A Reachability-Based Access Control Model for Online Social Networks*

Talel Abdessalem
Institut Télécom - Télécom ParisTech
46 rue Barrault, 75634 Paris, France
Talel.Abdessalem@telecom-paristech.fr

Imen Ben Dhia
Institut Télécom - Télécom ParisTech
46 rue Barrault, 75634 Paris, France
Imen.Bendhia@telecom-paristech.fr

ABSTRACT

As a result of the widespread use of social networking sites, millions of individuals can today easily share personal and confidential information with an incredible amount of possibly unknown other users. This raises the need of giving users more control on the distribution of their resources, which may be accessed by a community far wider than they can expect. Our concern is how to specify and enforce access rules in a social network. The solution proposed in this paper relies on connection characteristics between users, in an extended sense that includes indirect connections. It provides a conditional access to shared resources based on reachability constraints, between the owner and the requester of a resource, specified through access rules. Experimental results show the effectiveness of our approach over a real social network dataset.

Categories and Subject Descriptors

H.3.4 [**Social networking**]: Security, integrity, and protection

General Terms

Algorithm, Security

Keywords

Access control, online social networks, privacy policies

1. INTRODUCTION

With the development of Web 2.0 technologies in the last few years, social networks (e.g., Facebook, LinkedIn, Flickr, Twitter, etc.) have become among the most successful services on the Web, exploited by an exponentially increasing number of users. Actually, Facebook now reports over 500 million active users [2] and Twitter has 200 million users [1]. These so-called Online Social Networks (OSNs) are online communities whose main goal is to make available an information space, where each social network participant can publish and share information (e.g., personal data, photos, videos, opinions, contacts, etc.), as well as meet other people for a variety of purposes (e.g., business, entertainment, religion, dating etc.).

The availability of this information obviously raises privacy and confidentiality issues. Users typically do not want to share all of their information with everyone. Consequently, OSNs must provide the right mechanisms in order to give users more control on the distribution of their resources, which may be accessed by a community far wider than they can imagine. For example, many employers search for their candidates on social networking sites before they hire them [16]. In a tight job market, information that people share (e.g., political views, status updates, funny pictures, etc.) might be a deal breaker. The private information that is available on OSNs could endanger the future employment chances of job candidates, even before the chance of having an interview.

Most developed social networks provide only the most basic access control policies, e.g., a user can specify whether a piece of information shall be publicly available, private (no one can see it) or accessible to direct contacts only. This simple access control paradigm has the advantage of being intuitive and easy to implement. However, it is not flexible enough to fit the requirements of all users. It is either too loose since it grants access to all users (i.e, public), or it is too restrictive by limiting too much information sharing (i.e, private). Thus, social networking applications need mechanisms that support high-level policies.

For instance, Facebook allows users to organize their friends into lists and specify whether a specific piece of information should be available to a particular list. However, since the average number of friends of Facebook users is estimated to be 130 [2], the process of categorizing friends into lists turns out to be tedious and time-consuming. Moreover, it may happen that users find themselves forced to create a large number of friend lists due to the fact that their privacy preferences may vary depending on the piece of information to share.

It is thus clear that users should be provided with more flexible mechanisms to control access to their own information. In real world scenarios, interpersonal relationships and perceived reliability (or trust) are natural criteria for privacy management. For instance, one can say *"only my family and my friends can view my birthday photos"*, *"only my children*

*This work has been supported by the French ANR project ISICIL.

DBSocial '11, June 12, Athens, Greece

and their friends can read my notes on The Simpsons" or *"only my reliable neighbors can have access to the details of my next holidays"*. As in the real world, OSN members will have in mind a specific audience for their resources. OSNs should then enable them to specify the audience and enforce their access rules.

We propose in this paper a network-aware access control model for OSNs where access control rules are expressed as reachability constraints. These constraints express an encoding of the type of the path that have to exist between the seeker of a resource and its owner. Thus, each user can specify the target audience for his resources. Our experiments show that access rules enforcement can be done on the fly when seekers try to access some shared resources.

Access control in social networks is a recent problem that has emerged with the growing popularity of online social networks. We can classify previous work into two categories: (i)*machine learning-based* approaches as in [8], which try to automatically configure user privacy settings, based on available explicit access authorizations, and, (ii)*rule based* approaches as in [6], which introduced trust and distance in the social graph as the key criteria for access rules.

Our work is a new approach in the latter category, which generalizes access constraints by taking into account the properties of the users, the indirect connections between these users, and is able to express complex relationships (i.e, sequence of direct relationships of different types). The main idea is the specification of the target audience of each access rule in terms of a reachability constraint, which is expressed as a path expression over the social network graph. Thus, the enforcement of an access rule consists in the evaluation of a path, which can be computed on the fly when the resource is requested by the seeker. The experiments we performed on a real social network dataset confirmed this intuition, and showed that the reference monitor of our access control system can decide in real time whether a seeker is part of the audience of a given resource or not.

The remainder of this paper is organized as follows: Section 2 presents a running example and the requirements of access control models in social networks. In Section 3, we discuss and present our access control model. Section 4 discusses the time complexity of our privacy policy enforcement algorithm, and Section 5 presents some experimental results. In Section 6, we discuss some related work. Finally, Section 7 concludes the paper.

2. PRELIMINARIES

As depicted in Figure 1, a social network is a dynamic structure made of nodes, which are connected to each other through various relations. The nodes and the edges of the graph denote, respectively, the social network users and the relationships that exist between them. Labels describe the relationship type associated to each edge, i.e., *Alice* considers *Bill* her friend, *Colin* considers *David* his friend, and so on. In this paper relationships are not supposed to be symmetric, i.e., if *Alice* considers *Bill* as a friend, that does not mean that *Bill* considers *Alice* a friend too. Thus, we consider directed social network graphs, where each edge have an initial node and a terminal node.

Two types of relationships can be distinguished: *direct* and *indirect* relationships. A direct relationship involves only two nodes: the initial one and the terminal one. However, an indirect relationship has intermediary nodes, and

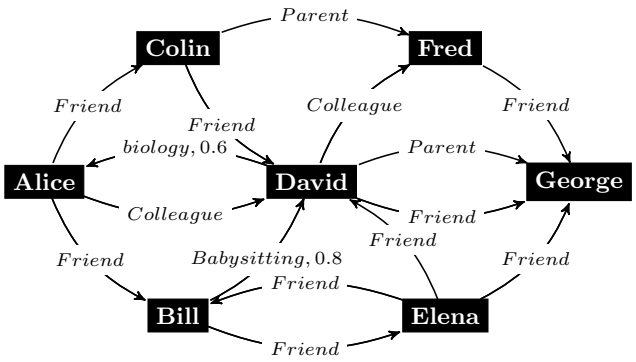

Figure 1: An Online Social Network Subgraph

consists of a finite number of direct relationships. For instance, *Alice* has a direct friend-typed relationship with *Bill*, and an indirect colleague-typed relationship with *Fred*. The *depth* of a given relationship corresponds to the number of direct relationships (of the same label) it is composed of.

In the real world, some interpersonal relationships are based on trust or reliability estimations. This can be denoted as weights on the corresponding relationships in the social network graph. In this case, the relationship type (for instance, babysitting) is called the utility of the trust, and the weight is the value, which is assigned to the trust relationship. As depicted in Figure 1, *Bill* trusts *David* for taking care of children up to 80%, and *David* trusts *Alice* up to 60% in the field of biology. When the trust relationship is direct, it corresponds to an explicit trust (i.e., given by the initial user). When it is an indirect relationship, its value has to be inferred based on available explicit trust values. Different trust propagation approaches are proposed in the literature [12, 10, 5], but trust propagation details are out of the scope of this paper.

The design and implementation of a suitable access control model for online social networks present a number of challenges. We consider that the following requirements are keys to developing such a model:

Dynamicity. In the real world, interpersonal relationships are varied, numerous, and changing over time. Consequently, as a first requirement, an access control model for OSNs should take into account relationships diversity and dynamics.

Suppose that *Bill* is authorized to access *Alice*'s holiday album because she considers him her friend. If *Alice* does not consider *Bill* her friend anymore, he should no longer be able to access her holiday album. In this case, *Alice* does not need to change access rules that she has set for her holiday album, she just has to update her relationship with *Bill* in the OSN.

Precision. Access control should allow targeting the audience of a given resource with the granularity that a user might need. Thus, in an OSN context, access control models should be able to take into account user properties (age, gender, location, status, hobbies, etc.), relationships between users in an extended sense (i.e., not limited to direct relationships), and trust measures when they can be inferred based on user input or their previous interactions.

In our example, *Alice* should be able to share her resources

with her friends, her colleagues, her colleagues' friends, etc. *David* should be able to share his jokes with those who consider him as a friend (*Elena* and *Colin*), and he should be able to extend the audience to their friends (*George* and *Bill*, for *Elena*), and so on. Suppose now that *Elena* is looking for a baby-sitter for her kids, she may want to publish an advertisement and make it accessible to only some trusted baby-sitters. For instance, those she personally trusts for baby-sitting and those trusted by her friends (let us say with a trust level > 0.5). In this case, the advertisement could be accessible to *David*.

Scalability. Access control protocols have to be able to decide on the fly whether to allow or deny an access request sent by a user. This is important in the context of online social networks, where the number of active users can reach several hundreds of millions, and much more relationships between them. The response time in this case is a critical issue.

3. THE ACCESS CONTROL MODEL

Devising an access control model implies the specification of both the access control policy, and the access control enforcement mechanism. The former corresponds to the desired rules according to high level requirements, and the latter denotes the access policy implementation.

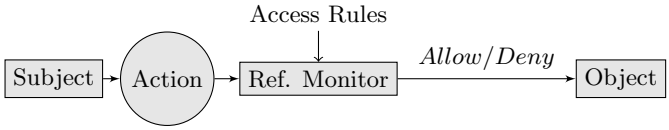

Figure 2: Access control model

As shown in Figure 2, the fundamental components of our access control model are:

- The *Object* is the target resource, to which access may need to be controlled.
- The *Subject*, also called *Principal*, is a user who tries to get access to a particular resource.
- The *Action* is the operation that the *subject* wants to execute over the *object*.
- *AccessRules* specify the access control policy, i.e., user privacy preferences.
- The *ReferenceMonitor* is the component that implements user privacy preferences. It takes as input a request (an action sent by a subject) and a set of access rules according to which it will allow or deny access to the targeted *object*.

Access control policies are presented in section 3.1 and the reference monitor is described in section 3.2.

3.1 Access Control Policy

We propose a reachability-based access control model that enables targeting the audience of the shared resources in a network-aware manner. The intuition behind the design of this model comes from the observation that, in the real world, we generally conceive our privacy preferences based on relationships that bind us to each other. In our model, an OSN is considered as a labeled and oriented graph, where nodes denote subscribed users and links (edges) denote direct relationships between users. User properties (age, gender, etc.) are expressed as attributes of the graph nodes.

Privacy preferences are expressed by a set of access rules, each one is associated to a given object (i.e, a shared resource) and specifies, through a reachability constraint, the profiles of the target audience (i.e., users that are allowed to have access to the considered object). Each reachability constraint is represented by a path expression over the OSN graph.

DEFINITION 1. *Online Social Network (OSN)*
We formally define an OSN as a tuple:

$$(V, E, \Sigma, \phi)$$

where V is a set of nodes denoting the social network members. $E = \{(v_1, v_2) \mid v_1 \in V, v_2 \in V\}$ is a set of edges denoting relationships between users. Σ is a function that assigns each edge $e \in E$ a pair (r, t), where r is a relationship label (e.g., friend, colleague, etc.) and $t \in [0, 1]$ is a trust value explicitly given by the user v_1. A default value (e.g, 0.5) can be used for missing explicit trust values.
$\phi : V \times V \times P \rightarrow [0, 1]$ is a trust propagation function that computes an implicit trust level between any two users v_1 and v_2, based on a path $p \in P$ that links v_1 to v_2 and the available explicit trust level at the intermediary steps of p. P is the set of all the paths that could exist in the social network graph. The product (or the average) of the trust values at each step of p can constitute a simple ϕ function. More sophisticated trust propagation functions can be found in the literature [12, 10, 5]. Since the focus in this paper is not on trust propagation, we can consider ϕ as a black box.

DEFINITION 2. *Access Rule (AR)*
An access rule expresses a set of access conditions that should be satisfied in order to access a given resource. An access rule is a tuple:

$$(rid, ACS)$$

where *rid* is the identifier of the resource and ACS=$\{AC\}$ is a set of access conditions expressing the requirements that should be satisfied by the requester in order to be allowed to access the resource.

DEFINITION 3. *Access Condition (AC)*
An access condition specifies the profiles of authorized users. It is formally defined as a tuple:

$$(o, p, t_{min})$$

where o is the starting node (representing the resource owner), p is a path in the social network graph, and t_{min} is a trust threshold. We consider $p = s_1, s_2, ..., s_n$ as a sequence of ordered steps. Each step s_i is represented by a tuple (r, dir, I, C), where r is a label denoting the relationship, dir specifies the corresponding edge orientation in the graph: $dir = +$ (respectively, $dir = -$) means that the relationship must be outgoing (respectively, incoming). A default value $*$ is used for dir indicating that both incoming and outgoing relationships are authorized. I is the set of authorized depth levels, and $C = c_1, c_2, ..., c_n$ is the set of conditions on user properties.

In order to get access to a given resource, a requester v must have a direct or an indirect relationship with the resource owner o that matches with the specified path p, and

that satisfies the trust threshold: $\phi(o, v, p) \geq t_{min}$.
Let us consider again the example of the OSN depicted in Figure 1. Suppose that *Elena* wants to make her baby-sitting advertisement accessible to her direct friends. In this case, the reachability constraint will be specified through the following path: $Friend^{+}[1]$. The access rule will be $AR_1 = (ad, \{(Elena, Friend^{+}[1], 0)\})$. Note that, the trust threshold is set to 0, which means that trust is not considered in this rule. If *Elena* wants to extend this authorization to her indirect friends (i.e, the friends of her friends) the reachability constraint can be updated as follows: $Friend^{+}[1, 2]$. If she wants to extend it again to the users that consider her as a direct friend, the new access rule will be $AR_1 = (ad, \{(Elena, Friend^{+}[1, 2], 0), (Elena, Friend^{-}[1], 0)\})$. In this case, the access rule consists of two access conditions, the first one is for outgoing friendship relations and the second one is for incoming friendship relations. If *Elena* wants to change the access rule associated to her baby-sitting advertisement and make it accessible to the trustworthy baby-sitters of her friends, the access rule changes to $AR_1 = (ad, \{(Elena, Friend^{+}[1]/babysitting^{+}[1], 0.5)\})$. Finally, the authorization can be limited to the baby-sitters living in Paris by adding an additional condition to the reachability constraint as follows:
$Friend^{+}[1]/babysitting^{+}[1][location = Paris]$.

It is important to note that the proposed model considers authorization access rules only. Thus, we avoid access authorization-denial conflicts.

3.2 Access Control Enforcement

The access control enforcement mechanism is performed by the *reference monitor*, which is a trusted software module that intercepts each access request submitted by a subject to access an object, and, on the basis of the specified access policy, determines whether access should be granted or denied to the requester. The decision module of the reference monitor is detailed in **Algorithm 1**. Suppose that a user

Algorithm 1: Access Control Protocol

Precondition: A requester R wants to get access to an object *res* of O.
Input : A requester R and an object identifier *rid*.
Output : *Allow* or *Deny* access.

Begin

```
1  ARS ← getRules(rid);
2  if ARS= ∅ then
3  |  ARS ← default access rule;
   else
4  foreach AR ∈ ARS do
5  |  foreach AC ∈ AR do                    // AC=(O,p,t_min)
6  |  |  if (checkPath(O,R,p) and
   |  |  trust(O,R,p) ≥ t_min) then
7  |  |  |  return Allow;
   |  |  else
8  |  |  |  Skip to next AR;
9  return Deny;
```

End

R is requesting for a resource *rsc* where *rid* is the resource identifier and O is the node owning such a resource. When R submits his/her access request to access the resource *rsc*, the system retrieves the set of access rules ARS associated to the resource *rsc* (Line 1). If there are no access rules related to the requested resource, then, the system will apply the *default access rule* (Lines 2-3). The *default access rule* is defined by the user and applied whenever there are no access rules associated to the requested resource. It prevents the access control strategy from being too loose (by setting resources having no associated rules to public) or too restrictive (by setting resources having no associated rules to private).

Then, the reference monitor will check if the requester is authorized to access the resource *rsc*, by evaluating the retrieved access rules (Lines 4-9). More precisely, the system evaluates the set of retrieved access rules one by one and stops, either when the requester satisfies one of these rules, or when all the rules were evaluated and the requester satisfies no rule. In the former case, the requester is authorized to access the resource that he asked for (Line 7). In the latter case, the requester is denied access because his profile is not consistent with the target audience (Line 9).

The evaluation of access rules is an iterative process (Lines 4 to 9). For each access rule, the algorithm evaluates the associated access conditions. For each access condition, the $checkPath()$ function evaluates the access path p. It performs its search beginning from the owner node and checks if the requester could be reached via the path p. If there is no node satisfying any of the paths p of the access conditions, then, the current access rule is not satisfied and the system goes directly to the next rule (Line 8). If a satisfying node is found during the search, the $trust()$ function will be called to determine if the found node satisfies the trust condition (i.e,$\phi(O, R, p) \geq t_{min}$). The $checkPath()$ function is performed using a breadth first search algorithm (*BFS-algorithm*).

The trust computation process between O and R is done at the same time, when the graph is explored. Our model focuses on access control given such a trust computing function. In our implementation, we consider that the trust value between two nodes $v1$ and $v2$ is the average of all trust values of a path that links these two nodes. We plan to consider a more elaborated trust computation function in future work.

4. COMPLEXITY ANALYSIS

In this section, we focus on estimating the time complexity of the proposed algorithm for access control rule enforcement, in order to measure its efficiency. Let S be the sum of all maximum depth levels among those specified in each set of authorized depth levels $s.I$ (see Definition 3):

$$S = \sum_{s \in p} s.I_{max}$$

By assuming that, in real world scenarios, S is always less than the social graph diameter, the worst case occurs when there are no constraints neither on edge labels ($s.r = *$), nor on depth levels ($s.I = *$). More precisely, in such a case, we are called to discover all social graph nodes. Since we use a breadth-first search algorithm, exploring the network graph requires ($\mid V \mid + \mid E \mid$) time complexity, where $\mid V \mid$ and $\mid E \mid$ denote, respectively, the OSN nodes and edges. After evaluating an access step, the algorithm will consider a new list of nodes satisfying the previous step s. In the worst case, this list will contain $\mid V \mid$ nodes. Consequently, the social

graph will be browsed $\mid V \mid$ times, and the time complexity required to evaluate an access step s is of the order of:

$$\mid V \mid \times(\mid V \mid + \mid E \mid) \quad (1)$$

We can say that the time complexity required to evaluate an access path p, i.e, an access condition AC, is of the order of:

$$Nb_s \times \mid V \mid \times(\mid V \mid + \mid E \mid) \quad (2)$$

Nb_s is the maximum number of access steps that an access path of a given access condition could contain. In the worst case, the evaluation of an access condition is iterated $(Nb_{AC} \times Nb_{AR})$ times, where Nb_{AC} is the maximum number of access conditions that an access rule could contain and Nb_{AR} is the maximum number of access rules that may be associated to a resource. Thus, the required time complexity to evaluate the proposed algorithm is of the order of:

$$Nb_{AR} \times Nb_{AC} \times [Nb_s \times \mid V \mid \times(\mid V \mid + \mid E \mid)] \quad (3)$$

Then, we can conclude that the problem for which we implemented Algorithm 1 can be solved in polynomial time depending on the number of access rules, access conditions, access steps and the social graph size. Thus, the above equation still hold when we assume that S (detailed in the beginning of this paragraph) is always less than the social graph diameter. Otherwise, the time complexity of Algorithm 1 becomes of the order of:

$$Nb_{AR} \times Nb_{AC} \times [Nb_s \times I \times \mid V \mid \times(\mid V \mid + \mid E \mid)] \quad (4)$$

I is the maximum depth level value that could be specified in an access path p. More clearly, in this case, I could be greater than the social graph diameter.

Time costs given in the previous formulas are determined in the worst case. When users specify their access control conditions, constraints they set on the type, direction and relationship depth levels, considerably reduce the part of the graph to be browsed at each step s. Consequently, the cost of evaluating s is far lower then the worst case cost (i.e, $|V| \times (|V| + |E|)$). However, the number of access rules Nb_{AR}, the number of access conditions Nb_{AC} and the number of steps Nb_s have a clear impact on the response time of our algorithm.

5. EXPERIMENTS

In this section, we perform experimental studies on a real social graph dataset to evaluate the performance of the proposed algorithm in terms of response time to the access requests. We chose to implement the social graph in *Neo4j 1.2* [3], which is becoming one of the foremost graph database systems. Instead of static and rigid tables, rows and columns, it manipulates a flexible graph network consisting of nodes, relationships and properties. Developers describe *Neo4j* as a system that can handle graphs of several billions of nodes on a single machine, and can scale up with a cache sharding technique on multiple machines. Its high-speed traversal framework is able to traverse one million nodes per second [4]. We conducted our testing on a PC with a 2.34 GHz Intel processor, and 2 GB memory running Windows 7.

We performed our studies on a sample of 984K unique users that represents the *groundtruth* of the Facebook OSN, i.e., a truly uniform sample of Facebook anonymized user IDs [9]. This Facebook dataset provides information about relations between users. It also provides the total number of friends each user has, his privacy settings, and his network membership. We associated access rules to randomly selected user information. These rules are stored in *Neo4j* as properties of the OSN graph nodes. As reported in Table 1, we have considered subgraphs consisting of a number of nodes ranging from 8,787 to 984,830. We measured the response time of access requests on different-sized graphs. For each graph, we tested a large number of access rules and varied the depth of access condition paths from 1 to the graph diameter. We fixed the owner nodes and the requester nodes in a targeted manner, i.e. two nodes connected by a path length corresponding to the desired path depth, and we measured for each test the average response time.

	|V|	|E|	Diameter
subgraph 1	8787	8733	7
subgraph 2	88971	88268	8
subgraph 3	310299	309083	31
subgraph 4	984830	92704874	35

Table 1: Facebook Dataset Samples

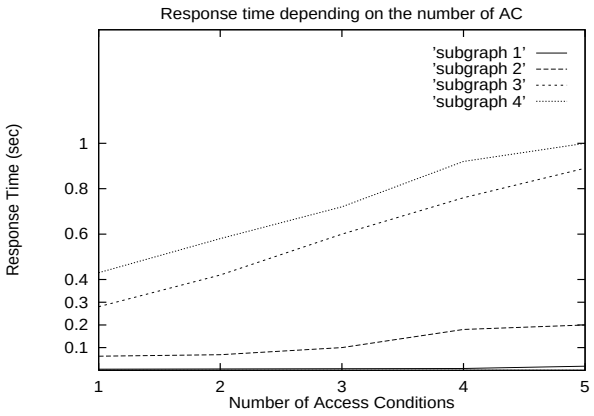

Figure 3: Experimental results on the reference monitor performance with a number of access rules varying from1 to 5

According to the obtained results (see Figure 3), the response time of the reference monitor depends on the number of nodes in the OSN graph: it ranges from 0.001 sec for a graph consisting of 8,787 nodes to 1 sec for a graph consisting of 984,830 nodes. We recall that the size is not referred to the order of the whole social network graph, rather, given a relationship type, it represents the number of nodes having at least a relationship of such type. When the path depth increases, the time required to traverse the social graph, i.e., to get the query result, increases. As shown in our complexity study, in the previous section, the response time depends on the number of access rules to evaluate and the depth of the access condition paths. This was confirmed by our experiments. Experimental results have shown that the social graph nodes number, i.e. number of users in the OSN, also influences the reference monitor performance.

6. RELATED WORK

Access control in social networks is a new research area. It has emerged with the growing popularity of OSNs, which have become an important part of our daily digital life. We can classify previous work into two main categories: machine learning-based approaches and rule-based approaches.

In the first category, we can find papers that have proposed automatic extraction of communities from OSNs as a way to simplify privacy preferences specification. Danezis [7] proposed to classify users contacts into non-overlapping lists, so that contacts of the same list can have only access to information that is shared by their list members. Fang and LeFevre [8] proposed a privacy wizard that considers user explicit privacy preferences as well as automatically extracted communities to build a privacy preference model. This privacy preference model can be automatically applied and adapted whenever the social network graph evolves. Additionally, Shehab et al. [14] proposed a supervised learning mechanism that generates access control policies based on user provided policy settings example, in a collaborative way.

In the second category, and more related to our work, is the rule-based access control model proposed by Carminati et al. [6]. This work introduced trust and distance in the social graph as key criteria for access preferences. The target of an access authorization is specified as a sub-graph based on one simple relationship (friendship, for instance), having in its center the owner of the resource with a fixed radius (i.e, the maximum distance between the beneficiary of the authorization and the owner of the resource). Squicciarini et al. [15] considered an additional problem: co-ownership. They proposed an automated collective privacy management solution where data may have multiple owners, and where owners might have different and possibly contradictory privacy preferences. This model uses a game-theoretical algorithm to control access to resources that are owned by more that one OSN member.

Some other research work has relied on cryptography techniques to protect user information and interactions on the online social networks. For instance, the NOYB model [11] encrypts personal information using a pseudo-random substitution technique which replaces a personal information with a pseudo-randomly selected information from a public dictionary. Another approach, called FlyByNight [13], presents a Facebook application that stores sensitive data in an encrypted form. However, these two approaches does not support selective access control.

7. CONCLUSION

In this paper, we presented a network-aware access control model for online social networks that enables a fine-grained description of privacy policies. User privacy preferences are specified in terms of reachability constraints, combined with user properties and trust considerations. Reachability constraints are expressed as paths in the social network graph. Thus, an access rule enforcement consists in the evaluation of a path, which can be made on the fly when the resource is requested by a seeker. The experiments we performed on an anonymized sample Facebook dataset confirmed this intuition, and showed that the reference monitor of our access control system can decide on the fly whether a seeker is part of the audience of a given resource or not.

8. ACKNOWLEDGMENTS

The authors would like to thank Pierre Senellart for his help in devising the access control protocol, and his comments on earlier versions of the paper.

9. REFERENCES

[1] http://en.wikipedia.org/wiki/twitter.
[2] http://www.facebook.com/press/info.php?statistics.
[3] http://www.neo4j.org/.
[4] http://www.slideshare.net/thobe/nosqleu-graph-databases-and-neo4j.
[5] T. Abdessalem, B. Cautis, and A. Souihli. Trust management in social networks. Technical report, Telecom ParisTech, 2009.
[6] B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In *OTM Workshops*, 2006.
[7] G. Danezis. Inferring privacy policies for social networking services. AISec, 2009.
[8] L. Fang and K. LeFevre. Privacy wizards for social networking sites. WWW, 2010.
[9] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Proceedings of IEEE INFOCOM*, 2010.
[10] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. WWW, 2004.
[11] S. Guha, K. Tang, and P. Francis. Noyb: privacy in online social networks. WOSP, 2008.
[12] H. Liu, E.-P. Lim, H. W. Lauw, M.-T. Le, A. Sun, J. Srivastava, and Y. A. Kim. Predicting trusts among users of online communities: an epinions case study. In *EC*, 2008.
[13] M. Lucas and N. Borisov. Flybynight: mitigating the privacy risks of social networking. SOUPS, 2009.
[14] M. Shehab, G. Cheek, H. Touati, A. C. Squicciarini, and P.-C. Cheng. Learning based access control in online social networks. WWW, 2010.
[15] A. C. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. In *WWW*, 2009.
[16] J. Wortham. More employers use social networks to check out applicants. *The New York Times*, 2009.