Multiparty Access Control for Online Social Networks: Model and Mechanisms

Hongxin Hu, Gail-Joon Ahn, Senior Member, IEEE, and Jan Jorgensen

Abstract—Online social networks (OSNs) have experienced tremendous growth in recent years and become a de facto portal for hundreds of millions of Internet users. These OSNs offer attractive means for digital social interactions and information sharing, but also raise a number of security and privacy issues. While OSNs allow users to restrict access to shared data, they currently do not provide any mechanism to enforce privacy concerns over data associated with multiple users. To this end, we propose an approach to enable the protection of shared data associated with multiple users in OSNs. We formulate an access control model to capture the essence of multiparty authorization requirements, along with a multiparty policy specification scheme and a policy enforcement mechanism. Besides, we present a logical representation of our access control model which allows us to leverage the features of existing logic solvers to perform various analysis tasks on our model. We also discuss a proof-of-concept prototype of our approach as part of an application in Facebook and provide usability study and system evaluation of our method.

Index Terms—Social network, multiparty access control, security model, policy specification and management.

1 INTRODUCTION

Online social networks (OSNs) such as Facebook, Google+, and Twitter are inherently designed to enable people to share personal and public information and make social connections with friends, coworkers, colleagues, family and even with strangers. In recent years, we have seen unprecedented growth in the application of OSNs. For example, Facebook, one of representative social network sites, claims that it has more than 800 million active users and over 30 billion pieces of content (web links, news stories, blog posts, notes, photo albums, etc.) shared each month [3]. To protect user data, access control has become a critical feature of OSNs [2], [4].

A typical OSN provides each user with a virtual space containing profile information, a list of the user’s friends, and web pages, such as wall in Facebook, where users and friends can post content and leave messages. A user profile usually includes information with respect to the user’s birthday, gender, interests, education and work history, and contact information. In addition, users can not only upload a content into their own or others’ spaces but also tag other users who appear in the content. Each tag is an explicit reference that links to a user’s space. For the protection of user data, current OSNs indirectly require users to be system and policy administrators for regulating their data, where users can restrict data sharing to a specific set of trusted users. OSNs often use user relationship and group membership to distinguish between trusted and untrusted users. For example, in Facebook, users can allow friends, friends of friends, groups or public to access their data, depending on their personal authorization and privacy requirements.

Although OSNs currently provide simple access control mechanisms allowing users to govern access to information contained in their own spaces, users, unfortunately, have no control over data residing outside their spaces. For instance, if a user posts a comment in a friend’s space, s/he cannot specify which users can view the comment. In another case, when a user uploads a photo and tags friends who appear in the photo, the tagged friends cannot restrict who can see this photo, even though the tagged friends may have different privacy concerns about the photo. To address such a critical issue, preliminary protection mechanisms have been offered by existing OSNs. For example, Facebook allows tagged users to remove the tags linked to their profiles or report violations asking Facebook managers to remove the contents that they do not want to share with the public. However, these simple protection mechanisms suffer from several limitations. On one hand, removing a tag from a photo can only prevent other members from seeing a user’s profile by means of the association link, but the user’s image is still contained in the photo. Since original access control policies cannot be changed, the user’s image continues to be revealed to all authorized users. On the other hand, reporting to OSNs only allows us to either keep or delete the content. Such a binary decision from OSN managers is either too loose or too restrictive, relying on the OSN’s administration and requiring several people to report their request on the same content. Hence, it is essential to develop an effective and flexible access control mechanism for OSNs, accommodating the special authorization requirements coming from multiple associated users for managing the shared data collaboratively.

In this paper, we pursue a systematic solution to facilitate collaborative management of shared data in OSNs. We begin by examining how the lack of multiparty access control for data sharing in OSNs can undermine the protection of user data. Some typical data sharing patterns...
with respect to multiparty authorization in OSNs are also identified. Based on these sharing patterns, a multiparty access control (MPAC) model is formulated to capture the core features of multiparty authorization requirements which have not been accommodated so far by existing access control systems and models for OSNs (e.g., [10], [11], [17], [18], [23]). Our model also contains a multiparty policy specification scheme. Meanwhile, since conflicts are inevitable in multiparty authorization enforcement, a voting mechanism is further provided to deal with authorization and privacy conflicts in our model.

Another compelling feature of our solution is the support of analysis on multiparty access control model and systems. The correctness of implementation of an access control model is based on the premise that the access control model is valid. Moreover, while the use of multiparty access control mechanism can greatly enhance the flexibility for regulating data sharing in OSNs, it may potentially reduce the certainty of system authorization consequences due to the reason that authorization and privacy conflicts need to be resolved elegantly. Assessing the implications of access control mechanisms traditionally relies on the security analysis technique, which has been applied in several domains (e.g., operating systems [20], trust management [25], and role-based access control [26]). In our approach, we additionally introduce a method to represent and reason about our model in a logic program. In addition, we provide a prototype implementation of our authorization mechanism in the context of Facebook. Our experimental results demonstrate the feasibility and usability of our approach.

The rest of the paper is organized as follows. In Section 2, we present multiparty authorization requirements and access control patterns for OSNs. We articulate our proposed MPAC model, including multiparty authorization specification and multiparty policy evaluation in Section 3. Section 4 addresses the logical representation and analysis of multiparty access control. The details about prototype implementation and experimental results are described in Section 5. Section 6 discusses how to tackle collusion attacks followed by the related work in Section 7. Section 8 concludes this paper and discusses our future directions.

2 Multiparty Access Control for OSNs: Requirements and Patterns

In this section we proceed with a comprehensive requirement analysis of multiparty access control in OSNs.
policy that permits his friend list visible to anyone. In this case, if OSNs can solely enforce one party’s policy, the relationship between Alice and Bob can still be learned through Bob’s friend list. Figure 1(b) shows a relationship sharing pattern where a user called owner, who has a relationship with another user called stakeholder, shares the relationship with an accessor. In this scenario, authorization requirements from both the owner and the stakeholder should be considered. Otherwise, the stakeholder’s privacy concern may be violated.

**Content sharing**: OSNs provide built-in mechanisms enabling users to communicate and share contents with other members. OSN users can post statuses and notes, upload photos and videos in their own spaces, tag others to their contents, and share the contents with their friends. On the other hand, users can also post contents in their friends’ spaces. The shared contents may be connected with multiple users. Consider an example where a photograph contains three users, Alice, Bob and Carol. If Alice uploads it to her own space and tags both Bob and Carol in the photo, we call Alice the owner of the photo, and Bob and Carol stakeholders of the photo. All of them may specify access control policies to control over who can see this photo. Figure 2(a) depicts a content sharing pattern where a user called owner shares the content with other OSN members, and the content has multiple stakeholders who may also want to involve in the control of content sharing. In another case, when Alice posts a note stating “I will attend a party on Friday night with @Carol” to Bob’s space, we call Alice the contributor of the note and she may want to make the control over her notes. In addition, since Carol is explicitly identified by @-mention (at-mention) in this note, she is considered as a stakeholder of the note and may also want to control the exposure of this note. Figure 2(b) shows a content sharing sharing pattern reflecting this scenario where a contributor publishes a content to other’s space and the content may also have multiple stakeholders (e.g., tagged users). All associated users should be allowed to define access control policies for the shared content.

OSNs also enable users to share others’ contents. For example, when Alice views a photo in Bob’s space and decides to share this photo with her friends, the photo will be in turn posted in her space and she can specify access control policy to authorize her friends to see this photo. In this case, Alice is a disseminator of the photo. Since Alice may adopt a weaker control saying the photo is visible to everyone, the initial access control requirements of this photo should be compliant with, preventing from the possible leakage of sensitive information via the procedure of data dissemination. Figure 2(c) shows a content sharing pattern where the sharing starts with an originator (owner or contributor who uploads the content) publishing the content, and then a disseminator views and shares the content. All access control policies defined by associated users should be enforced to regulate access of the content in disseminator’s space. For a more complicated case, the disseminated content may be further re-disseminated by disseminator’s friends, where effective access control mechanisms should be applied in each procedure to regulate sharing behaviors. Especially, regardless of how many steps the content has been re-disseminated, the original access control policies should be always enforced to protect further dissemination of the content.

### 3 Multiparty Access Control Model for OSNs

In this section, we formalize a MultiParty Access Control (MPAC) model for OSNs (Section 3.1), as well as a policy scheme (Section 3.2) and a policy evaluation mechanism.
3.1 MPAC Model

An OSN can be represented by a relationship network, a set of user groups and a collection of user data. The relationship network of an OSN is a directed labeled graph, where each node denotes a user and each edge represents a relationship between two users. The label associated with each edge indicates the type of the relationship. Edge direction denotes that the initial node of an edge establishes the relationship and the terminal node of the edge accepts the relationship. The number and type of supported relationships rely on the specific OSNs and its purposes. Besides, OSNs include an important feature that allows users to be organized in groups [38], [37] (or called circles in Google+ [6]), where each group has a unique name. This feature enables users of an OSN to easily find other users with whom they might share specific interests (e.g., same hobbies), demographic groups (e.g., studying at the same schools), political orientation, and so on. Users can join in groups without any approval from other group members. Furthermore, OSNs provide each member a Web space where users can store and manage their personal data including profile information, friend list and content.

Recently, several access control schemes (e.g., [10], [11], [17], [18]) have been proposed to support fine-grained authorization specifications for OSNs. Unfortunately, these schemes can only allow a single controller, the resource owner, to specify access control policies. Indeed, a flexible access control mechanism in a multi-user environment like OSNs should allow multiple controllers, who are associated with the shared data, to specify access control policies. As we identified previously in the sharing patterns (Section 2), in addition to the owner of data, other controllers, including the contributor, stakeholder and disseminator of data, need to regulate the access of the shared data as well. We define these controllers as follows:

Definition 1: (Owner). Let d be a data item in the space of a user u in the social network. The user u is called the owner of d.

Definition 2: (Contributor). Let d be a data item published by a user u in someone else’s space in the social network. The user u is called the contributor of d.

Definition 3: (Stakeholder). Let d be a data item in the space of a user in the social network. Let T be the set of tagged users associated with d. A user u is called a stakeholder of d, if u \in T.

Definition 4: (Disseminator). Let d be a data item shared by a user u from someone else’s space to his/her space in the social network. The user u is called a disseminator of d.

We now formally define our MPAC model as follows:

- **U** = \{u_1, \ldots, u_n\} is a set of users of the OSN. Each user has a unique identifier;
- **G** = \{g_1, \ldots, g_m\} is a set of groups to which the users can belong. Each group also has a unique identifier;
- **P** = \{p_1, \ldots, p_n\} is a collection of user profile sets, where p_i = \{q_{i1}, \ldots, q_{im}\} is the profile of a user \(i \in U\). Each profile entry is a <attribute: profile-value> pair, \(q_{ij} = \langle\text{attr}_{rj}: \text{pvalue}_{rj}\rangle\), where attr_{rj} is an attribute identifier and pvalue_{rj} is the attribute value;
- **RT** is a set of relationship types supported by the OSN. Each user in an OSN may be connected with others by relationships of different types;
- **R** = \{r_1, \ldots, r_n\} is a collection of user relationship sets, where r_i = \{s_{i1}, \ldots, s_{is}\} is the relationship list of a user \(i \in U\). Each relationship entry is a <user: relationship-type> pair, \(s_{ij} = \langle u_j : r_{tj}\rangle\), where \(u_j \in U\), \(r_{tj} \in RT\);
- **C** = \{c_1, \ldots, c_r\} is a collection of user content sets, where c_i = \{e_{i1}, \ldots, e_{im}\} is a set of contents of a user \(i \in U\), where e_{ij} is a content identifier;
- **D** = \{d_1, \ldots, d_m\} is a collection of data sets, where \(d_i = p_i \cup r_i \cup e_i\) is a set of data of a user \(i \in U\);
- **CT** = \{OW, CB, ST, DS\} is a set of controller types, indicating ownerOf, contributorOf, stakeholderOf, and disseminatorOf, respectively;
- **UU** = \{UU_{rt1}, \ldots, UU_{rtm}\} is a collection of uni-directional binary user-to-user relations, where \(UU_{rt} \subseteq U \times U\) specifies the pairs of users having relationship type \(rt \in RT\);
- **UG** \subseteq U \times G is a set of binary user-to-group membership relations;
- **UD** = \{UD_{ct1}, \ldots, UD_{ctn}\} is a collection of binary user-to-data relations, where \(UD_{ct} \subseteq U \times D\) specifies a set of <user, data> pairs having controller type \(ct \in CT\);
- **relation_members** : U \(\xrightarrow{RT} 2^U\), a function mapping each user u \in U to a set of users with whom s/he has a relationship \(rt \in RT\):
  \[ \text{relation_members}(u : U, rt : RT) = \{u' \in U \mid (u, u') \in UU_{rt}\} \]
- **ROR_members** : U \(\xrightarrow{RT} 2^U\), a function mapping each user u \in U to a set of users with whom s/he has a transitive relation of a relationship \(rt \in RT\), denoted as relationships-of-relationships (ROR). For example, if a relationship is friend, then its transitive relation is friends-of-friends (FOF):
  \[ \text{ROR_members}(u : U, rt : RT) = \{u' \in U \mid u'' \in \text{relation_members}(u, rt) \lor (\exists u'' \in U)[u'' \in \text{ROR_members}(u, rt) \land u' \in \text{ROR_members}(u'', rt)]\} \]
- **controllers** : D \(\xrightarrow{CT} 2^U\), a function mapping each data item d \in D to a set of users who are the controller with the controller type \(ct \in CT\):
  \[ \text{controllers}(d : D, ct : CT) = \{u \in U \mid (u, d) \in UD_{ct}\} \]
- **group_members** : G \(\xrightarrow{gU}\), a function mapping each group \(g \in G\) to a set of users who belong to the group:
  \[ \text{group_members}(g : G) = \{u \in U \mid (u, g) \in UG\} \]
  \[ \text{groups}(u : U) = \{g \in G \mid (u, g) \in UG\} \]

Figure 3 depicts an example of multiparty social network representation. It describes relationships of five in-
individuals, Alice (A), Bob (B), Carol (C), Dave (D) and Edward (E), along with their relations with data and their groups of interest. Note that two users may be directly connected by more than one edge labeled with different relationship types in the relationship network. For example, in Figure 3, Alice (A) has a direct relationship of type colleagueOf with Bob (B), whereas Bob (B) has a relationship of friendOf with Alice (A). In addition, two users may have transitive relationships, such as friends-of-friends (FOF), colleagues-of-colleagues (COC) and classmates-of-classmates (LOL) in this example. Moreover, this example shows that some data items have multiple controllers. For instance, Relationship_A has two controllers: the owner, Alice (A) and a stakeholder, Carol (C). Also, some users may be the controllers of multiple data items. For example, Carol (C) is a stakeholder of Relationship_A as well as the contributor of Content_C. Furthermore, we can notice there are two groups in this example that users can participate in: the “Fashion” group and the “Hiking” group, and some users, such as Bob (B) and Dave (D), may join in multiple groups.

3.2 MPAC Policy Specification

To enable a collaborative authorization management of data sharing in OSNs, it is essential for multiparty access control policies to be in place to regulate access over shared data, representing authorization requirements from multiple associated users. Our policy specification scheme is built upon the proposed MPAC model.

**Accessor Specification:** Accessors are a set of users who are granted to access the shared data. Accessors can be represented with a set of user names, a set of relationship names or a set of group names in OSNs. We formally define the accessor specification as follows:

**Definition 5:** (Accessor Specification). Let ac ∈ U ∪ RT ∪ G be a user u ∈ U, a relationship type rt ∈ RT, or a group g ∈ G. Let at ∈ {UN, RN, GN} be the type of the accessor specification (user name, relationship type, and group name, respectively). The accessor specification is defined as a set, accessors = {a₁, …, aₙ}, where each element is a tuple < ac, at >.

**Data Specification:** In OSNs, user data is composed of three types of information, user profile, user relationship and user content. To facilitate effective privacy conflict resolution for multiparty access control, we introduce sensitivity levels for data specification, which are assigned by the controllers to the shared data items. A user’s judgment of the sensitivity level of the data is not binary (private/public), but multi-dimensional with varying degrees of sensitivity. Formally, the data specification is defined as follows:

**Definition 6:** (Data Specification). Let dt ∈ D be a data item. Let sl be a sensitivity level, which is a rational number in the range [0,1], assigned to dt. The data specification is defined as a tuple < dt, sl >.

**Access Control Policy:** To summarize the above-mentioned policy elements, we introduce the definition of a multiparty access control policy as follows:

**Definition 7:** (MPAC Policy). A multiparty access control policy is a 5-tuple P =< controller, ctype, accessor, data, effect >, where

- controller ∈ U is a user who can regulate the access of data;
- ctype ∈ CT is the type of the controller;
- accessor is a set of users to whom the authorization is granted, representing with an access specification defined in Definition 5.
- data is represented with a data specification defined in Definition 6; and
- effect ∈ {permit, deny} is the authorization effect of the policy.

Suppose a controller can leverage five sensitivity levels: 0.00 (none), 0.25 (low), 0.50 (medium), 0.75 (high), and 1.00 (highest) for the shared data. We illustrate several examples of MPAC policies for OSNs as follows:

The MPAC policies

1. “Alice authorizes her friends to view her status identified by status01 with a medium sensitivity level, where Alice is the owner of the status.”
2. “Bob authorizes users who are his colleagues or in hiking group to view a photo, summer.jpg, that he is tagged in with a high sensitivity level, where Bob is a stakeholder of the photo.”
3. “Carol disallows Dave and Edward to watch a video, play.avi, that she uploads to someone else’s spaces with a highest sensitivity level, where Carol is the contributor of the video.”

are expressed as:

1. p₁ = (Alice, OW, {< friendOf, RN >}, < status01, 0.50 >, permit)
2. p₂ = (Bob, ST, {< colleagueOf, RN >}, < hiking, GN >), < summer.jpg, 0.75 >, permit)
3. p₃ = (Carol, CB, {< Dave, UN >}, < Edward, UN >, < play.avi, 1.00 >, deny)

3.3 Multiparty Policy Evaluation

Two steps are performed to evaluate an access request over multiparty access control policies. The first step checks the access request against the policy specified by each controller and yields a decision for the controller. The accessor element in a policy decides whether the policy is applicable to a request. If the user who sends the request belongs to the user set derived from the accessor of a policy, the policy
is applicable and the evaluation process returns a response with the decision (either permit or deny) indicated by the effect element in the policy. Otherwise, the response yields deny decision if the policy is not applicable to the request. In the second step, decisions from all controllers responding to the access request are aggregated to make a final decision for the access request. Figure 4 illustrates the evaluation process of multiparty access control policies. Since data controllers may generate different decisions (permit and deny) for an access request, conflicts may occur. In order to make an unambiguous decision for each access request, it is essential to adopt a systematic conflict resolution mechanism to resolve those conflicts during multiparty policy evaluation.

The essential reason leading to the conflicts—especially privacy conflicts—is that multiple controllers of the shared data item often have different privacy concerns over the data item. For example, assume that Alice and Bob are two controllers of a photo. Both of them defines their own access control policy stating only her/his friends can view this photo. Since it is almost impossible that Alice and Bob have the same set of friends, privacy conflicts may always exist when considering multiparty control over the shared data item.

A naive solution for resolving multiparty privacy conflicts is to only allow the common users of accessor sets defined by the multiple controllers to access the data item. Unfortunately, this strategy is too restrictive in many cases and may not produce desirable results for resolving multiparty privacy conflicts. Consider an example that four users, Alice, Bob, Carol and Dave, are the controllers of a photo, and each of them allows her/his friends to see the photo. Suppose that Alice, Bob and Carol are close friends and have many common friends, but Dave has no common friends with them and also has a pretty weak privacy concern on the photo. In this case, adopting the naive solution for conflict resolution may turn out that no one can access this photo. However, it is reasonable to give the view permission to the common friends of Alice, Bob and Carol. A strong conflict resolution strategy may provide a better privacy protection. Meanwhile, it may reduce the social value of data sharing in OSNs. Therefore, it is important to consider the tradeoff between privacy and utility when resolving privacy conflicts. To address this issue, we introduce a simple but flexible voting scheme for resolving multiparty privacy conflicts in OSNs.

3.3.1 A voting scheme for decision making of multiparty control

Majority voting is a popular mechanism for decision making [24]. Inspired by such a decision making mechanism, we propose a voting scheme to achieve an effective multiparty conflict resolution for OSNs. A notable feature of the voting mechanism for conflict resolution is that the decision from each controller is able to have an effect on the final decision. Our voting scheme contains two voting mechanisms, decision voting and sensitivity voting.

**Decision Voting:** A decision voting value (DV) derived from the policy evaluation is defined as follows, where Evaluation(p) returns the decision of a policy p:

$$DV = \begin{cases} 0 & \text{if } \text{Evaluation}(p) = \text{Deny} \\ 1 & \text{if } \text{Evaluation}(p) = \text{Permit} \end{cases}$$  \hspace{1cm} (1)

Assume that all controllers are equally important, an aggregated decision value (DV$_{ag}$) (with a range of 0.00 to 1.00) from multiple controllers including the owner (DV$_{ow}$), the contributor (DV$_{cb}$) and stakeholders (DV$_{st}$), is computed with following equation:

$$DV_{ag} = (DV_{ow} + DV_{cb} + \sum_{i \in SS} DV_{st}^i) \times \frac{1}{m}$$  \hspace{1cm} (2)

where SS is the stakeholder set of the shared data item, and m is the number of controllers of the shared data item.

Each controller of the shared data item may have (i) a different trust level over the data owner and (ii) a different reputation value in terms of collaborative control. Thus, a generalized decision voting scheme needs to introduce weights, which can be calculated by aggregating trust levels and reputation values [19], on different controllers. Different weights of controllers are essentially represented by different importance degrees on the aggregated decision. In general, the importance degree of controller $x$ is “weight of sum of weights”. Suppose that $\omega_{ow}$, $\omega_{cb}$ and $\omega_{st}$ are weight values for owner, contributor and stakeholders, respectively, and $n$ is the number of stakeholders of the shared data item.

A weighted decision voting scheme is as follows:

$$DV_{ag} = (\omega_{ow} \times DV_{ow} + \omega_{cb} \times DV_{cb} + \sum_{i=1}^{n} (\omega_{st}^i \times DV_{st}^i)) \times \frac{1}{\omega_{ow} + \omega_{cb} + \sum_{i=1}^{n} \omega_{st}^i}$$ \hspace{1cm} (3)

**Sensitivity Voting:** Each controller assigns a sensitivity level (SL) to the shared data item to reflect her/his privacy concern. A sensitivity score (Sc) (in the range from 0.00 to 1.00) for the data item can be calculated based on following equation:

$$Sc = (SL_{ow} + SL_{cb} + \sum_{i \in SS} SL_{st}^i) \times \frac{1}{m}$$ \hspace{1cm} (4)

Note that we can also use a generalized sensitivity voting scheme like equation (3) to compute the sensitivity score (Sc).
3.3.2 Threshold-based conflict resolution

A basic idea of our approach for threshold-based conflict resolution is that the sensitivity score (Sc) can be utilized as a threshold for decision making. Intuitively, if the sensitivity score is higher, the final decision has a high chance to deny access, taking into account the privacy protection of high sensitive data. Otherwise, the final decision is very likely to allow access, so that the utility of OSN services cannot be affected. The final decision is made automatically by OSN systems with this threshold-based conflict resolution as follows:

\[
\text{Decision} = \begin{cases} 
\text{Permit} & \text{if } DV_{ag} > \text{Sc} \\
\text{Deny} & \text{if } DV_{ag} \leq \text{Sc}
\end{cases}
\]  

(5)

It is worth noticing that our conflict resolution approach has an adaptive feature which reflects the changes of policies and sensitivity levels. If any controller changes her/his policy or sensitivity level for the shared data item, the aggregated decision value (DV\(_{ag}\)) and the sensitivity score (Sc) will be recomputed and the final decision may be changed accordingly.

3.3.3 Strategy-based conflict resolution with privacy recommendation

Above threshold-based conflict resolution provides a quite fair mechanism for making the final decision when we treat all controllers equally important. However, in practice, different controllers may have different priorities in making impact on the final decision. In particular, the owner of data item may be desirable to possess the highest priority in the control of shared data item. Thus, we further provide a strategy-based conflict resolution mechanism to fulfill specific authorization requirements from the owners of shared data.

In this conflict resolution, the sensitivity score (Sc) of a data item is considered as a guideline for the owner of shared data item in selecting an appropriate strategy for conflict resolution. We introduce following strategies for the purpose of resolving multiparty privacy conflicts in OSNs.

- **Owner-overrides**: the owner's decision has the highest priority. This strategy achieves the owner control mechanism that most OSNs are currently utilizing for data sharing. Based on the weighted decision voting scheme, we set \(\omega_{ow} = 1\), \(\omega_{ch} = 0\) and \(\omega_{st} = 0\), and the final decision can be made as follows:

\[
\text{Decision} = \begin{cases} 
\text{Permit} & \text{if } DV_{ag} = 1 \\
\text{Deny} & \text{if } DV_{ag} = 0
\end{cases}
\]  

(6)

- **Full-consensus-permit**: if any controller denies the access, the final decision is deny. This strategy can achieve the naive conflict resolution that we discussed previously. The final decision can be derived as:

\[
\text{Decision} = \begin{cases} 
\text{Permit} & \text{if } DV_{ag} = 1 \\
\text{Deny} & \text{otherwise}
\end{cases}
\]  

(7)

1. In real system implementation, those weights need to be adjusted by the system.

- **Majority-permit**: this strategy permits (denies, resp.) a request if the number of controllers to permit (deny, resp.) the request is greater than the number of controllers to deny (permit, resp.) the request. The final decision can be made as:

\[
\text{Decision} = \begin{cases} 
\text{Permit} & \text{if } DV_{ag} \geq 1/2 \\
\text{Deny} & \text{if } DV_{ag} < 1/2
\end{cases}
\]  

(8)

Other majority voting strategies [27] can be easily supported by our voting scheme, such as strong-majority-permit (this strategy permits a request if over 2/3 controllers permit it), super-majority-permit (this strategy permits a request if over 3/4 controllers permit it).

3.3.4 Conflict resolution for dissemination control

A user can share others’ contents with her/his friends in OSNs. In this case, the user is a disseminator of the content, and the content will be posted in the disseminator’s space and visible to her/his friends or the public. Since a disseminator may adopt a weaker control over the disseminated content but the content may be much more sensitive from the perspective of original controllers of the content, the privacy concerns from the original controllers of the content should be always fulfilled, preventing inadvertent disclosure of sensitive contents. In other words, the original access control policies should be always enforced to restrict access to the disseminated content. Thus, the final decision for an access request to the disseminated content is a composition of the decisions aggregated from original controllers and the decision from the current disseminator. In order to eliminate the risk of possible leakage of sensitive information from the procedure of data dissemination, we leverage a restrictive conflict resolution strategy, Deny-overrides, to resolve conflicts between original controllers’ decision and the disseminator’s decision. In such a context, if either of those decisions is to deny the access request, the final decision is deny. Otherwise, if both of them are permit, the final decision is permit.

4 Logical representation and analysis of multiparty access control

In this section, we adopt answer set programming (ASP), a recent form of declarative programming [28], [29], to formally represent our model, which essentially provide a formal foundation of our model in terms of ASP-based representation. Then, we demonstrate how the correctness analysis and authorization analysis of multiparty access control can be carried out based on such a logical representation.

4.1 Representing Multiparty Access Control in ASP

We introduce a translation module that turns multiparty authorization specification into an ASP program. This interprets a formal semantics of multiparty authorization specification in terms of the Answer Set semantics\(^2\).

---

2. We identify “\(\wedge\)” with “\(\land\)”, “\(\leftrightarrow\)” with “\(\Rightarrow\)” and a policy of the form \(A \leftrightarrow B, C \lor D\) as a set of the two policies \(A \leftarrow B, C\) and \(A \leftarrow B, D\).
4.1.1 Logical definition of multiple controllers and transitive relationships

The basic components and relations in our MPAC model can be directly defined with corresponding predicates in ASP. We have defined $UD_{ct}$ as a set of user-to-data relations with controller type $ct \in CT$. Then, the logical definition of multiple controllers is as follows:

- The owner of a data item can be represented as:
  $\text{OW}(\text{controller}, \text{data}) \leftarrow U(\text{controller}) \land D(\text{data})$.
- The contributor of a data item can be represented as:
  $\text{CB}(\text{controller}, \text{data}) \leftarrow U_{\text{CB}}(\text{controller}, \text{data}) \land U(\text{controller}) \land D(\text{data})$.
- The stakeholder of a data item can be represented as:
  $\text{ST}(\text{controller}, \text{data}) \leftarrow U_{\text{ST}}(\text{controller}, \text{data}) \land U(\text{controller}) \land D(\text{data})$.
- The disseminator of a data item can be represented as:
  $\text{DS}(\text{controller}, \text{data}) \leftarrow U_{\text{DS}}(\text{controller}, \text{data}) \land U(\text{controller}) \land D(\text{data})$.

Our MPAC model supports transitive relationships. For example, David is a friend of Allice, and Edward is a friend of David in a social network. Then, we call Edward is a friend of friend of Allice. The friend relation between two users Allice and David is represented in ASP as follows:

$$\text{friendsOf}(\text{Allice}, \text{David}).$$

It is known that the transitive closure (e.g., reachability) cannot be expressed in first-order logic [29], however, it can be easily handled in the stable model semantics. Then, friends-of-friends can be represented as a transitive closure of friend relation with ASP as follows:

$$\text{friendsOf}(\text{friends}(\text{U1}, \text{U2})) \leftarrow \text{friendsOf}(\text{U1}, \text{U2}),$$

$$\text{friendsOf}(\text{friends}(\text{U1}, \text{U3})) \leftarrow \text{friendsOf}(\text{friends}(\text{U1}, \text{U2})),$$

$$\text{friendsOf}(\text{friends}(\text{U2}, \text{U3})).$$

4.1.2 Logical policy specification

The translation module converts a multiparty authorization policy containing the type of the accessor specification $ata = RN$ into a set of ASP rules

$$\text{decision}(\text{controller}, \text{effect}) \leftarrow \bigvee_{1 \leq k \leq n} U(\text{controller}) \land D(\text{dt}) \land \text{ct}(\text{controller}, \text{data}) \land \text{U}(\text{controller}) \land \text{X}(\text{X}) \land \text{D}(\text{dt}).$$

For example, $p_3$ in Section 3.2 can be translated into an ASP rule

$$\text{decision}(\text{Carol}, \text{deny}) \leftarrow U(\text{Dave}) \lor U(\text{Edward}) \land \text{CB}(\text{Carol}, \text{videoId}) \land U(\text{Carol}) \land \text{video}(\text{videoId}).$$

The translation module converts a multiparty authorization policy including the type of the accessor specification $ata = GN$ into an ASP rule

$$\text{decision}(\text{controller}, \text{effect}) \leftarrow \bigvee_{1 \leq k \leq n} U(\text{controller}) \land \text{ct}(\text{controller}, \text{data}) \land \text{U}(\text{controller}) \land \text{X}(\text{X}) \land \text{D}(\text{dt}).$$

For example, $p_5$ in Section 3.2 specifies the accessor with a relationship name (RN) and a group name (GN). Thus, this policy can be translated into an ASP rule as:

$$\text{decision}(\text{Bob}, \text{permit}) \leftarrow \text{colleagueOf}(\text{Bob}, \text{X}) \lor \text{U}(\text{Bob}) \land \text{X}(\text{X}) \land \text{ST}(\text{Bob}, \text{photoId}) \land \text{U}(\text{Bob}) \land \text{X}(\text{X}) \land \text{photo}(\text{photoId}).$$

4.1.3 Logical representation of conflict resolution mechanism

Our voting schemes are represented in ASP rules as follows:

$$\text{decision_voting}(\text{C}) = 1 \leftarrow \text{decision}(\text{C}, \text{permit}).$$

$$\text{decision_voting}(\text{C}) = 0 \leftarrow \text{decision}(\text{C}, \text{deny}).$$

$$\text{aggregation_weight}(\text{K}) \leftarrow \text{sum}\{\text{weight}(\text{C}) : \text{controller}(\text{C})\}.$$ 

$$\text{aggregation_decision}(\text{N}) \leftarrow \text{N} = \text{sum}\{\text{decision_voting}(\text{C}) \times \text{weight}(\text{C}) : \text{controller}(\text{C})\}.$$ 

$$\text{aggregation_sensitivity}(\text{M}) \leftarrow \text{M} = \text{sum}\{\text{sensitivity_voting}(\text{C}) \times \text{weight}(\text{C}) : \text{controller}(\text{C})\}.$$ 

Our threshold-based conflict resolution mechanism is represented as:

$$\text{decision}(\text{controllers}, \text{permit}) \leftarrow \text{N} > \text{M} \land$$

$$\text{aggregation_decision}(\text{N}) \land \text{aggregation_sensitivity}(\text{M}).$$

Our strategy-based conflict resolution mechanism is represented with ASP as well. For example,

- The conflict resolution strategy Owner overrides is represented in ASP rules as follows

$$\text{weight}(\text{controllers}) = 1 \leftarrow \text{OW}(\text{controller}, \text{data}).$$

$$\text{weight}(\text{controllers}) = 0 \leftarrow \text{CB}(\text{controller}, \text{data}).$$

$$\text{decision}(\text{controllers}, \text{permit}) \leftarrow \text{N} / \text{K} = 1 \land$$

$$\text{aggregation_weight}(\text{K}) \land \text{aggregation_decision}(\text{N}).$$

$$\text{decision}(\text{controllers}, \text{deny}) \leftarrow \text{not} \text{decision}(\text{controllers}, \text{permit}).$$
• The conflict resolution strategy Full-consensus-permit is represented as:

\[
\text{decision(controllers, permit)} \leftarrow N/K = 1 \land \text{aggregation_weight}(K) \land \text{aggregation_decision}(N),
\]

\[
\text{decision(controllers, deny)} \leftarrow \neg \text{decision(controllers, permit)}.
\]

• The conflict resolution strategy Majority-permit is represented with following ASP rules:

\[
\text{decision(controllers, permit)} \leftarrow N/K > 1/2 \land \text{aggregation_weight}(K) \land \text{aggregation_decision}(N),
\]

\[
\text{decision(controllers, deny)} \leftarrow \neg \text{decision(controllers, permit)}.
\]

• The conflict resolution strategy Deny-overrides for dissemination control is represented as follows:

\[
\text{decision(deny)} \leftarrow \text{decision(controllers, deny)}.
\]

\[
\text{decision(deny)} \leftarrow \text{decision(distributor, deny)}.
\]

\[
\text{decision(permit)} \leftarrow \neg \text{decision(deny)}.
\]

4.2 Reasoning about Multiparty Access Control

One of the promising advantages in logic-based representation of access control is that formal reasoning of the authorization properties can be achieved. Once we represent our multiparty access control model into an ASP program, we can use off-the-shelf ASP solvers to carry out several automated analysis tasks.

The problem of verifying an authorization property against our model description can be cast into the problem of checking whether the program

\[
\Pi \cup \Pi_{\text{query}} \cup \Pi_{\text{config}}
\]

has no answer sets, where \( \Pi \) is the program corresponding to the model specification, \( \Pi_{\text{query}} \) is the program corresponding to the program that encodes the negation of the property to check, and \( \Pi_{\text{config}} \) is the program that can generate arbitrary configurations, for example:

\[
\text{user_attributes}(alice, bob, carol, dave, edward).
\]

\[
\text{group_attributes}(fashion, hiking).
\]

\[
\text{photo_attributes}(photoid).
\]

\[
1\{\text{user}(X) : \text{user_attributes}(X)}\).
\]

\[
1\{\text{group}(X) : \text{group_attributes}(X)}\).
\]

\[
1\{\text{photo}(X) : \text{photo_attributes}(X)}\).
\]

Correctness Analysis: is used to prove if our proposed access control model is valid.

Example 1: If we want to check whether the conflict resolution strategy Full-consensus-permit is correctly defined based on the voting scheme in our model, the input query \( \Pi_{\text{query}} \) can be represented as follows:

\[
\text{decision(controllers, permit)} \leftarrow \bigvee_{C \in \Pi_{\text{OS}}} \text{decision}(C, \text{deny}).
\]

If no answer set is found, this implies that the authorization property is verified. Otherwise, an answer set returned by an ASP solver serves as a counterexample that indicates why the description does not entail the authorization property. This helps determine the problems in the model definition.

Authorization Analysis: is employed to examine over-sharing (does current authorization state disclose the data to some users undesirable?) and under-sharing (does current authorization state disallow some users to access the data that they are supposed to be allowed?). This analysis service should be incorporated into OSN systems to enable users checking potential authorization impacts derived from collaborative control of shared data.

Example 2: (Checking over-sharing) Alice has defined a policy to disallow her family members to see a photo. Then, she wants to check if any family members can see this photo after applying conflict resolution mechanism for collaborative authorization management considering different privacy preferences from multiple controllers. The input query \( \Pi_{\text{query}} \) can be represented as follows:

\[
\text{check} :- \text{decision(permit)}, \text{familyof}(alice, x), \text{ow}(alice, photoid), \text{user}(alice), \text{user}(x), \text{photo}(photoid).
\]

\[
:- \text{not check}.
\]

If any answer set is found, it means that there are family members who can see the photo. Thus, from the perspective of Alice’s authorization, this photo is over shared to some users. The possible solution is to adopt a more restricted conflict resolution strategy or increase the weight value of Alice.

Example 3: (Checking under-sharing) Bob has defined a policy to authorize his friends to see a photo. He wants to check if any friends cannot see this photo in current system. The input query \( \Pi_{\text{query}} \) can be specified as follows:

\[
\text{check} :- \text{decision(deny)}, \text{friendof}(bob, x), \text{ow}(alice, photoid), \text{user}(bob), \text{user}(x), \text{photo}(photoid).
\]

\[
:- \text{not check}.
\]

If an answer set contains check, this means that there are friends who cannot view the photo. Regarding Bob’s authorization requirement, this photo is under shared with his friends.

5 Implementation and Evaluation

5.1 Prototype Implementation

We implemented a proof-of-concept Facebook application for the collaborative management of shared data, called MController (http://apps.facebook.com/MController). Our prototype application enables multiple associated users to specify their authorization policies and privacy preferences to co-control a shared data item. It is worth noting that our current implementation was restricted to handle photo sharing in OSNs. Obversely, our approach can be generalized to deal with other kinds of data sharing, such as videos and comments, in OSNs as long as the stakeholder of shared data are identified with effective methods like tagging or searching.

Figure 5 shows the architecture of MController, which is divided into two major pieces, Facebook server and application server. The Facebook server provides an entry point via the Facebook application page, and provides
references to photos, friendships, and feed data through API calls. Facebook server accepts inputs from users, then forwards them to the application server. The application server is responsible for the input processing and collaborative management of shared data. Information related to user data such as user identifiers, friend lists, user groups, and user contents are stored in the application database. Users can access the *MController* application through Facebook, which serves the application in an iFrame. When access requests are made to the decision making portion in the application server, results are returned in the form of access to photos or proper information about access to photos. In addition, when privacy changes are made, the decision making portion returns change-impact information to the interface to alert the user. Moreover, analysis services in *MController* application are provided by implementing an ASP translator, which communicates with an ASP reasoner. Users can leverage the analysis services to perform complicated authorization queries.

*MController* is developed as a third-party Facebook application, which is hosted in an Apache Tomcat application server supporting PHP and MySQL database. *MController* application is based on the iFrame external application approach. Using the Javascript and PHP SDK, it accesses users’ Facebook data through the Graph API and Facebook Query Language. Once a user installs *MController* in her/his Facebook space and accepts the necessary permissions, *MController* can access a user’s basic information and contents. Especially, *MController* can retrieve and list all photos, which are owned or uploaded by the user, or where the user was tagged. Once information is imported, the user accesses *MController* through its application page on Facebook, where s/he can query access information, set privacy for photos that s/he is a controller, or view photos s/he is allowed to access.

A core component of *MController* is the decision making module, which processes access requests and returns responses (either permit or deny) for the requests. Figure 6 depicts a system architecture of the decision making module in *MController*. To evaluate an access request, the policies of each controller of the targeted content are enforced first to generate a decision for the controller. Then, the decisions of all controllers are aggregated to yield a final decision as the response of the request. Multiparty privacy conflicts are resolved based on the configured conflict resolution mechanism when aggregating the decisions of controllers. If the owner of the content chooses automatic conflict resolution, the aggregated sensitivity value is utilized as a threshold for decision making. Otherwise, multiparty privacy conflicts are resolved by applying the strategy selected by the owner, and the aggregated sensitivity score is considered as a recommendation for strategy selection. Regarding the access requests to disseminated content, the final decision is made by combining the disseminator’s decision and original controllers’ decision adopting corresponding combination strategy discussed previously.

A snapshot of main interface of *MController* is shown in Figure 7 (a). All photos are loaded into a gallery-style interface. To control photo sharing, the user clicks the “Owned”, “Tagged”, “Contributed”, or “Disseminated” tabs, then selects any photo to define her/his privacy preference by clicking the lock below the gallery. If the user is not the owner of selected photo, s/he can only edit the privacy setting and sensitivity setting of the photo. Otherwise, as shown in Figure 7 (c), if the user is the owner of the photo, s/he has the option of clicking “Show Advanced Controls” to assign weight values to different types of controllers and configure the conflict resolution mechanism for the shared photo. By default, the conflict resolution is set to automatic. However, if the owner chooses to set a manual conflict resolution, s/he is informed of a sensitivity score of shared photo and receives a recommendation for choosing an appropriate conflict resolution strategy. Once a controller saves her/his privacy setting, a corresponding feedback is provided to indicate the potential authorization impact of her/his choice. The controller can immediately determine how many users can see the photo and should be denied, and how many users cannot see the photo and should be allowed. *MController* can also display the details of all users who violate against the controller’s privacy setting (See Figure 7 (d)). The purpose of such feedback information is to guide the controller to evaluate the impact of collaborative authorization. If the controller is not satisfied with the current privacy control, s/he may adjust her/his privacy setting, contact the owner of the photo

![Fig. 5. Overall Architecture of MController Application.](image)

![Fig. 6. System Architecture of Decision Making in MController.](image)

3. Note that users are able to predefine their privacy preferences, which are then applied by default to photos they need to control.
4. In our future work, we will explore how each controller can be assigned with a weight. Also, an automatic mechanism to determine weights of controllers will be studied, considering the trust and reputation level of controllers for the collaborative control.
to ask her/him to change the conflict resolution strategies, or even report a privacy violation to OSN administrators who can delete the photo. A controller can also perform authorization analysis by advanced queries as shown in Figure 7 (b). Both over-sharing and under-sharing can be examined by using such an analysis service in MController.

5.2 System Usability and Performance Evaluation

5.2.1 Participants and Procedure

MController is a functional proof-of-concept implementation of collaborative privacy management. To measure the practicality and usability of our mechanism, we conducted a survey study (n=35) to explore the factors surrounding users’ desires for privacy and discover how we might improve those implemented in MController. Specifically, we were interested in users’ perspectives on the current Facebook privacy system and their desires for more control over photos they do not own. We recruited participants through university mailing lists and through Facebook itself using Facebook’s built-in sharing API. Users were given the opportunity to share our application and play with their friends. While this is not a random sampling, recruiting using the natural dissemination features of Facebook arguably gives an accurate profile of the ecosystem.

Participants were first asked to answer some questions about their usage and perception of Facebook’s privacy controls, then were invited to watch a video (http://bit.ly/MController) describing the concept behind MController. Users were then instructed to install the application using their Facebook profiles and complete the following actions: set privacy settings for a photo they do not own but are tagged in, set privacy settings for a photo they own, set privacy settings for a photo they contributed, and set privacy settings for a photo they disseminated. As users completed these actions, they answered questions on the usability of the controls in MController. Afterward, they were asked to answer further questions and compare their experience with MController to that in Facebook.

5.2.2 User Study of MController

For evaluation purposes, questions (http://goo.gl/eDkaV) were split into three areas: likeability, simplicity, and control. Likeability is a measure of a user’s satisfaction with a system (e.g. “I like the idea of being able to control photos in which I am tagged”). Simplicity is a measure how intuitive and useful the system is (e.g. “Setting my privacy settings for a photo in MController is Complicated (1) to Simple (5)” with a 5-point scale). Control is a measure of the user’s perceived control of their personal data (e.g. “If Facebook implemented controls like MController’s to control photo privacy, my photos would be better protected”). Questions were either True/False or measured on a 5-point likert scale, and all responses were scaled from 0 to 1 for numerical analysis. In the measurement, a higher number
indicates a positive perception or opinion of the system while a lower number indicates a negative one. To analyze the average user perception of the system, we used a 95% confidence interval for the users’ answers. This assumes the population to be mostly normal.

**Before Using MController.** Prior to using MController, users were asked a few questions about their usage of Facebook to determine the user’s perceived usability of the current Facebook privacy controls. Since we were interested in the maximum average perception of Facebook, we looked at the upper bound of the confidence interval.

An average user asserts at most 25% positively about the likability and control of Facebook’s privacy management mechanism, and at most 44% on Facebook’s simplicity as shown in Table 1. This demonstrates an average negative opinion of the Facebook’s privacy controls that users currently must use.

**After Using MController.** Users were then asked to perform a few tasks in MController. Since we were interested in the average minimum opinion of MController, we looked at the lower bound of the confidence interval.

An average user asserts at least 80% positively about the likability and control, and at least 67% positively on MController’s simplicity as shown in Table 1. This demonstrates an average positive opinion of the controls and ideas presented to users in MController.

**Improvement.** Besides viewing user opinions and analyzing the usability of our proof-of-concept application, we also briefly investigated the potential improvement of our application. The lowest score for MController is in the area of simplicity. On average, users found setting privacy settings for a photo more complicated in MController than Facebook. This means that while users appreciated the privacy controls of MController presented to them, it would be desirable to further simplify the process implemented in MController. This could be achieved by adopting learn-based generation of privacy settings for OSNs [15], [32].

### 5.2.3 Performance Evaluation

To evaluate the performance of the policy evaluation mechanism in MController, we changed the number of the controllers of a shared photo from 1 to 20, and assigned each controller with the average number of friends, 130, which is claimed by Facebook statistics [3]. Also, we considered two cases for our evaluation. In the first case, each controller allows “friends” to access the shared photo. In the second case, controllers specify “friends of friends” as the accessors instead of “friends”. In our experiments, we performed 1,000 independent trials and measured the performance of each trial. Since the system performance depends on other processes running at the time of measurement, we had initial discrepancies in our performance. To minimize such an impact, we performed 10 independent trials (a total of 10,000 calculations for each number of controllers). For both cases, the experimental results showed that the policy evaluation time increases linearly with the increase of the number of controllers. With the simplest implementation of our mechanism, where $n$ is the number of controllers of a shared photo, a series of operations essentially takes place $n$ times. There are $O(n)$ MySQL calls and data fetching operations and $O(1)$ for additional operations. Moreover, we could observe there was no significant overhead when we run MController in Facebook.

### 6 Discussions

In our multiparty access control system, a group of users could collude with one another so as to manipulate the final access control decision. Consider an attack scenario, where a set of malicious users may want to make a shared photo available to a wider audience. Suppose they can access the photo, and then they all tag themselves or fake their identities to the photo. In addition, they collude with each other to assign a very low sensitivity level for the photo and specify policies to grant a wider audience to access the photo. With a large number of colluding users, the photo may be disclosed to those users who are not expected to gain the access. To prevent such an attack scenario from occurring, three conditions need to be satisfied: (1) there is no fake identity in OSNs; (2) all tagged users are real users appeared in the photo; and (3) all controllers of the photo are honest to specify their privacy preferences.

Regarding the first condition, two typical attacks, Sybil attacks [14] and Identity Clone attacks [8], have been introduced to OSNs and several effective approaches have been recently proposed to prevent the former [16], [36] and latter attacks [22], respectively. To guarantee the second condition, an effective tag validation mechanism is needed to verify each tagged user against the photo. In our current system, if any users tag themselves or others in a photo, the photo owner will receive a tag notification. Then, the owner can verify the correctness of the tagged users. As effective automated algorithms (e.g., facial recognition [13]) are being developed to recognize people accurately in contents such as photos, automatic tag validation is feasible. Considering the third condition, our current system provides a function (See Figure 7 (d)) to indicate the potential authorization impact with respect to a controller’s privacy preference. Using such a function, the photo owner can examine all users who are granted the access by the collaborative authorization and are not
explicitly granted by the owner her/himself. Thus, it enables the owner to discover potential malicious activities in collaborative control. The detection of collusion behaviors in collaborative systems has been addressed by the recent work [31], [35]. Our future work would integrate an effective collusion detection technique into MPAC. To prevent collusion activities, our current prototype has implemented a function for owner control (See Figure 7 (c)), where the photo owner can disable any controller, who is suspected to be malicious, from participating in collaborative control of the photo. In addition, we would further investigate how users’ reputations–based on their collaboration activities—can be applied to prevent and detect malicious activities in our future work.

7 RELATED WORK

Access control for OSNs is still a relatively new research area. Several access control models for OSNs have been introduced (e.g., [10], [11], [17], [18], [23]). Early access control solutions for OSNs introduced trust-based access control inspired by the developments of trust and reputation computation in OSNs. The D-FOAF system [23] is primarily a Friend of a Friend (FOAF) ontology-based distributed identity management system for OSNs, where relationships are associated with a trust level, which indicates the level of friendship between the users participating in a given relationship. Carminati et al. [10] introduced a conceptually-similar but more comprehensive trust-based access control model. This model allows the specification of access rules for online resources, where authorized users are denoted in terms of the relationship type, depth, and trust level between users in OSNs. They further presented a semi-decentralized discretionary access control model and a related enforcement mechanism for controlled sharing of information in OSNs [11]. Fong et al. [18] proposed an access control model that formalizes and generalizes the access control mechanism implemented in Facebook, admitting arbitrary policy vocabularies that are based on theoretical graph properties. Gates [12] described relationship-based access control as one of new security paradigms that addresses unique requirements of Web 2.0. Then, Fong [17] recently formulated this paradigm called a Relationship-Based Access Control (ReBAC) model that bases authorization decisions on the relationships between the resource owner and the resource accessor in an OSN. However, none of these existing work could model and analyze access control requirements with respect to collaborative authorization management of shared data in OSNs.

The need of joint management for data sharing, especially photo sharing, in OSNs has been recognized by the recent work [7], [9], [21], [33]. Squicciarini et al. [33] provided a solution for collective privacy management in OSNs. Their work considered access control policies of a content that is co-owned by multiple users in an OSN, such that each co-owner may separately specify her/his own privacy preference for the shared content. The Clarke-Tax mechanism was adopted to enable the collective enforcement of policies for shared contents. Game theory was applied to evaluate the scheme. However, a general drawback of their solution is the usability issue, as it could be very hard for ordinary OSN users to comprehend the Clarke-Tax mechanism and specify appropriate bid values for auctions. Also, the auction process adopted in their approach indicates that only the winning bids could determine who can access the data, instead of accommodating all stakeholders’ privacy preferences. Carminati et al. [9] recently introduced a new class of security policies, called collaborative security policies, that basically enhance topology-based access control with respect to a set of collaborative users. This work considers user collaboration during both access control enforcement and policy specification, and employs semantic web technologies to support a rich way for denoting collaborative users based on their relationships with the associated resources. In contrast, our work proposes a formal model to address the multiparty access control issue in OSNs, along with a general policy specification scheme and a simple but flexible conflict resolution mechanism for collaborative management of shared data in OSNs. In particular, our proposed solution can also conduct various analysis tasks on access control mechanisms used in OSNs, which has not been addressed by prior work.

8 CONCLUSION

In this paper, we have proposed a novel solution for collaborative management of shared data in OSNs. A multiparty access control model was formulated, along with a multiparty policy specification scheme and corresponding policy evaluation mechanism. In addition, we have introduced an approach for representing and reasoning about our proposed model. A proof-of-concept implementation of our solution called MController has been discussed as well, followed by the usability study and system evaluation of our method.

As part of future work, we are planning to investigate more comprehensive privacy conflict resolution approach and analysis services for collaborative management of shared data in OSNs. Also, we would explore more criteria to evaluate the features of our proposed MPAC model. For example, one of our recent work has evaluated the effectiveness of MPAC conflict resolution approach based on the tradeoff of privacy risk and sharing loss [21]. In addition, users may be involved in the control of a larger number of shared photos and the configurations of the privacy preferences may become time-consuming and tedious tasks. Therefore, we would study inference-based techniques [15], [34] for automatically configure privacy preferences in MPAC. Besides, we plan to systematically integrate the notion of trust and reputation into our MPAC model and investigate a comprehensive solution to cope with collusion attacks for providing a robust MPAC service in OSNs.

REFERENCES
