Regularity Based Decentralized Social Networks

Zhe Wang(✉) and Naftaly H. Minsky

Rutgers University, New Brunswick, USA
{zhewang,minsky}@cs.rutgers.edu

Abstract. Centralized online social networks (OSNs) have drawbacks, chief among which are the risks posed to the security and privacy of the information maintained by them; and the loss of control over the information contributed by their members. The attempts to create decentralized OSNs (DOSNs) enable each member of an OSN keeps its own data under its control, instead of surrendering it to a central place; providing its own access-control policy. However, they are unable to subject the membership of a DOSN, and the interaction between its members, to any global policy. We adopt the decentralization, complementing it with a means for scalably specifying and enforcing regularities over the membership of a community, and over the interaction between its members.

1 Introduction

An *online social network* (OSN) is a *community* of people that interact with each other, which operates subject to some *laws* that regulate the membership and the manner in which its members interact with each other. Such laws are easy to implement in traditional OSNs (Facebook, etc.). Because they mediate all interactions between their members, and maintain the information supplied by the members of the community.

However, this convenient way of implementing OSNs has several drawbacks. Chief among them are the risks posed to the security and privacy of the information maintained by them and the loss of control over the information contributed by individual members. Security may not be of much concern to the millions of users of Facebook or Twitter. But they are, or should be, of serious concern to other types of OSNs, that exchanges more sensitive information—such as medical and financial information; or internal information of an enterprise.

There are several attempts to create decentralized OSNs (DOSNs); such as LotusNet [1], Safebook [2], PeerSoN [3]. The essential idea of these attempts is that each member should keep its own data under its control, instead of store it to a central place, providing its own access-control policies. However, it's not sufficient as they cannot establish *regularities* over the membership, and the interaction between its members. A set of enforced global law is essential for an OSN, and makes it into a social community.

We designed a model of OSNs with ways to scalably specify and enforce regularities over the membership and interaction between the distributed members.

J. Lopez et al. (Eds.): CRiSIS 2014, LNCS 8924, pp. 255–262, 2015.
DOI: 10.1007/978-3-319-17127-2_19

The rest of this paper is organized as follows. Section 2 introduces an example of OSNs for which security is critical and would thus benefit from decentralization. Section 3 introduces examples of law that are essential for an OSN—particularly for the types of OSNs for which security tends to be critical, but cannot be established under DOSN. Section 4 introduces our model of decentralized OSN. Section 5 is an implemented case study that demonstrates how this model can be used for a concrete application. And we conclude in Sect. 6.

2 OSN in an Enterprise—Case Study

Consider a large and geographically distributed enterprise that provides an OSN for its employees, which distinguishes between groups of employees. The groups may overlap, as an employee could belong to several groups. Let the members have a profile, which is a set of attributes visible to the whole OSN and can be indexed and searched. The members can publish *posts* and build *following* relationships with each other.

There are two types of security needed. First, the information exchanged between the employees can carry sensitive information about the business of company. It is important for this information not to be exposed to the outside. Second, there is need of preventing information exchanged within a certain group from being accessible to anybody else, or other groups.

3 Some of Regularities Imposed over an OSN

We show here the type of *regularities* that an OSN may need to establish. All the regularities discussed here can be easily achieved in centralized OSNs, but cannot be established under the DOSNs' architecture.

3.1 Global Access Control (AC) Policies

DOSNs enable each member to apply its own AC policy to its own data, which are maintained in its own database. The problem is that, unlike in Facebook or Twitter, a member of an enterprise may not have the complete authority over the data it maintains. It could really belong to the enterprise, which thus has the ultimate authority about how they should be distributed. The enterprise may relegate to individual members the right to apply their own AC policies, *provided* that these policies conform to the global policy of an enterprise.

3.2 Associative Sending via Gossip Protocol

By *associative sending*, we mean the ability to send a message, to a subset of the members of the OSN, having constraint on the profile of receivers. We will describe below how both of these capabilities can be provided via *gossip protocol*.

Now a member wants to send a message to all members whose profile satisfies a condition. It starts gossiping, by sending the query to its acquaintances.

Any member that gets the query continues gossiping it; and one should be able to read the message only if it is in the target of this associative sending.

Unfortunately, this mechanism cannot be implemented under DOSN without seriously undermining the security and privacy of members. For example: (1) the basic protocol of gossip could be violated via some members of DOSN. They withhold the message instead of forwarding it; they may violate the frequency or forwarding limit to overwhelm the network; and they may even manipulate the content or source of the message. (2) when a message is sent via gossip, one needs to distinguish between the targets of the message and its carriers. A carrier gets the message and is supposed to gossip it to others, but it is not itself in the target of this message, so it should not read this message. But since it got this message there is no way under DOSN to prevent it from reading the message, and thus learning not only the profile of sender, but also the kind of members that the sender wants to communicate with, judging from the search condition. This is a massive violation of privacy, because every member would see most messages issued by other members.

3.3 Profile Search

Although all published versions of DOSN provide *identity search*, none of them supports *profile search*, despite its obvious importance in OSNs. This is, in part, due their concern about the negative effect that such search would have on privacy [4]. Indeed, simpler and most efficient way to provide for such a search is by creating a central database of the profiles of all members. But this would seriously undermine the privacy of the members of the OSN.

3.4 Membership Control

Control over membership is critical to many OSNs. Such control may have several complementary aspects. First, one may require that to be a member of an OSN, one needs to authenticate itself via a certificate. One may think that this policy can be established under the DOSN architecture by having every member of the DOSN require everyone to authenticate itself in a specified manner. But DOSN has no way for ensuring that all its members behave in this way. Second, one may require that to be a member of an OSN, one needs to garner the support of several current members of it. Third, it is often important to establish some procedure for removing members from a given OSN. This can be done in many ways. For example, consider an OSN that has a member that plays the role of a *manager*. Now, let the manager be given the power to remove any current member of the OSN, simply by sending it a message *remove*. Then it should lose its ability to interact with other members of the OSN.

3.5 Constraints on What Members Can Do

Sometimes we want to impose constraints on what members can do. Constraints may depend on the profile of members, or on the history of their interaction with others. As shown before, only a member that plays the role of manager can send the *remove* message to others. And any member that gets such a message must cease all communication with others. The type of messages that members are allowed to send, or the type of posts that they are allowed to issue, depend on their roles in the OSN.

4 A Model of Decentralized OSN—OSC

We introduce here a model of decentralized OSNs that differs from the current approach employed under the DOSN architecture, in that it enables the enforcement of law over it. We call a specific OSN under this model an *online social community* (OSC). The model employs our previous work—the Law-Governed Interaction (LGI) middleware. LGI is a middleware that can govern the interaction (via message exchange) between distributed *actors*, by enforcing an explicitly specified law about such interaction. A detailed presentation of LGI, and a tutorial of it, can be found in its manual [5]—which describes the release of an experimental implementation of the main parts of LGI.

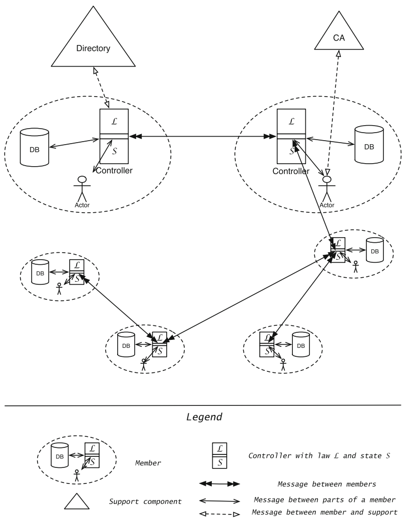
Directory
CA
DB
$\mathcal{L}$
S
Controller
Actor
Legend
Member
Controller with law $\mathcal{L}$ and state S
Message between members
Support component
Message between parts of a member
Message between member and support

Fig. 1. The Anatomy of an OSC Community

Now, a community C under the OSC model is broadly defined as a 4-tuple $\langle M, \mathcal{L}, T, S\rangle$, where M is the set of members; $\mathcal{L}$ is the set of law that governs this community; T is a set of generic LGI controllers that serve as the middleware that enforces law $\mathcal{L}$; and S is a set of components that support the operations of C, and is specific to it—this set may be empty.

4.1 The Anatomy of a Community Under OSC

We describe here the anatomy of a community C under this model by elaborating on its various components, and on the relations between them. This anatomy is depicted schematically in Fig. 1.

The Set M of Members. An individual member m of a community C is a triple $\langle user, mediator, database\rangle$, where *user* is usually a human, operating via some computer; *mediator* is an LGI-controller that mediates all interactions between this member and the rest of the community—subject to law $\mathcal{L}C$ (which we denote by $\mathcal{L}_C$); and *database*, which is an optional part of the member, maintains information associated with this member.

The Law $\mathcal{L}_C$ of Community. It is the law that endows an OSC-community with its overall structure and behavior. And the fact that the law can be any well formed LGI law endows this model with great deal generality regarding the law that can be enforced over a community.

The set T of LGI Controllers. Every user can create its own controller, using the LGI middleware. But if malicious corruption of controllers by their users is of concern, then it is better for them to adopt controllers maintained by a trusted *controller service* (CoS). In particular, the CoS may be managed by the organization in the context of which the community is to operate. Alternatively, the CoS may be managed by a reputed and trusted organization. For more about the security and trustworthiness of controllers, please see [5].

The Support S. An OSC-community may require services of various components that are not themselves members of this community. For example: (a) a certification authority (CA) used for the authentication the various members of the community; (b) a *naming service* that provides unique names of community members; (c) an index service for searching. It is worth pointing out that this set of support components may be empty for some communities.

4.2 The Launching of an OSC-Community

A specific OSC-community, C is launched by constructing its *foundation*—described below—and then having individual members join it. The construction of the foundation of a community C consists of the following steps: (a) defining law $\mathcal{L}_C$ under which this community is to operate; (b) implementing the required support components; and (c) selecting, or constructing, a *controller-service* (CoS) for the use of this community.

Once the foundation of C is constructed, anybody can attempt to join it as a member, via adopting an LGI-controller, and loading law $\mathcal{L}_C$ into it. It should be pointed out that such an attempt to join a given community C may fail, if the conditions for joining imposed by law $\mathcal{L}_C$ are not satisfied.

4.3 The Operation of a Community

Consider a member x sending a message m to another member y. The message first arrives at the controller of x, that operates under law $\mathcal{L}_C$. These controllers would then carry out the ruling of law $\mathcal{L}_C$, which can mandate the execution of any number of the following kind of actions: (a) change its own state in some way; (b) send the message m, or some other message, to the controller of the original target of y; and (c) send some other messages to the controllers of some other members, or to some of the support components of the community. Among other things, this means that members of a community interact with each other via their controllers, and the controllers communicate with each other.

The ruling of a law for a given event that occurs at a controller depends on the state of this controller, which may be different for different members. This difference can come from the role of the user in the organization. Or the state may change dynamically in response to some interactive activity of the community. For example, the manager of the community may be allowed by the law of community to transfer its managerial baton to some other member, which would then be able to send *revoke* messages. In other words, *the members of a community C may not be equal under its law $\mathcal{L}_C$.*

5 Implementation of OSN in an Enterprise

In this section, we describe the implementation of the OSC community, introduced in Sect. 2. It has been implemented in the scale of two hundred users as a proof of concept. The law $\mathcal{C}$ of the Community is used for regulating every aspects of the operations and behaviors of the community. We split it into several parts according to their functionalities. Due to lack of space, we only discuss the detailed law of some functionalities of the communities.

5.1 Member Profile and Membership Control

To join the community, a member needs to adopt a controller under law $\mathcal{C}$. Rule $\mathcal{R}1$ allows a user to join the community by presenting a certificate to prove that it is an employee. Once certificate is verified by the controller, the set of attributes in its profile will be inserted into the user's control state. Rule $\mathcal{R}2$ allows the user to join the group t_i by providing a group certificate (Fig. 2).

```
R1. UPON adopted(X,cert(issuer(ca),subj(X),attr(A))) :-
              do(+A).
R2. UPON certified(X,cert(issuer(ca),subj(X),attr(ti))) :-
              do(+ti); do(+filter(group(ti))).
R3. UPON sent(X,addProfile(Attribute(Value)),X) :-
              if ( ¬ (Attribute in controlledAttributes) ) then do(+Attribute(Value)).
R4. UPON sent(X,updateProfile(Attribute(Value)),X) :-
              if ( ¬ (Attribute in controlledAttributes) ) then do(-Attribute); do(+Attribute(Value)).
R5. UPON sent(X,addFilter(Attribute(Value)),X) :-
              do(+filter(Attribute(value))).
R6. UPON sent(X,#revoke#,Y) :-
              if(role(manager)@CS) then do(Forward);
              else do(Deliver(X,notAllow,X)).
R7. UPON arrived(X,#revoke#,Y) :-
              update(certificateBlacklist); inform(certificateBlacklist); do(Quit).
```

Fig. 2. Law $\mathcal{C}$: Member's Profile and Membership Control

User can directly add attributes into its profile via Rule $\mathcal{R}3$ and update them via Rule $\mathcal{R}4$. Rule $\mathcal{R}5$ shows how a user sets up its subscription filter. When user adds the filter content into control state. Its controller will only allow the members who have the required attributes to subscribe to it. The following operations will be described in Sect. 5.2.

Finally, rule $\mathcal{R}6$ shows that only the manager role can remove a member from the community. Non-managers are not allowed to use the type *revoke* when sending messages. When the *revoke* message arrives at the member's controller, according to Rule $\mathcal{R}7$, the controller will directly terminate the connection to the actor. The member has no way to control or avoid that.

```
R8.  UPON sent(X,publish(P),X) :- group(t_i)@CS
               if (typeof(P) == #management# and ¬ role(manager)@CS) then return;
               updateProfile(lastTenPosts(P)); updateDB(P);
               if(subList[group(t_i)] = []) then return;
               else forEach(subscriber in subList[group(t_i)])
               do(Forward(X,P,subscriber)).
R9.  UPON arrived(X,P,Y) :-
               do(Deliver); do(inform(X,P,Y)).
R10.
     UPON sent(X,requestSubscribe(profile),Y) :-
               do(Forward).
R11.
     UPON arrived(X,requestSubscribe(profile),Y) :- group(t_i)@CS
               if(filter(Attribute(Value))@CS and Attribute(Value)@profile) then do(Forward(Y,subscribeNotAllowed,X));
               else do(updateSublist[group(t_i)]); do(Forward(Y,subscribeAllowed,X)).
```

Fig. 3. Law $\mathcal{C}$: Communication

5.2 Communication

Members can publish *posts* and build subscription relationships with each other. The control over communication has two complementary parts: global control and local control. The global control is imposed on every member of the community, but can be sensitive to the state of members, while the local control is discretionary to each member (Fig. 3).

The global control is imposed on both publishing and subscription. The control over publishing is on what types of posts a member can publish. For example, only the managerial staff can publish posts with type *management*. The control on the subscription regulates who can subscribe to whom, and to which types of posts. An example is that only the members from a same group can talk to each other. Sometimes, a member does not want to be subscribed by certain members, it can block the subscription requests from them. That's what we call *Local Access Control*. To achieve this, a member adds a filter *filter(X)* in its profile, then its cannot be subscribed by the member who has attribute X in profile.

In Rule $\mathcal{R}8$, when the user wants to publish a post to its subscribers, the controller will read local subscriber list and push the post to each of them. When the subscriber receives the post or message, according to the Rule $\mathcal{R}9$, controller will show the post to the user.

By Rule $\mathcal{R}10$, any user can send a subscription request to any user. The controller will attach its profile to the request. In Rule $\mathcal{R}11$, when the request arrives at the user, the controller will check whether there is an access control filter in its control state. If not, it will add the request user to the subscriber list. If there are filters, it will examine whether this user satisfies by checking the required attributes of the profile. If the requester satisfies, the controller will add it to the subscriber list and send back the result to the request user.

5.3 Search

The method we are using for search is gossip protocol. Rule $\mathcal{R}12$ initiates the search request. Rule $\mathcal{R}13$ describe the whole forward procedure of gossip search. When controller receives a search query, it first check whether this user satisfies the search. If it is, it will send back the hit message. Then the controller will check whether the Time To Live is reduced to zero. When it is not, it will send the

```
R12.
    UPON sent(X,search(M),Y) :-
            if(X=Y) then do(Forward).
R13.
    UPON arrived(X,search(M,C,TTL),Y) :-
            if(M@CS) then do(Forward(Y,hit,X))
            if(TTL > 0) then if(subList=[]) then return
            else if(fanout < subList.length) then subSubList = randomPick(subList,fanout);
            do(Forward(Y,search(M,C,TTL-1),subSubList))
            else do(Forward(Y,search(M,C,TTL-1),subList))
            if(C@CS) then do(Deliver).
```

Fig. 4. Law $\mathcal{C}$: Search

search query to its subscriber. If the user is qualified to see the search message, the controller will display the message (Fig. 4).

A main feature of our search method is the enforcement of the law can make sure the user on the search topology will not see the search query unless it is qualified for that search. Therefore, the sender will not need worry about the leak of the secret, while making the best use of the whole network.

6 Conclusion

This paper addresses the risks to privacy and security posed by centralized online social networks (OSNs). These risks, which are the consequence of centralization, should be of serious concerns to many OSNs. Several recent attempts have been made to decentralize OSNs, by letting each member keep maintaining its own data. But this DOSN approach to decentralization is not able to establish any kind of regularity over the social network, which is necessary for both real life social community, as well as for OSNs. We have introduced a decentralized architecture of OSNs, called *online social community*, which is able to establish law concerning both the membership of OSC and the manner in which its members interact. The preliminary testing and experiments of our implementation show that our method is feasible and promising.

References

1. Aiello, L.M., Ruffo, G.: LotusNet: Tunable privacy for distributed online social network services. Comput. Commun. **35**(1), 75–88 (2010)
2. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: Feasibility of transitive cooperation for privacy on a decentralized social network. In: WOWMOM, 1–6. IEEE (2009)
3. Bodriagov, O., Buchegger, S.: P2p social networks with broadcast encryption protected privacy, QC 20120126 (2011)
4. Datta, A., Buchegger, S., Vu, L.H., Strufe, T., Rzadca, K.: Decentralized online social networks. In: Furht, B. (ed.) Handb. Soc.Netw. Technol., pp. 349–378. Springer, NewYork (2010)
5. Minsky, N.H.: Law Governed Interaction (LGI): A Distributed Coordination and Control Mechanism (An Introduction, and a Reference Manual), February 2006. http://www.moses.rutgers.edu/