be merged, both document edit logs and communication logs are merged. In addition to merging obligations, conflict resolution between rights is required.

4 LOG STRUCTURE

In this section, we present the structure of logs and give an example to illustrate how logs are created and stored locally at sites of peers.

Definition 1. *An event is denoted as* $e = a_r^s$ *where* a *represents an action that can be either a local action on document or a communication action, and* r *represents parameters which are in form of pair of name and value. The notation* s *has the temporal meaning for an event, in which* $s = -1$ *represents the actual event that a user performed while* $s = +1$ *represents the obligation event he has to follow.*

Similar to event structure in Z language [16], each event in our log is composed of an action or an obligation and several parameters. For instance, $share^{-1}_{\{by,P_1\},\{to,P_2\}}$ is an event of sharing the document from P_1 to P_2 (P_1 shared document to P_2) while $share^{+1}_{\{by,P_1\},\{to,P_2\}}$ is an obligation P_1 gives to P_2 (P_2 can share the document).

Definition 2. *A log is composed of a time-ordered sequence of pairs (logical clock, event):* $[(c_1, e_1), (c_2, e_2),..., (c_m, e_m)]$.

For ordering events we use the logical clock with *happened_before* relation [11] among events. Event e_1 is ordered before event e_2 if e_1 happened before e_2. The system of each site maintains a counter that is incremented each time an event is generated at that site. Each event is assigned the value of the counter at the moment of its generation. This counter called also logical time c_j is simply used to order events according to their order of occurrence. The logical clock of obligations of sender is replaced by the new logical clock of receiver according to the order when he receives the document. This helps checking if events generated by a user conform to obligations previously received. We can track backward the logical clock value assigned to obligation by sender through the logical clock value of *share* event. From the logical clock of *share* action, we know when such obligation events are shared by the sender.

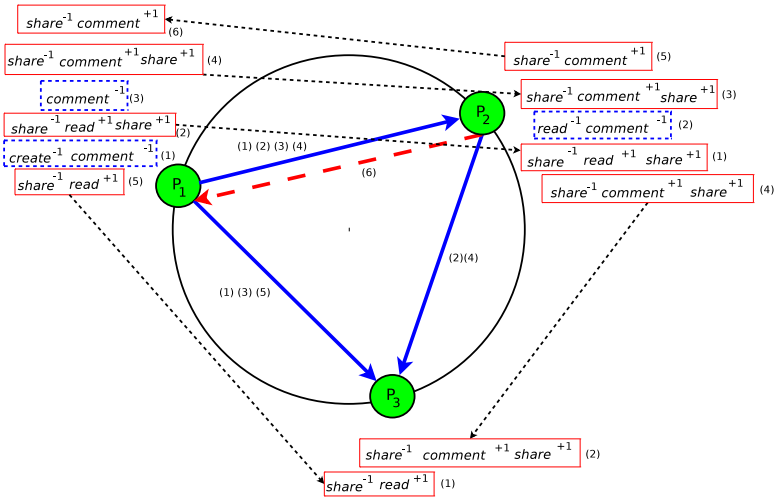

Figure 1: Logs for 3 peers collaboration, *edit* actions are inside dash box and *communication* actions are inside line-box. Only local edit actions of each peer and obligations are showed in this figure.

Consider an example of sharing data in a distributed peer to peer social network. Users can *share* photos, videos or music documents between them and add *comment* to these documents (but they can not modify the content of these

documents). The obligations could be: *"may read"*, *"may not read"*, *"may add comment"*, *"may not add comment"*, *"may delete comment"*, *"may not delete comment"*, *"may share"*, *"can not be shared"*. Figure 1 shows an example of three peers P_1, P_2, P_3 sharing photos. Let P_1 be the creator of data d. P_1 shares d with P_2, and then P_2 shares it with P_3. In parallel, P_1 shares the same data directly to P_3. The logs of actions (edit log l_{P_x-edt} and communication log l_{P_x-com}) are created locally at peers as follows:

1. At the local site, P_1 creates document d for which he adds a comment and shares this document with P_2 with the usage obligation *" may read"*, *"may share further"*. Logical clock starts from 1.

   $\mathbf{l_{P_1-edt,d}} = [(1, create^{-1}_{\{by,P_1\}}), (1, comment^{-1}_{\{by,P_1\}})]$;

   $\mathbf{l_{P_1-com,d}}$ (with P_2) $= [(2, share^{-1}_{\{by,P_1\},\{to,P_2\}}),$
   $(2, read^{+1}_{\{by,P_1\},\{to,P_2\}}), (2, share^{+1}_{\{by,P_1\},\{to,P_2\}}),$
   $(2, not\ comment^{+1}_{\{by,P_1\},\{to,P_2\}})]$

2. P_2 receives document d with associated logs and adds a comment to this document. The edit log will be updated continuously. Note that the logical clock of obligations in communication log is updated to the value of the local logical clock at the site P_2. Note also that P_2 received document d with the permission of reading it and sharing it further, but without the permission of commenting it. However he did the comment action, therefore, P_2 violated the received obligation.

   $\mathbf{l_{P_2-com,d}} = [(2, share^{-1}_{\{by,P_1\},\{to,P_2\}}), (1, read^{+1}_{\{by,P_1\},\{to,P_2\}}),$
   $(1, share^{+1}_{\{by,P_1\},\{to,P_2\}}), (1, not\ comment^{+1}_{\{by,P_1\},\{to,P_2\}})]$

   $\mathbf{l_{P_2-edt,d}} = [(1, create^{-1}_{\{by,P_1\}}), (1, comment^{-1}_{\{by,P_1\}})] \cup$
   $[(2, read^{-1}_{\{by,P_2\}}), (2, comment^{-1}_{\{by,P_2\}})]$;

3. After sending to P_2, P_1 adds another comment to document d and re-sends to P_2 this document with obligation *"may add comment"*. The logical clock of P_1 is increased after each action. P_1 also shares data with P_3 with the same obligation *"may add comment"*.

   $\mathbf{l_{P_1-edt,d}} = [(1, create^{-1}_{\{by,P_1\}}, (1, comment^{-1}_{\{by,P_1\}}),$
   $(3, comment^{-1}_{\{by,P_1\}})]$;

   $\mathbf{l_{P_1-com,d}}$ (with P_2) $= [(2, share^{-1}_{\{by,P_1\},\{to,P_2\}}),$
   $(2, read^{+1}_{\{by,P_1\},\{to,P_2\}}), (2, share^{+1}_{\{by,P_1\},\{to,P_2\}}),$
   $(2, not\ comment^{+1}_{\{by,P_1\},\{to,P_2\}}),$
   $(4, share^{-1}_{\{by,P_1\},\{to,P_2\}}), (4, comment^{+1}_{\{by,P_1\},\{to,P_2\}})]$
   $\mathbf{l_{P_1-com,d}}$ (with P_3) $= [(5, share^{-1}_{\{by,P_1\},\{to,P_3\}}),$
   $(5, comment^{+1}_{\{by,P_1\},\{to,P_3)\}})]$.

4. P_2 receives document d again from P_1 and then shares it with P_3 with the obligation *"not share"* to do not share further the document. The edit log of P_2 is updated to include the last action done by P_1 of commenting the document.

   $\mathbf{l_{P_2-com,d}}$ (local) = $[(2, share^{-1}_{\{by,P_1\},\{to,P_2\}}),$
   $(1, read^{+1}_{\{by,P_1\},\{to,P_2\}}),\ (1, share^{+1}_{\{by,P_1\},\{to,P_2\}}),$
   $(1, not\ comment^{+1}_{\{by,P_1\},\{to,P_2\}})]$
   $\cup\ [(4, share^{-1}_{\{by,P_1\},\{to,P_2\}}),\ (3, comment^{+1}_{\{by,P_1\},\{to,P_2\}})$
   $\mathbf{l_{P_2-edt,d}} = [1, create^{-1}_{\{by,P_1\}},\ (1, comment^{-1}_{\{by,P_1\}})]$
   $\cup\ [(2, read^{-1}_{\{by,P_2\}}),\ (2, comment^{-1}_{\{by,P_2\}})]\ \cup$
   $[(3, comment^{-1}_{\{by,P_1\}})]$
   $\mathbf{l_{P_2-com,d}}$ (with P_3) = $\mathbf{l_{P_2-com,d}}$ (local) $\cup$
   $[(4, share^{-1}_{\{by,P_2\},\{to,P_3\}}),(4, not\ share^{+1}_{\{by,P_2\},\{to,P_3\}})]$.

5. At local site of P_3, suppose that P_3 receives document d from P_1 before receiving it from P_2. The local communication log of P_3 is obtained by merging $\mathbf{l_{P_1-com,d}}$ (with P_3) computed at step 3 with $\mathbf{l_{P_2-com,d}}$ (with P_3) computed at step 4.

   $\mathbf{l_{P_3-com,d}} = [(5, share^{-1}_{\{by,P_1\},\{to,P_3\}}),$
   $(1, comment^{+1}_{\{by,P_1\},\{to,P_3\}})] \cup l_{P_2-com}$ (local)
   $\cup\ [(4, share^{-1}_{\{by,P_2\},\{to,P_3\}}),\ (2, not\ share^{+1}_{\{by,P_2\},\{to,P_3\}})]$
   $\mathbf{l_{P_3-edt,d}} = \mathbf{l_{P_2-edt,d}}$

In order to detect cheaters, each peer analyzes the received logs. A user with actions that do not conform to obligations is considered as a cheater. In the above example, P_3 detects the violation of action *comment* of P_2 in l_{P_2-edt}.

5 OBLIGATIONS

Collaborating in a distributed system makes a user possible to receive document from many collaborators. In the previous example P_3 receives the same document from P_1 and P_2. P_3 will update document based on the received changes under certain obligations. Up on the obligations, the system decides to accept or reject the new copies of data.

When a user receives different obligations, the conflict between obligations may occur. An obligation conflict means the subjects are both required and required not to perform the same actions on target objects. In multi-policies environment, it is possible that one policy overrides another. Conflict detection should be performed in order to decide which usage policy is performed and which is ignored. Moreover, in case a user receives a set of obligations instead of a single obligation, the conflict may occur between sets. Two sets are in conflict if they contain at least one conflict between two single obligations. One of the solution to avoid conflict is giving priority to certain obligations.

Obligations can be ordered based on the ability they offer to work on data. We denote two obligations α_1, α_2 that $\alpha_1 > \alpha_2$ if α_1 has more ability to work on document than α_2. For example $add-comment > read$ and $share > not\ share$. The comparison of two sets of obligations can be performed based on comparing each single obligation belonging to the two sets. For example, $[add-comment, share] > [not\ add-comment, share]$.

With the approach using obligations in sharing document, if some obligation is not specified, for example, neither *share* nor *not share* is given, peers can do the actions as they want without any violation, e.g they can either *share* or *not share* the document.

In obligation-based collaborating systems, the more user respects obligations, the more trust he gains, and the more possibility others want to collaborate with him.

Unlike single centralized system, in distributed P2P application, peers are faced with conflicts between rights and obligations referring to the same document, but also between changes on the same document. When a peer receives many copies of the same document, it analyzes the associated logs in order to assess the local trust values of other peers who collaborated on copies. Afterward that peer checks for merging the logs and the document. If the obligations permit that peer to get the changes on document, a merge algorithm is performed. Due to space limitation, we do not present in this paper our algorithms related to merging document and ordering rights and obligations.

6 TRUST ASSESSMENT

In our decentralized system, each peer evaluates trustworthiness of other peers based on its experience. During collaboration between peers, trust values are adjusted mainly upon the result of log analysis. Checking a log is a basic mechanism to detect cheaters and help to predict the probability that they will continue cheat in future actions.

We denote $T_{P_i}^{log}(P_j)$ as the trust value that a peer P_i evaluates and assigns to peer P_j. In order to manage trust values for peer P_j, we can use any existing decentralized trust model. The trust values are initially assigned a default value by system.

The algorithm 1 takes as input linear logs (edit log and communication log). This is a local algorithm that peers can apply in order to determine trust on other peers over the collaborative network. The peer P_i updates value $T_{P_i}^{log}(P_j)$ for peer P_j based on the result of log analysis.

All peers are set the highest trust value at the beginning (max_trust_value). In order to detect misbehaviors, peer's actions are considered violating the obligations if there is one right or obligation which not permit to do that action. With each event in log, parameter (by, P_i) helps extracting the user P_i who performed the action. We consider action is made by P_i at logical time c. If P_i is the creator of the document, it has full rights to do any action on the document, therefore, no need to check for its obligation. In case P_i has received the document from another peer, we will check for its actions to compare with the given obligations. The obligations are kept as a special "event" in communication log $(c, (b)_r^s)$ with $s = +1$. We extract parameter (to, P_j) from this "event". If $P_j = P_i$ and the logical clock value c_j when obligation was received is less than

Algorithm 1: LOCAL-TRUST-ASSESSMENT

Input: The edit log l_{edt} and the communication log l_{com}, the document d, user A who assessing trust.
Output: $T_A^{log}(P_i)$ for each P_i that appears in logs.

```
begin
    for each (c, e = (a)_r^s) with s = −1 do
        misbehaved = FALSE;
        extract a, P_i in {by, P_i} from r;
        T_A^log(P_i) = max_trust_value;
        if P_i is not the creator of d then
            k = lengthOf(l_com);
            checked = FALSE;
            while (k ≥ 1) and (checked = FALSE) do
                get (c_k, e_k = (b)_r^s) ∈ l_com ;
                extract b, P_j in {to, P_j} from e_k ;
                if (s = +1) and (P_j = P_i) and (c_k < c) then
                    if (b = not a) then
                        misbehaved = TRUE;
                        checked = TRUE;
                    end
                end
                k = k − 1;
            end
            if (misbehaved = TRUE) then
                adjust decemently trust value T_A^log(P_i) based one specific trust
                model;
            end
        end
    end
end
```

logical clock value c_i when action was performed, that action is considered valid. As the logical clock of obligation is transformed from sender's to receiver's, we can check whether an action was done before or after a peer received the corresponding obligation. It should be noticed that rights or obligations are possible to be overridden and the latest ones are taken in account in our algorithm only.

When an assessed peer P_j is detected as a cheater, its local trust value is decremented by assessing peer P_i. The local trust values could be aggregated from log-based trust, reputation or recommendation trust. That depends on the trust model being used. Research on the trust models is out of scope of this paper.

Our algorithm serves for trust assessment by using logs. The violation in case a cheating user copies the content of document to create a new one, then claims him as owner can not be detected by using log auditing itself. However, communication log could be used to discover the history actions on document that helps to detect cheaters.

7 Related Work

In this section we first compare our work with some approaches that address data privacy in peer-to-peer systems. Then we continue by describing and comparing our proposed mechanism with other approaches that use some related solutions to our approach but in different contexts.

In order to return data ownership to users rather than to a third party central authority, some recent works [2, 21] explore the coupling between social networks and peer-to-peer systems. In this context privacy protection is understood as allowing users to encrypt their data and control access by appropriate key sharing and distribution. Our approach is complementary to this work and refers to what happens to data after it has been shared.

Another approach that addresses data privacy violation in peer-to-peer environments is Priserv [6], a DHT privacy service that combines the Hippocratic database principles with the trust notions. Hippocratic databases enforce purpose-based privacy while reputation techniques guarantee trust notions. However, this approach focuses on a database solution, being limited to relational tables. Moreover, as opposed to our solution, the Priserv approach does not propose neither a mechanism of discovering the malicious users that do not respect the obligations required for using the data nor an approach for updating the trust values associated to users.

OECD (Organization for Economic Cooperation and Development) defined basic privacy principles including: collection limitation, data quality, purpose specification, use limitation, security safe, openness, individual participation, accountability. We consider data privacy in collaborative working from the point of *use limitation* that users will specify how their data may and may not be used. We consider the decentralized system which documents are exchanged and shared among users. When a user receives a document, he is expected to work on the document by respecting obligations. The log mechanism is used to detect cheaters who do not respect their obligations. Unlike access control which is concerned with granting access to sensitive data based on conditions that relate to past or present, obligation which impose conditions on the future is concerned with commitments of the involved users. At the moment access is granted to data, adherence to these commitments cannot be ensured. The formal framework in [5] allows specification of obligations. They present different mechanisms for checking adherence to commitments. However, all their proposed solutions are based on a central reference monitor that can ensure that data protection requirements are adhered to. As opposed to our approach, these solutions are not suitable for peer-to-peer environments where there is no central authority.

Keeping and managing event logs is frequently used for ensuring security and privacy. This approach has been studied in many works. In [7], a log auditing approach is used for detecting misbehavior in collaborative work environments, where a small group of users share a large number of documents and policies. In [10, 17], a logical policy-centric for behavior-based decision-making is presented. The framework consists of a formal model of past behaviors of principals which is based on event structures. However, these models require a central authority to audit the log to help the system making decisions and this is a limitation for using these models in a fully decentralized environment.

Trust management is an important aspect of the solution that we proposed. The concept of trust in different communities varies according to how it is computed and used. Our work relies on the concept of trust which is based on past encounters: "Trust is a subjective expectation an agent has about another's future behavior based on the history of their encounters" [12]. Various trust models for peer to peer systems exist such as NICE model [18], EigenTrust model [8] and global trust model [1] and our mechanism for discovering misbehaved users can be coupled with any existing trust model in order to manage user trust values.

8 Conclusion

Our vision is to replace central authority-based social software collaboration with a distributed collaboration that offers support for decentralization of services. In this context, our paper addressed the issue of data privacy violation due to data disclosure to malicious peers in a peer-to-peer collaboration. In our collaboration model users share their private data by specifying some obligations the receivers must follow. Modifications done by users on the shared data and the obligations that must be followed when data is shared are logged in a distributed manner. A mechanism of distributed log auditing is applied during collaboration and users that did not conform to the required obligations are detected and therefore their trust value is updated. Any distributed trust model can be applied to our proposed mechanism. Users can perform concurrent modifications on the shared documents as well as they can share documents with different specified obligations according to their preferences.

A direction of future work is the evaluation of the proposed mechanism. We will test first the efficiency and complexity of our algorithms in peer-to-peer simulators such as PeerSim [14]. We plan afterward to apply our trust management approach to existing research peer-to-peer online social networks such as PeerSoN [15].

References

[1] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the tenth international conference on Information and knowledge management*, 2001.

[2] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta. PeerSoN: P2P social networking - early experiences and insights. In *Proceedings of the Second ACM Workshop on Social Network Systems 2009, co-located with Eurosys 2009*, Nürnberg, Germany, March 31, 2009.

[3] D. Clark. Face-to-Face with Peer-to-Peer Networking. *Computer*, 34(1):18–21, January 2001.

[4] M. Hilty, D. A. Basin, and A. Pretschner. On obligations. In *ESORICS*, pages 98–117, 2005.

[5] M. Hilty, D. A. Basin, and A. Pretschner. On Obligations. In *European Symposium on Research in Computer Security (ESORICS)*, 2005.

[6] M. Jawad, P. Serrano-Alvarado, and P. Valduriez. Protecting data privacy in structured p2p networks. In *Globe*, pages 85–98, 2009.

[7] J.G.Cerderquist, R.Corin, M.A.C.Dekker, S.Etalle, J. Hartog, and G.Lenzini. Audit-based Compliance Control. *Information Security*, 9(1):37–49, march 2007.

[8] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International Conference on World Wide Web*, 2003.

[9] B. B. Kang, R. Wilensky, and J. Kubiatowicz. The Hash History Approach for Reconciling Mutual Inconsistency. In *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS '03)*, 2003.

[10] K. Krukow, M. Nielsen, and V. Sassone. A logical framework for history-based access control and reputation systems. *Journal of Computer Security*, 16(1):63–101, 2008.

[11] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558 – 565, July 1978.

[12] L. Mui and M. Mohtashemi. A computational model of trust and reputation. In *Proceedings of the 35th Hawaii International Conference on System Science (HICSS)*, 2002.

[13] J. Park and R. Sandhu. The UCONABC usage control model. *ACM Transactions on Information and Systems Security*, 7(1):128–174, 2004.

[14] Peersim. http://peersim.sourceforge.net/.

[15] Peerson. http://www.peerson.net.

[16] A. Pretschner, F. Schutz, C.Schaefer, and T.Walter. Policy Evolution in Distributed Usage Control. In *Electronic Notes in Theoretical Computer Science*, 2008.

[17] M. Roger and J. Goubault-Larrecq. Log Auditing through Model-Checking. In *Proceedings from the 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, 2001.

[18] B. B. Seungjoon Lee, Rob Sherwood. Cooperative Peer Groups in NICE. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 50(4):523 – 544, 2006.

[19] Q. H. Vu, M. Lupu, and B. C. Ooi. *Peer-to-Peer Computing*. Springer, Singapore, 2009.

[20] A. F. Westin. *Privacy and Freedom*. NewYork, 1967.

[21] D. I. Wolinsky, P. S. Juste, P. O. Boykin, and R. Figueiredo. Oversoc: Social profile based overlays. In *Proceedings of the workshop on Collaborative Peer-to-Peer Systems (COPS 2010)*, TEI of Larissa, Greece, June 2010.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399