Secure and Flexible Framework for Decentralized Social Network Services

Luca Maria Aiello, Giancarlo Ruffo
Computer Science Department - Università degli Studi di Torino
Corso Svizzera 185, 10149 Turin, Italy
{aiello,ruffo}@di.unito.it

***Abstract*—The rapid growth of the volume of user-generated contents in online social networks has raised many privacy concerns, mainly due to the data exploitation operated by providers. In order to address this problem, the idea of supporting social network services with open peer-to-peer systems has gained ground very recently. Nevertheless, the development of social network applications on decentralized layers involves several new security and design issues. In this paper we define an architectural model which embeds user identity management in a DHT overlay, providing a very robust and flexible support for any identity-based application. Important features for social applications like reputation management, modular expandability of the application suite and discretionary access control to shared resources can be easily implemented on our framework.**

***Keywords*-social networks, privacy, access control, peer-to-peer**

I. Introduction

Online social networks (OSNs) has surprisingly proliferated and evolved in the last few years, attracting the attention of increasingly larger portions of web users. Accordingly to the user-centered Web 2.0 paradigm, contents in *social networking services* (SNSs) are generated and updated by participants themselves. Nowadays, the most popular OSNs (e.g. Facebook, Flickr, MySpace) boast several millions of users and manage far more user-generated contents.

Such huge volume of information, kept in the hands of SNS providers, poses some concerns on the privacy and on the right to use the data; an emblematic example of this problem is the recent controversy on Facebook content management policy[1]. As a matter of fact, when a single control entity retains user information, undesirable mining and exploitation of user data can be easily achieved.

A drastic but effective solution to this privacy problem is to replace the centralized architecture hosting the SNS with a peer-to-peer (p2p) system. If the resources are scattered among the social network participants and the content lookup procedure is fully distributed, then there is no need to involve any centralized control on information exchanged; network crawling activities aimed to data collection can be avoided through content encryption.

Obviously, this radical change of design perspective implies new design and security issues [1]. Some of the basic SNS requirements like non-stop availability of data, synchronous updates, authentication, confidentiality and resource access control are often not trivial to be efficiently implemented over a pure p2p layer. Furthermore, decentralized systems like structured p2p networks (DHTs) are extremely vulnerable to a wide range of attacks [2] that undermines the safety and robustness of overlying applications.

Given this setting, it is necessary to reconcile, through a robust decentralized framework, the user demand for data privacy with the need of application reliability and security. This direction has been followed by some very recent works (see Section II) but, since this research line is still in an early stage, several security aspects could be analyzed thoroughly and different design alternatives deserves to be explored.

This paper follows this direction by defining a DHT-based architectural framework for social networking applications development. The relevant aspects of our contribution are the following.

First, the DHT we use is an enhanced version of Kademlia which embeds user identity management, allowing user authentication at routing level. Such customization enacts an effective protection against well known overlay attacks, thus granting higher reliability to applications above, compared to classical DHTs. Second, the identity-based primitives offered by this DHT allow to easily implement important social networking features: in particular, discretionary access control and reputation management are analyzed in detail. Furthermore, our approach allows a flexible integration between different applications and a sharp, identity-based resources retrieval from the DHT. Lastly, we put forward a proposal for the integration of a distributed tag-based search engine in the p2p OSN, in order to better fill the functional gap with corresponding web-based services.

The rest of the paper is organized as follows. In Section II an overview on related works is given. In Section III the main features of the p2p layer used in our framework are described. In Section IV the proposed decentralized SNS architecture is drawn. Conclusions are given in Section V.

II. Related works

Even if attempts in building social applications over a distributed layer were made in the past (e.g. [3]), the idea of exploiting p2p frameworks in order to solve SNSs privacy issues is more recent [1], [4].

[1] http://news.cnet.com/8301-13577_3-10165190-36.html

978-1-4244-5328-3/10/$26.00 ©2010 IEEE

The *PeerSon*[2] project [5] focuses on privacy preservation in OSNs through encryption of contents stored in the DHT. The p2p network is used mainly as a contact lookup service: once a pair of friends retrieve each other's address from the DHT, the interaction can continue through direct connections. When a friend is unreachable, offline asynchronous notifications and messages are stored in the DHT first, and retrieved afterward by the addressee.

The *Safebook*[3] system [6], combines the DHT functionalities with a p2p trust network called *Matryoshka*. Resources are stored at highly trusted peers and the content lookup is made by searching a pointer to a Matryoshka member through the DHT. A trusted, offline identification service is introduced mainly to avoid Sybils; encryption is used to preserve privacy as well.

A different approach, proposed in [7], consists in a gossip protocol, implemented on the Tribler network [8], for OSN participants discovery. When a searched friend is unavailable and the search initiator goes offline, a set of online *helper* peers is delegated to carry on a probing activity in order to find the target peer.

III. DHT LAYER FEATURES

The Distributed Hash Table at the basis of our architecture is a secure version of Kademlia [9], called *Likir*, whose main feature is that every p2p node is indissolubly bound to a user identity. The choice of applying such identity features to Kademlia, instead of on another DHT, is founded mainly on the simplicity of its design as well as on its resistance to particular DDoS attacks.

In short, Likir's architecture needs the support of a trusted Certification Service (CS) that issues signed identifiers binding the overlay ID of a joining node with a user identifier (an OpenId) and a public RSA key. When a new user joins the services she first generate a RSA key pair. Then she contacts the CS web portal, authenticates with OpenId and thus obtain her signed identifier; this is the *registration* phase. The CS does not represent a single point of failure because it has to be contacted only once by each new node, during the registration.

Certified identifiers allow to customize the node communication protocol so that every overlay interaction is authenticated. Furthermore, a certificate, signed by the owner, is attached to every resource stored in the DHT; since owner's identity and resource's hash are included in the certificate, resource integrity and ownership verifiability are assured. For details on Likir protocol and for an evaluation on its computational and bandwidth overhead sustainability we refer to [10].

The first advantage of Likir over other classical DHTs is security. As widely shown in [10], its identity-based features protect the network from all the most dangerous classes of attacks against the overlay, like the Eclipse attack or Sybil-based attacks [2]. The robustness of the p2p layer is very important to make a decentralized SNS implementation as reliable as server-based solutions.

Second, since identity is embedded at routing layer, Likir offers the opportunity to retrieve resources on a safe identity basis. Specifically, we can define three basic primitives that can be easily implemented on Likir:

$$\text{PUT}(key,\ obj,\ type,\ public,\ ttl) \quad (1)$$

$$\text{GET}(key,\ type,\ userId,\ recent,\ grant) \quad (2)$$

$$\text{BLACKLIST}(userId) \quad (3)$$

PUT and GET are basic insertion and retrieval operations. The first stores the object obj at overlay nodes responsible for key; an arbitrary content $type$ and a resource expiration time must be specified. $public$ is a boolean parameter for access control purposes whose meaning is discussed in Section IV-B. Content deletion can be easily implemented through a slight adaptation of the GET primitive.

The latter looks up the DHT nodes responsible for key and query them for saved objects marked with $key.\ grant$ is the dual of the $public$ parameter and it specifies the permissions of the querier to resource access (see Section IV-B). Three optional *index side filtering* parameters can be set in GET. A queried node can be asked to return only resources of a certain $type$ or belonging to a specific $userId$ (certificates bound to resources allow it in a safe and verifiable way). Furthermore, since different *versions* of the same content can be stored in the DHT (i.e. resources marked with the same key and $type$, inserted by the same owner at different moments), the boolean parameter $recent$ allows to ask only for the last published version.

Finally, the BLACKLIST operation adds the provided $userId$ to a local blacklist; every future session established with the node bound to the identity $userId$ will be aborted (remember that each node interaction is authenticated). This simple primitive is at the basis of the reputation system we define in Section IV-C.

IV. ARCHITECTURE

From a user point of view, a SNS can be seen as a *customizable suite of interoperable, identity-based applications*. Applicative modules, or *widgets*, are software components which are downloaded and mounted on a common platform which provides communication and content management primitives. We assume that every widget, regardless of its specific application logic, needs to fulfill two main tasks: to share data provided by the user with a selected set of her friends and to gather contents from other different widgets owned by other SNS participants. This model is very general because contents can vary from textual messages to complex multimedia objects and data may be exchanged in both synchronous or asynchronous mode.

[2]http://www.peerson.net

[3]http://lnx.0131mmp.com/Safebook_site

A shared storage for data publishing and retrieval is the simplest instrument to realize such service. DHTs are very suitable decentralized implementations of shared storages, because content dissemination and maintenance are transparently managed and high scalability is assured. In Figure 1 we depict the appliance of our identity-aware DHT to this context: custom application suites are deployed on overlay nodes that encapsulate the user identity notion for the purpose of authenticated overlay interactions.

Given this setting, we can define three privacy requirements on user-related information. First is *confidentiality*: contents saved in the distributed environment should be readable only by authorized users. Contents can range from shared files to informations about friendship ties, to feeds generated by applications and so on. Second is *anonymity*: the participation of a user to a particular SNS should not be revealed to unauthorized parties. Third, related with confidentiality, is *authorized disclosure* of information. A widget without a proper authorization should not be able to access data belonging to other widgets of the same application suite. This property assures the protection against *trojan* applications aimed to collect user data; for example, the feeds that are generated by an application should not showed in the context of other applications unless a specific authorization is given.

Such requirements can be satisfied easily in any centralized setting that provides the possibility of a proper user profile privacy configuration. However our aim is to rescue data from the control of a centralized authority; but dealing with the mentioned privacy features becomes harder in a distributed setting.

Accomplishment of privacy properties, together with interaction, integration, and reputation features of the model are discussed in next Sections, which describe the architectural modules that composes our p2p SNS client (Figure 2).

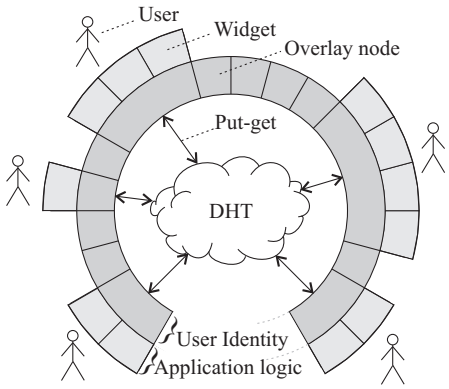

Figure 1. SNS built on identity-based DHT

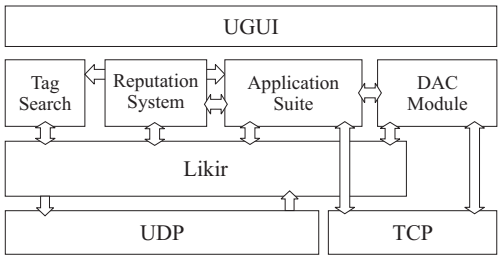

Figure 2. SN client architecture

A. Modular application suite

1) Integration between modules: Data exchange between widgets is mainly achieved through the DHT, using the PUT-GET primitives. This setting differs from a classical DHT context because of the strong identity bound to every user at overlay level: regardless of which application performs a PUT operation, resources stored in the DHT by the same p2p client are marked with a sole user identity.

Such feature is particularly suitable for the integration of different id-based modules. Every widget can retrieve resources belonging to different widgets, but submitted by the same user, by specifying the target $userId$ in the filtering parameters of the GET primitive. Gathering cross-application information related to an identity allows any user-based information *mash-up*, which is a very attractive feature for a SNS. Furthermore, identity and type selection facilities in GET allow to retrieve only the desired resources, thus relieving the application from any filtering tasks. For example, if all GET filtering parameters are set, at most one resource is returned (the most recent content published by a given user, under a given type).

Obviously, the application-specific content insertion rules must be public in order to perform integration between modules. Exactly as in the context of Web 2.0 services, we imagine that every Likir application should provide a simple API listing lookup keys and types for every resource it manages.

It is worth noting that identity-based resource retrieval could be implemented also on classical DHT with a proper resource type system, but naive identity-based classifications, where ownership is a mere label attached to the content, are vulnerable to trivial storage poisoning attacks and thus are useless if application robustness is required. Our approach, conversely, grants the data ownership through certificates.

If the application logic of a widget requires also synchronous communication as well as DHT-mediated asynchronous interaction, then the DHT can be used in a preliminary phase to exchange both peers' network contacts and information needed for a Diffie Hellman's key agreement protocol. Since key agreement is performed in a fully authenticated environment, subsequent messages can then

be sent on an encrypted *and* authenticated channel.

2) Tear down the walled garden: A problem that has arisen with the proliferation of social networking sites is the rigidity with which the information flow between one SNS and another is managed. Very often, SNS are structured like *information silos* that do not allow data to be shared with different OSNs or applications to be reused. Driven by the user demand for a more flexible use of SNSs, the research community has recognized this issue, known as the *"walled garden problem"*, as a priority to be solved in the near future [11].

Several proposals for unifying SNS services have been made (e.g. the OpenId-compliant Global Social Platform [12]); these kind of solutions are tailored for a context where SNSs are implemented in a centralized way.

As well as the data privacy issue, also the walled garden problem can be tackled through our design. Indeed, if we see a SNS as a bunch of related applications, different OSNs can cohabit on the same distributed platform and interact with each other as long as applications' API are published. A single social graph can thus emerge through the integration of modules belonging to different application suites (if the integration is profitable, and, of course, desired). Content's ownership and integrity verifiability makes the integration safe.

In our architecture, we emphasize the possibility of several OSN cohabitation by introducing a Unified Graphical User Interface at the top of the design sketch. Social tools from different OSNs can be mounted on a single "social desktop" for the sake of interaction and cooperation.

B. Resource access control

1) Protocol: Previous distributed OSNs proposals used content encryption in order to grant privacy on data. However, such solution is not suitable in systems where the group membership is highly dynamic; when a member is expelled from a group, the resource owner must share a new key with the remaining members and each resource must be newly encrypted to prevent possible unauthorized access by the former member.

We address this problem by delegating the access control task to overlay index nodes, i.e. those peers that are supposed to keep published resources in their storages. Contents inserted in the DHT through the PUT primitive with the $public$ parameter unset are not accessible to everyone; index nodes return such resources only to those peers that can provide a proper access permission (the $grant$ parameter in GET).

Emission of grant certificates is performed by the Discretionary Access Control Module (DACM). DACM is an applicative daemon which listens for new friendship requests, providing the first point of contact between users. The network address on which the module is listening is inserted (and properly refreshed) in the DHT under a well known key and type.

A friendship request from user A to user B is inserted in the DHT first:

$$\text{PUT}(Req||B, FriendReq, DACReq, true, ttl); \quad (4)$$

The lookup key is the concatenation of a default string with the recipient's userId. We do not define the details on $FriendReq$ object's structure; in principle it is customizable by the sender. In order to solicit its retrieval, the request publication is promptly notified to the DACM by the requesting peer; the DACM checks the DHT at every startup, to collect requests published during its offline period.

After the request is received and evaluated by the user, a grant certificate (expr. 5) is issued and published (expr. 6); B's DACM is then notified as well.

$$Grant_A(B) = \{A||B||regExp||expireTime\}_{Sig_A} \quad (5)$$

$$\text{PUT}(Res||A, Grant_A(B), DACRes, true, ttl); \quad (6)$$

The grant certificate, signed by A, includes the identities of the issuer and of the requester, an expiration time and a regular expression which determines the set of content types whose access is granted to B.

The whole set of regular expression is determined by the applications. When a new widget is installed, it informs the DACM about the set of types it manages. Gathering the types of every application, the DACM is able to display the whole type set to the user. In turn, for each new friendship request, the user chooses the allowed types and a proper regular expression is then produced.

When B tries to retrieve a non-public resource belonging to A, the $Grant_A(B)$ included in the GET request is evaluated by the index nodes and the content is returned only if the grant matches the request. Note that the index nodes are able to check the grant signature because, during the content publishing protocol, the signed identifier of A, which contains her public key, is sent to the index nodes together with the content.

Revocation of access permission is managed through certificate expiration. For this reason, the choice of a convenient certificate life span is crucial. Obviously, the lowest is the life span, the shortest is the time in which a no-more-authorized user can gain access to protected resources. We believe that the expiration time should be customizable by the user, depending on the trust he places on a specific friend. However, since the grant certificates renewal cost is linear with the number of friends (it does not depend on the number of resources), a low (e.g. one week) expiration time should be affordable.

Access control introduces computational overhead due to the RSA signature verification that index nodes have to bear for each GET request received. Since the Likir node interaction protocol requires several RSA signature

verifications and creations (see [10]), we can venture to state that an additional check operation does not significantly affect system performance nor scalability.

2) Privacy properties: DAC protocol provides *confidentiality*, but unauthorized index nodes may access to contents saved in their storages even without a proper grant certificate. In order to prevent unauthorized access by index nodes, data encryption can be applied. The encryption key exchange protocol between friends can be performed directly by the DACM endpoints. Note that when a user authorization is revoked (i.e. her grant is not renewed) there is no need to change the encryption key, because of the DAC mechanism.

It is hard to grant complete *anonimity* when the ownership of any content is verifiable. However, if we suppose that an index node replies with a generic *"content unavailable"* error message to any unauthorized content request, then DHT crawling activities cannot disclose the participation of a user to a specific service. Besides, since Likir does not allow any peer to determine its position on the overlay (overlay IDs are random generated by the CS), a malicious peer cannot acquire an overlay position aimed to intercept contents marked with specific keys and types.

Lastly, *unauthorized disclosure* of information performed by malicious applications can be avoided simply extending the grant certificate distribution policy also to widgets. Untrusted widgets should be equipped only with grant certificates that do not allow them to access other widgets' resources, thus avoiding trojan horse attacks.

C. Reputation management

In social networking environments, establishment of trust bonds between participants is very important for the selection of reliable partners and for the exclusion of malicious participants from social interaction. Often, a social link in a OSN is established only if a trust tie has been previously created in the real world, but in general participants met the first time through the SNS; in this case, a typical instrument to compute the reciprocal trust is reputation.

For these reasons, it is important that SNSs provide reputation management tools to applications. We believe that the interaction with a Reputation System (RS) should be an optional feature rather than a structural SNS component, so we consider the RS like a custom applicative module.

The RS collects feedbacks on the known contacts' behavior directly from the applications and interacts with remote RS instances to spread local information. Local feedback and data received from outside are processed to associate a reputation score to every known identity. When a specific reputation score falls under a given threshold, then the corresponding identity is considered malicious and must be expelled from the network.

For this purpose, the Likir layer provides the BLACKLIST primitive. When an identity is added to the local blacklist, future interactions with the corresponding user are avoided. Malicious users cannot bypass the identity check because overlay communications are authenticated.

Our approach has two main advantages. First, each SNS can choose the RS implementation that best suits the nature of the social network. Any RS that leverages the DHT services to share local feedbacks can well match our design; a suitable example can be found in [13].

Second, the BLACKLIST function is very effective against *whitewashing*. Since the blacklist is managed at overlay level, users banned in the ambit of an application will be rejected also by any other service. Therefore, if the information about misbehaviors is effectively spread across the network, malicious users are excluded from the overlay interactions and restricted into a "dark" network partition. Such feature is advisable because a severe punishment for malicious users represents an effectual deterrent against baleful behaviors.

D. Tag-based search engine

In very populated OSNs, where a huge number of resources is submitted every day, an efficient indexing mechanism for content retrieval is crucial. In file sharing, the most pervasive application on DHTs, lookup keys are usually computed giving in input the words that compose the file title to a hash function. This exact-match lookup approach is not flexible because the name assigned to the resource can be not very informative to people that are interested in its retrieval.

An alternative to the exact key search is the tag-based search, very popular in web OSNs. Users assign arbitrary labels to resources, creating a *folksonomy* used for subsequent content classification and retrieval. Social tagging systems were born in a centralized context, but some distributed solutions have been recently proposed (e.g. [14]). We believe that a tag-search feature is an essential element to make p2p SNSs competitive compared to their server-based counterparts; for this reason we include a general purpose tagging module in our architecture.

The main task of the search engine is to map and update the bipartite tag-resource graph (refer to [15] for details on folksonomies graph structure) on the DHT. Each node in the graph, together with its outgoing edges, is mapped on a lookup key computable from the name of the corresponding resource or tag. A similarity tag-tag graph based on co-occurrences [15] is also mapped to explicitly link together similar tags. The tag search module is able to navigate tag-tag and tag-resource connections, returning sets of labels and contents related to a provided tag. Graph edges are properly updated in order to implement the tagging of a retrieved resource.

Applications interact with the tag search module through a INSERT primitive, which adds a specified resource, together with a list of related tags, in the distributed folksonomy.

Applications can index the resources they publish by inserting a pointer object to the desired resource, together with a set of tags, in the folksonomy. A pointer contains the lookup key and type of the resource, together with a unique application identifier; when a pointer is retrieved through the tag search, the related content can be located in the DHT, retrieved and then passed to the proper module, according to the application identifier.

Such search mechanism offers a global view over the resources in the DHT, no matter what application they belong to, thus promoting even more the opportunity of information mash-up.

V. Conclusions

Distributed Hash Tables have proved to be suitable for social networking application development. Their open and decentralized nature is ideal for privacy preservation and for integration of various social services, thus avoiding the so-called "walled garden problem".

Nevertheless, even if important steps in the definition of DHT-based OSNs have been made, several issues have remained unsolved. First, the robustness of DHT-based application is limited due to overlay network vulnerability to common attacks. Second, scalable and dynamic resource access control represents an issue in unsupervised environments. Lastly, an explicit support for reputation tools serviceable by applications is not well defined.

Our architectural manifesto provides a solution to these problems through the embedding of the identity notion into the overlay level. In addition, a favorable identity-based integration between applicative modules is allowed by the framework. Furthermore, we point out the need for an effective tag-based resource retrieval for a complete SNS design; a tag search engine embedded in the architecture is proposed.

The feasibility of our design is supported with the implementation of some architectural modules[4]. In addition to the id-based DHT, we developed the tag search module and an instant messaging client, which shows how basic synchronous and asynchronous communications can be easily managed with our framework. The DACM module is under construction and will be released soon.

References

[1] S. Buchegger and A. Datta, "A Case for P2P Infrastructure for Social Networks - Opportunities and Challenges," in *WONS'09: 6th International Conference on Wireless On-demand Network Systems and Services*, Snowbird, Utah, USA, 2009.

[2] G. Urdaneta, G. Pierre, and M. van Steen, "A survey of DHT security techniques," *ACM Computing Surveys*, 2009, http://www.globule.org/publi/SDST_acmcs2009.html, to appear.

[3] H. Lundgren, R. Gold, E. Nordström, and M. Wiggberg, "A distributed instant messaging architecture based on the Pastry peer-to-peer routing substrate," in *SNCNW 2003, Swedish National Computer Networking Workshop, Stockholm*, 2003.

[4] P. Antoniadis and B. L. Grand, "Self-organised virtual communities; bridging the gap between web-based communities and P2P systems," *International Journal of Web Based Communities*, vol. 5, no. 2, pp. 179–194, 2009.

[5] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta, "PeerSoN: P2P Social Networking - Early Experiences and Insights," in *SNS'09: 2nd ACM Workshop on Social Network Systems Social Network Systems*, Nürnberg, Germany, 2009.

[6] L. A. Cutillo, R. Molva, and T. Strufe, "Leveraging social links for trust and privacy in networks," in *INet Sec 2009. Open Research Problems in Network Security. Zurich, Switzerland*, 2009.

[7] S. Abbas, J. Pouwelse, D. Epema, and H. Sips, "A gossip-based distributed social networking system," in *WETICE'09: 18th IEEE International Workshops on Enabling Technologies. Groningen, Netherlands*. IEEE Computer Society, June 29 - July 1, 2009, pp. 93–98.

[8] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips, "TRIBLER: a social-based peer-to-peer system," *Concurrency and Computation*, vol. 20, no. 2, pp. 127–138, 2008.

[9] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *IPTPS '02: 1st International Workshop on Peer-to-Peer Systems*, 2002, pp. 53–65.

[10] L. M. Aiello, M. Milanesio, G. Ruffo, and R. Schifanella, "Tempering Kademlia with a robust identity based system," in *P2P'08: 8th International Conference on Peer-to-Peer Computing*, 2008, pp. 30–39.

[11] C. M. A. Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee, "Decentralization: The Future of Online Social Networking," in *W3C Workshop on the Future of Social Networking*, 2009.

[12] M. Mostarda, D. Palmisano, F. Zani, and S. Tripodi, "Towards an OpenID-based solution to the Social Network Interoperability problem," in *W3C Workshop on the Future of Social Networking*, 2009.

[13] M. Srivatsa, L. Xiong, and L. Liu, "TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks," in *WWW '05: 14th international conference on World Wide Web*, 2005, pp. 422–431.

[14] O. Görlitz, S. Sizov, and S. Staab, "Tagster - Tagging-based distributed content sharing," in *ESWC '08: 5th European Semantic Web Conference, Demo Session*, ser. LNCS, vol. 5021. Springer, June 1-5 2008, pp. 807–811.

[15] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme, "Evaluating similarity measures for emergent semantics of social tagging," in *WWW '09: 18th World Wide Web conference*, 2009.

[4] Source code, written in Java, is available at: http://likir.di.unito.it