LAMP - Label-Based Access-Control for More Privacy in Online Social Networks

Leila Bahri, Barbara Carminati, Elena Ferrari, William Lucia

LAMP - Label-based Access-control for More Privacy in Online Social Networks

Leila Bahri, Barbara Carminati, Elena Ferrari, and William Lucia

DiSTA, Insubria University, Italy
{fname.lname}@uninsubria.it

Abstract. Access control in Online Social Networks (OSNs) is generally approached with a relationship-based model. This limits the options in expressing privacy preferences to only the types of relationships users establish in the OSN. Moreover, current proposals do not address the privacy of *dependent* information types, such as comments or likes, at their atomic levels of ownership. Rather, the privacy of these data elements is holistically dependent on the aggregate object they belong to. To overcome this, we propose LAMP, a model that deploys fine grained label-based access control for information sharing in OSNs. Users in LAMP assign customized labels to their friends and to all types of their information; whereas access requests are evaluated by security properties carefully designed to establish orders between requestor's and information's labels. We prove the correctness of the suggested model, and we perform performance experiments based on different access scenarios simulated on a real OSN graph. We also performed a preliminary usability study that compared LAMP to Facebook privacy settings.

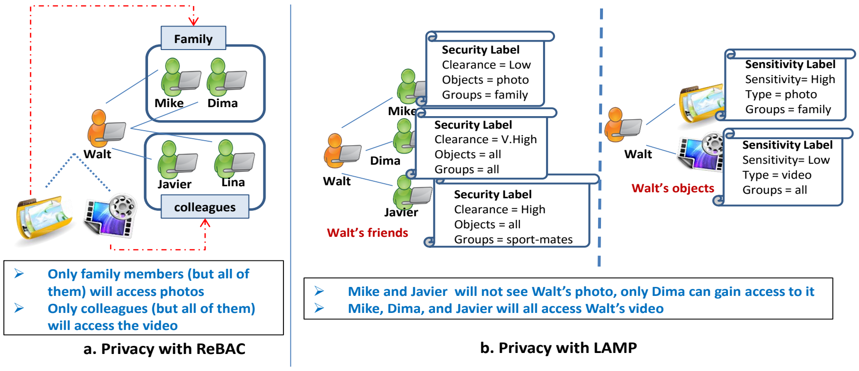
1 Introduction

Online Social Networks (OSNs) enable users to have more freedom and proximity in keeping in touch with their friends and in expanding their social contacts. However, they also create serious privacy concerns given the personal nature of information users share over them on almost a daily basis [7][3]. Users publish their personal stories and updates, as they might also express their opinion by interacting on information shared by others, but, in most cases, they are not fully aware of the size of the audience that gets access to their information.[1] Moreover, privacy settings currently available in OSNs remain both complicated to use, and not flexible enough to model all the privacy preferences that users may require [10].

This limitation seems to come, fundamentally, from relying solely on a relationship based model for access control (ReBAC), as mostly adopted by nowadays OSNs and research proposals. ReBAC is characterized by the explicit tracking of interpersonal relationships among users, and the expression of access control policies in terms of these relationships [5]. These relationships could refer to

[1] http://www.americanbar.org/publications/blt/2014/01/03a_claypoole.html

explicit friendship links established by users [2][6], or they could be inferred from relationships created between users through the resources they share [4] (e.g., connecting users who are tagged in the same photo), or by linking other public information, such as considering attendance to the same school, or originating from the same country as a basis for relationships between users (e.g., the work in [14]).



- Only family members (but all of them) will access photos
- Only colleagues (but all of them) will access the video

a. Privacy with ReBAC

- Mike and Javier will not see Walt's photo, only Dima can gain access to it
- Mike, Dima, and Javier will all access Walt's video

b. Privacy with LAMP

Fig. 1: Privacy management with ReBAC vs. LAMP

However, this type of access control limits, by design, the options for privacy settings. For instance, *defining privacy settings based only on the social relationships implicitly enforces that all the friends of a user who belong to the same relationship type are equal and, hence, will enjoy the same access and interaction rights.* For example, referring to Figure 1.a, if Walt categorizes Mike and Dima as *family*, and Javier and Lina as *colleagues*, then all the information Walt shares with the *family* group will be accessible by both Mike and Dima and the same goes for information shared with the *colleagues* one. In case Walt needs to share an item only with Mike and Javier, he is required to create a new group or categorization under which he declares both of them.

Additionally, the information that users create in OSNs is subject to interactions from their friends, resulting in the creation of intermingled nets of objects with multiple co-owners. For example, Walt shares a photo, and one of his friends tags Dima in it, or Dima shares a status update and Walt comments on it. Such scenarios exemplify the creation of aggregate content that is co-owned by multiple users in the system, who might have conflictual privacy preferences over it. Thus far, this problem has been approached mostly from a multiple ownership perspective. That is, the aggregate co-created content is considered as one single object over which multiple access preferences could be specified by its stakeholders, but only one is selected as dominant using different strategies, such as game theory, and majority voting [8][15][11][16][9]. This may limit the right of the co-owners in guarantying the enforcement of their individual privacy preferences over the specific content they have contributed with.

In contrast, our idea is that of looking at this co-owned content in terms of the relationships between its components. Thus, allowing us to model it as *separate pieces of data that are dependent on each other, but that are uniquely owned and for which unique privacy preferences could be specified by their owners.* Taking this approach simplifies the co-ownership problem and empowers users to be in full control over data they create at a fine granularity level. For example, if Walt is allowed to comment on Dima's status update, he may have the right to limit the audience of his comment and not to have it subject to the one set by Dima for the corresponding status update. Unfortunately, privacy settings available in today's OSNs do not allow the granularity of managing privacy for pieces of data that are dependent on others, such as comments or likes. Moreover, the available related research provides complex solutions that mostly rely on managing multi-ownership at an object's aggregate level, as mentioned above.

To overcome these limitations, in this paper, we propose a new model, LAMP (Label-based Access-control for More Privacy in OSNs), that introduces a new and more flexible way for users to express finer granularity levels of privacy for all information types, respecting the relationships and connections between objects as well. LAMP achieves this by exploiting the security principles of Mandatory Access Control (MAC), according to which a data administrator assigns ordered security levels (that is, labels) to subjects, and ordered sensitivity levels to objects [5]. However, differently from the centralized assignment of labels by a single data administrator as in MAC, LAMP operates based on a discretionary model wherein each user is considered as the ultimate owner of her data.[2] As such, users in LAMP express their privacy preferences by assigning *sensitivity labels* to their owned objects, and *security labels* to their friends.

Labels in LAMP are expressed in terms of security or sensitivity levels, types of relationships, and types of objects (see Figure 1.b). Access decisions are taken based on well defined axioms that consider the richness of interaction types and access privileges that OSNs allow. These axioms ensure that subjects can only gain specific privileges on objects, based on the relationship existing between the object sensitivity levels and their own security levels.

To the best of our knowledge, except from the work in [12], which has discussed design principles for privacy settings by considering co-owned aggregate objects as collections of related annotations, LAMP is the first proposal that considers the relationships between aggregate data objects to decompose them to atomic unique levels of ownership, and that uses a labeling strategy to provide more control and richer privacy settings to data owners on all types of objects they create.

We formalize and prove the security and complexity properties of LAMP, and we discuss how it allows users to express a richer set of privacy settings, compared to existing proposals. We conduct performance experiments using different access scenarios simulated on a real OSN graph. We also deployed LAMP as a Facebook

[2] Using the labeling strategy of MAC under a decentralized approach has been adopted in other systems, such as Oracle Label Security where labels are used in conjunction with DAC functionalities to provide access policy refinement at a table level [13].

test app and performed a preliminary study on LAMP's usability by comparing LAMP to Facebook privacy settings. For space considerations, we report here the performance experiments and refer the reader to a technical report for the preliminary results of the usability experiments [1].

The remainder of this paper is organized as follows. Section 2 introduces a background on the interactions allowed in OSNs nowadays. Section 3 describes the proposed model; whereas Section 4 provides its correctness and complexity properties. Section 5 reports the results of the performed experiments. Finally, Section 6 concludes the paper.

2 Background

In this paper, we consider the scenario of general-purpose OSNs that allow users to establish friendship links with each other, and share different types of information. There are number of such OSNs in the online market nowadays, but it remains safe to say that the two worldwide biggest ones currently in the box are Facebook and Google+. Generally, OSN users are associated with an information space typically made of two main blocks: a profile and a wall or timeline. The profile contains personal information, like full name, gender, age, education etc; whereas the wall organizes, in a graphically timelined manner, the data objects the user shared with her friends or those that her friends have shared with her. Users can upload content of different types into their information space, or interact on their friends' shared information (like, comment, tag, etc.), creating by this aggregate co-owned objects. These aggregate objects are typically made of an information that stands as *independent* by itself, and a tree of objects that are *dependent* on it (see Figure 2 for an example). Table 1 lists the different types of information created in nowadays OSNs, both dependent and independent. The Friend-Post (FP) type on Table 1 denotes an object posted on a user's wall by one of her friends.

Type	**Dependent**
Text (TX) [text posts & profile data]	Independent
Photo (P)	Independent
Video (V)	Independent
Like (L)	Dependent
Comment (C)	Dependent
Tag (TG)	Dependent
Geo-Location (GL)	Dependent
Friend-Post (FP)	Independent

Table 1: Types of information shared in OSNs.

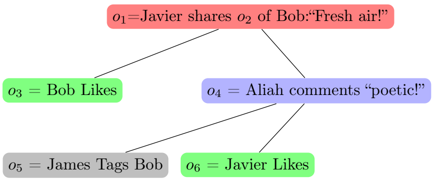

o_1=Javier shares o_2 of Bob:"Fresh air!"
o_3 = Bob Likes
o_4 = Aliah comments "poetic!"
o_5 = James Tags Bob
o_6 = Javier Likes

Fig. 2: Typical objects' structure in OSNs.

The privacy settings currently available in mainstream OSNs, such as Facebook or Google+, present limitations to the expression of users refined privacy needs, especially w.r.t dependent object types (e.g., no privacy settings for objects of types like (L) and comment (C)). LAMP addresses these limitations and offers a more flexible and a richer privacy setting functionality to OSN users. We define and detail the underlying model in the following section.

3 The LAMP Model

We start by presenting basic definitions, then we define how privacy requirements of users can be expressed using labels. Finally, we introduce the mechanism governing access decisions.

3.1 Basic Definitions

We model an OSN as an undirected graph $FG = (U, FR)$, where U is the set of users, and FR is the set of edges. We say that two users $a, b \in U$ are friends in the OSN if and only if there exists one direct edge connecting them, that is, $\exists e_{a,b} \in FR$.

OSN users create and share different types of information (see Table 1) leveraging on the relationships they establish with others. We define the set of information types in the OSN as $OTS = \{TX, P, V, L, C, TG, GL, FP\}$, and we refer to all pieces of information as an *object*, that we formally define as follows:

Definition 1 (OSN object) *An OSN object o is a tuple (type, owner, parent, children, copyof), where:*[3] *(i)* ***type*** *$\in OTS$ denotes the type of the object. (ii)* ***owner*** *$\in U$ denotes the owner of the object. If o.type $\notin \{TG, FP\}$, the owner is the user who generated the object (dependent or independent). Otherwise, it is the user who is tagged or the user on whose wall the friend post is made. (iii)* ***parent****: if o is a dependent object, this refers to the object on which it depends. It is set to null, otherwise. (iv)* ***children****: is the set of objects that depend on o, as generated by interactions on it, if any. It is set to null, otherwise. (v)* ***copyof****: if o is a shared copy of another object $\overline{o}$, this is equal to $\overline{o}$, otherwise it is set to null.*

In addition to the objects that users can create, we consider the existence of another special object type, that we refer to as the *root* object, and that models user walls. We consider that every OSN user owns exactly one *root* object that is created by default upon her subscription to the OSN. Root objects serve for controlling the function of *writing* on users' walls by their friends as we will detail later (see Algorithm 1). A *root* object associated with user a is modelled as *root*$_a$ = *(root, a, null, null, null)*.

Example 1. Consider the OSN objects in Figure 2. By Definition 1, object o_1 is modeled as *(TX, Javier, null,* $\{o_3, o_4\}$*,* o_2*)*, whereas the tag object o_5 is modeled as *(TG, Bob,* o_4*, null, null)*.

Besides creating and sharing objects, there are different types of actions that users can perform on them. We accordingly define the set of controllable privileges in LAMP as *PS = {read, add-comment, add-like, add-tag, share, write}*.[4]

[3] We use the dot notation to refer to the parameters in an object tuple.

[4] We recall that the *write* privilege refers to posting on friends' walls.

As mentioned earlier, access requirements in LAMP are expressed by means of labels, formally defined in the next section.

3.2 Privacy Requirements Formulation

To express privacy requirements, users assign security labels to each of their owned objects and to each of their friends. By labeling friends, users express the limitations they want to put on them, whereas object labeling serves for expressing the sensitivity of the information and its accessibility criteria. We note that we assume a set of possible groups that users can organize their friends into. These groups are created by users depending on their preferences and can be viewed as similar to users' lists or users' circles as available on Facebook and Google+, respectively. A possible example of a set of user groups might be {friends, colleagues, teammates, schoolmates, family, acquaintances}. This set is created at user's side and is only used for the purpose of creating their friends and objects security labels. In addition, we also assume in the system a totally ordered set of levels LOS = *{Unclassified (UC), Very Low (VL), Low (L), Medium (M), High (H), Very High (VH)}*, with UC < VL < L < M < H < VH.

We refer to a friend label as *Friend Clearance Label (FCL)* and we formally define it as follows:

Definition 2 (Friend Clearance Label - FCL) *Let $a, b \in U$ be two friends in the OSN (i.e., $\exists e_{a,b} \in FR$). The label $FCL_{a,b}$, assigned by user a to user b, is denoted by the tuple $(CL_{a,b}, TS_{a,b}, GS_{a,b})$, where: (i) $CL_{a,b} \in LOS$ is the clearance level that a grants to b; (ii) $TS_{a,b} \subseteq OTS$ is the set of object types that a wants b to get access to; (iii) $GS_{a,b}$ is the set of groups that a assigns to b.*

We refer to object labels as *Object Sensitivity Labels (OSL)*, and we formally define them as follows:

Definition 3 (Object Sensitivity Label - OSL) *Let $a \in U$ be a user in the OSN and let o be an OSN object such that o.owner = a. The label OSL_o of object o is denoted by the tuple (SL_o, TY_o, GS_o), where: (i) $SL_o \in LOS$ is the sensitivity level of o; (ii) TY_o = o.type is the type of o; (iii) GS_o is the set of groups for which object o should be available.*

Example 2. Assume Walt uploads a photo (GP) and assigns to it OSL_{GP} *=(L, P, {colleagues, family})*. This means that GP has a low sensitivity and it concerns Walt's colleagues, or family. Assume Walt assigned to his friend Javier an $FCL_{w,j}$ *=(H, {P, T, V}, {colleagues, university})*. This implies that Walt grants to Javier a high clearance level, allows him to see his photos, texts, and videos, and he considers him a member of the colleagues and university groups.

3.3 Access Control Decisions

Access to objects in an OSN can be related to one of the privileges enclosed in the privileges set (PS). To reflect this broader range of privileges, compared to traditional scenarios formalizing only read and write privileges, we refer to access requests as *interaction requests - (IR).* An IR can be of two types: 1) performed on an object (e.g., read or comment on an object), or 2) made to create an object related to another user (i.e., tag a user or write on her wall). We define an *interaction request* as follows:

Definition 4 (Interaction Request - IR) *Let $a, b \in U$ be two friends in the OSN (i.e., $\exists e_{a,b} \in FR$). An IR by user b on user a is denoted by the tuple $ir_{b,x} = (x,\ p,\ b)$, where $p \in PS$ is the requested privilege, and x has one of the following forms: $x = o$, where o is an object s.t. o.owner $= a$; **if** $p \in$ {read, add-like, add-comment}, or $x = o$, where o is an object s.t. o.owner $= a$ and o.parent $=$ null; **if** $p =$ share, or $x = a$, where a is the target user; **if** $p =$ write, or $x = (a, o)$, where a is the target user and o is an object; **if** $p =$add-tag.*

Users can explicitly issue a *read* IR on independent objects only. If such a *read* IR is positively evaluated, the system issues, on behalf of the requesting user, *read* IRs on all the children's tree of the granted independent object. This ensures that the children objects are evaluated independently of their parent object and only based on their proper OSLs. We explain this later under Algorithm 1.

Like in standard MAC, the evaluation of an IR is performed based on properties that define orders between subject and object labels. In standard MAC, there are two properties that govern the system, related to the *read* and to the *write* requests, respectively. LAMP requires not only the redefinition of these two properties to account for all the features in the FCLs and the OSLs, but it also requires defining new axioms to take into account the other interactions possible in an OSN.

An order relation among labels must be defined to have the basis on which they can be compared. In LAMP, we refer to this relation as the *dominance relationship*, formally defined as follows:

Definition 5 (Dominance relationship) *Let $a, b \in U$ be two friends in the OSN (i.e., $\exists e_{a,b} \in FR$) and let $FCL_{a,b}$=($CL_{a,b}$, $TS_{a,b}$, $GS_{a,b}$) be the label a assigned to b. Let o be an object s.t. o.owner $= a$, and let $OSL_o = (SL_o, TY_o, GS_o)$ be its OSL. $FCL_{a,b}$ dominates OSL_o, denoted as $FCL_{a,b} \gg OSL_o$, if an only if,*

$$CL_{a,b} > SL_o \wedge TY_o \in TS_{a,b} \wedge GS_o \cap GS_{a,b} \neq \emptyset$$

Example 3. Consider Example 2. The FCL that Walt assigned to Javier dominates the OSL assigned to GP photo because: 1) the clearance level of Javier (High) is higher than the sensitivity level of the photo (Low); 2) object type photo is in the types set in Javier's label; and 3) the group sets in Javier's and the photo's labels have a group in common (colleague).

The fundamental security property We are now ready to introduce the first property in LAMP, referred to as the *Fundamental Security Property (FSP)*:

Definition 6 (Fundamental Security Property) *Let $a, b \in U$ be two friends in the OSN (i.e., $\exists e_{a,b} \in FR$) and let $FCL_{a,b}$ be the label a assigned to b. Let o be an object s.t. o.owner = a, and let OSL_o be its label. Let $ir_{b,o} = (o, p, b)$ be an IR made by b on o s.t $p \in$ {read, share, add-like, add-comment}. $ir_{b,o}$ satisfies FSP, iff: $FCL_{a,b} \gg OSL_o$.*

Informally, the FSP dictates that to grant a *read*, *share*, *add-like*, or *add-comment* IR on an object, the FCL of the requestor must dominate the OSL of the object it targets. A requirement of the FSP is that the requestor and the object owner are friends in the OSN as otherwise the friend label to be compared would not exist. As a consequence, in LAMP, only friends of a user can have access to and interact on her information space. This is not fully aligned with the classical approach of OSNs where there is the possibility of making information available to friends-of-friends or public. Regarding the case of information desired to be made public (i.e., accessible to all users in the OSN), it can be easily enabled by making the object unclassified, and by letting the system assign a default label $FCL_{default}$=(UC, GS, OTS) to all OSN users who are not direct friends with the object owner, where GS, and OTS are the sets of all groups and of all object types in the OSN, respectively. For other information that is not made publicly available, our approach is to provide users with an environment that facilitates the understanding and control of their objects audiences. This is ensured by the suggested labels assignment, as users can be fully aware of (and track) who might gain access to what, contrary to the approach of allowing the unlimited and uncontrollable friend-of-friend access. However, LAMP can still be extended to cover such a scenario by complementing it with a mechanism for automatic FCLs assignment to friends of friends. This issue is a research subject in itself, that we plan to address as future work.

The FSP is sufficient for all the IRs it targets, except from the *share* one. This is what we address in the following sub-section.

The *share* privilege and the share up property When a *share* IR is granted, it results in the creation of a copy of the original object, that is owned by the sharing user. The problem here is on what should be the label of this shared copy. The simplest design idea is to make the copy inherit the same label as the original object, ensuring by this that only the audience of the original object is allowed to view the shared copy. This will map to what is being the standard in current OSNs, such as Facebook. However, this design choice results in two limitations that could not be desirable: 1) the user who performs the *share* interaction would have no way to influence the privacy settings of the shared copy; and 2) this strategy limits the intended purpose of the *share* functionality, since it does not result in an enlargement of the intended audience. Indeed, when a user allows a friend to share an object, this logically means that the user wants to delegate to her/him the dissemination of that object in order to make it available to a wider audience from the friend's side. Otherwise, if the user wants to limit the visibility of an object strictly to her allowed audience, then what could be the purpose from allowing a *share* interaction on it? Therefore, an alternative design decision, that would offer more flexibility and give meaning to the power of a *share* operation, is to have the shared copy have its distinct object label. However, the copy's label should still respect the privacy of the original object in such a way that the copy might be available to a wider audience without violating the privacy of the original object. Herein, the focus is on two concerns: 1) the shared copy *so* of an object *o* should not be made available to the friends of *o.owner* who are initially not allowed to access *o*; 2) the delegation offered by *o.owner* to the friend who creates the shared copy *so* should respect the sensitivity of *o*, such that *so* is shared only with those friends of *so.owner* who are at least as trusted as the sensitivity of *o*. We present Example 4 for a better illustration of these two concerns.

Example 4. Consider Example 2, and assume that Walt and Javier have Mina as a common friend in the OSN. Let Walt's FCL for Mina be $FCL_{w,m}$ *=(VL, {T}, {university})*. Recall that the OSL of Walt's photo is OSL_{GP} *=(L, P, {colleagues, family, university})*. Clearly, Mina does not have access to Walt's photo as $FCL_{w,m}$ does not dominate OSL_{GP}. Assume now that Walt allows Javier to share the photo; hence delegates to him its broader dissemination. This delegation means that Walt entrusts Javier to disseminate the picture to those of his friends that he trusts at least as high as the sensitivity of the photo (i.e., Walt's picture has a Low sensitivity, therefore it should be disseminated to those friends of Javier whom he trusts with at least a Low value). Furthermore, the shared photo by Javier should not be made available to Mina, no matter how Javier trust's in Mina could be different from Walt's.

To address these two concerns, we first introduce an additional axiom, called the *Share Higher Property*. Before formally defining it, we introduce a new required concept that governs the relationship between the label of an object and that of its copies made from a granted *share* IR. We refer to this as the *Not-declassify relationship*:

Definition 7 (Not-declassify relationship) *Let* $OSL_o = (SL_o, TY_o, GS_o)$ *be the label of an object o. Let* $OSL_{\overline{o}} = (SL_{\overline{o}}, TY_{\overline{o}}, GS_{\overline{o}})$ *be the label assigned to a copy* $\overline{o}$ *of o (i.e.,* $\overline{o}$*.copyof = o). Label* $OSL_{\overline{o}}$ *does not declassify label* OSL_o*, and we write* $OSL_{\overline{o}} \not\preceq OSL_o$*, if and only if:* $SL_{\overline{o}} \geq SL_o$*.*

The Not-declassify relationship requires that the sensitivity level of an object's copy is at least equal to that of the original object. This would ensure that a share action does not declassify the shared information, with the assumption of a delegated trust from the original object's owner to the sharing friend. It is also to be noted that no restriction is made on the groups set parameter of the label. This is because the organization of friends into groups is dependent on each user without necessarily mapping to the groupings adopted by their friends. For example, those who might be family to a user, might be colleagues to another. However, it is worth mentioning that the purpose of the share action is to disseminate objects to wider and different audiences, as long as this does not result in explicitly making the information available to a non-desired friend.

Based on the Not-declassify relationship, the second axiom of LAMP, the *Share Higher Property*, is defined as follows:

Definition 8 (Share Higher Property - SHP) *Let $a, b \in U$ be two friends in the OSN (i.e., $\exists e_{a,b} \in FR$), and let o be an object s.t. $o.owner = a$. Let OSL_o be the label a assigned to o. Let $ir_{b,o} = (o, share, b)$ be a share IR made by b on o, and let $OSL_{\overline{o}}$ be the object label that b intends to assign to the copy of o. We say that $ir_{b,o}$ satisfies SHP, if and only if: $OSL_{\overline{o}} \npreceq OSL_o$.*

Informally, the SHP enforces that users can share their friends' objects only if they assign to the shared copy an OSL that does not declassify the one of the original object. We recall that a *share* IR should first satisfy the FSP. Therefore, only those friends who have a clearance level at least equal to the sensitivity level of the object can perform a *share* IR. This ensures respecting the delegation of objects dissemination to only those friends who are at least as trusted as the sensitivity of the object they are entrusted to share.

The SHP addresses the delegation of dissemination concern, however it does not solve the first concern we discussed earlier related to when an object owner and the sharing friend have a friend in common who is not initially allowed to access the object by the owner (the case of Mina in Example 4). To prevent such *horizontal disclosures*, we enforce the preferences of objects' owners by considering their labels and not the copy's when the requestor, the sharer, and the owner are mutual friends. For the scenario in Example 4, for instance, if Mina issues a *read* IR on the shared copy of Walt's picture from the common friend Javier, the labels assigned by Walt for the original picture and for Mina are used to evaluate the IR, and not the ones assigned by Javier to the shared copy and to Mina. This is enforced by Algorithm 1 as it will be presented later.

The *add-tag* & *write* privileges and the write higher property Both the *add-tag* and the *write* interactions, if granted, result in the creation of new objects, and therefore they bring to the problem of what the labels for these resulting objects should be. For instance, assume Bob has a very high level of trust in Alice and allows her to post on his wall. Alice might hold sensitive information about Bob and, as such, her posts on his wall might reflect some of it. Thus, Bob would want that Alice's posts on his wall be managed with a label that reflects their expected sensitivity; that is, a label that is at least as high as the one he assigned to Alice. On the other hand, if Bob's trust on another user, say Aliah, is low, then he might need to impose more control on her posts on his wall as it could be that she holds some sensitive information about him that she might post to embarrass him, for example. The same applies to the *add-tag* interaction as well. To cope with this, we suggest enforcing two conditions on the labels that requestors of granted *write* or *add-tag* IRs can assign to the resulting objects. The first condition imposes a sensitivity level for the created object that is at least as high as the clearance level that the affected user assigned to the requestor, if this latter is higher than the medium level. For example, if Bob assigns a high clearance level to Alice, Alice's posts on his wall will have at least a high sensitivity level. However, if the clearance level assigned by the affected user to the requestor is lower than the medium level, its inverse is set as the least requirement for the sensitivity level of the resulting object. For instance, if Bob assigns to Aliah a low clearance level, Aliah's posts on his wall will have a sensitivity level at least equal to the inverse of Aliah's clearance level (i.e., a high sensitivity level).

The second condition imposes that the group set of the resulting object's label is exactly the group set that the affected user specified in her label for the requestor. For example, if Bob assigned the group "colleagues" to Alice, then it is likely that Alice's posts about him will also concern the "colleagues" group. To formalize these conditions, we introduce the third axiom of LAMP, referred to as the *Write Higher Property - (WHT)*:

Definition 9 (Write Higher Property - WHT) *Let $a, b \in U$ be two friends in the OSN (i.e., $\exists e_{a,b} \in FR$) and let $FCL_{a,b}$=($CL_{a,b}$, $TS_{a,b}$, $GS_{a,b}$) be the label a assigned to b. Let $ir_{b,x} = (x, p, b)$ be an IR such that $p \in$ {write, add-tag} and $x = a$, if $p = write$, $x = (a, ob)$, if p = add-tag, with ob the object to be tagged. Let o be the object resulting from granting $ir_{b,x}$ and OSL_o be its assigned label by b. $ir_{b,x}$ satisfies WHT if and only if: $o.owner = a \wedge OSL_o = (SL_o, TY_o, GS_o)$ s.t., $GS_o = GS_{a,b}$,*

$$TY_o = \begin{cases} TG & \textit{if p=add-tag.} \\ FP & \textit{if p=write.} \end{cases}$$

and

$$SL_o \geq \begin{cases} CL_{a,b} & \textit{if } CL_{a,b} \geq M. \\ f^{-1}(CL_{a,b}) & \textit{otherwise.} \end{cases}$$

With $f^{-1}(y), y \in LOS$ being the inverse function for security levels. For example $f^{-1}(VH) = VL$, $f^{-1}(L) = H$, etc.

Example 5. Consider Example 2, and assume Javier has the right to post on Walt's wall and that he posts on it a video of Walt's graduation ceremony. We recall that Walt assigned to Javier *(H, {P, T, V}, {colleagues, university})*. By the WHT property, the video's owner is Walt, and a possible granted label that Javier can assign to it is: *(H, FP, {colleagues, university})*. This means that the video will be available only to the friends of Walt to whom he assigned a high or a very high clearance level, who belong to the colleagues or to the university groups, and who are allowed to see Walt's friends' posts.

Access control enforcement Putting it up all together, Algorithm 1 defines how access control is enforced in the system based on the privacy preferences of users, as expressed through FCLs and OSLs and on the properties defined above.

Algorithm 1 takes as input an IR and produces as output a *granted* or *denied* message, based on the corresponding properties to the requested privilege. For a *read* IR, the algorithm first checks if the target object is a copy. If it is, the *NoHorDisclosure* function is called (lines 3,4) to rewrite the IR as targeting the original object, as long as the involved parties are mutual friends (lines 34,39). It is then checked if the retrieved IR satisfies the FSP. If it does, a similar IR is propagated through the children of the target object (i.e., calling the *propagate* procedure in line 6). The procedure traverses the object's tree in a depth first manner, evaluating each child against the FSP. If satisfied, *granted* is returned and the depth traversal continues. Otherwise, *denied* is returned (lines 39-46) and the traversal moves to the next breadth child. In case the input IR does not satisfy the FSP, *denied* is returned (line 8).

For a *write* IR, it is first checked if the requestor has a *write* privilege on the targeted user's wall, by evaluating a *read* IR on the *root* object of the targeted user (lines 10,11). If this IR satisfies the FSP and the input *write* IR satisfies the WHP, this latter is *granted*, otherwis *denied* is returned (lines 12-15).

In case the input IR is a *share* one (line 16), it is checked if it satisfies both the FSP and the SHP sending *granted* if it does, or *denied* in the failing case.

As for *add-tag* IRs, a *read* IR on the object to be tagged is formulated by the system (line 22). Then it is checked if this system IR satisfies the FSP and if the input *add-tag* IR satisfies the WHP. If the two conditions are satisfied, the input IR is *granted*, otherwise it is *denied*.

Finally, for *add-like* and *add-comment* IRs, they are granted only if they satisfy the FSP and if a system issued *read* IR on the object they target (line 29) is granted (i.e., it satisfies the FSP too) (lines 30-33).

4 Correctness Property

To formalize the correctness property of the system, we define the concept of its *state* as the set, $SIR = \{ir_1, ir_2, ..ir_n\}$, of interaction requests currently granted in the system. We say that the state of the system is correct, and we refer to it as the *correct state*, if and only if $\forall ir_i \in SIR, ir_i$ satisfies all the model's properties. The system changes its state only when a new IR processed by Algorithm 1 returns a granted message. Given a *correct state*, the following holds:[5]

Theorem 1. *Let SIR be the current state of the system. Let ir_{new} be a new interaction request issued to the system. Algorithm 1 issues a granted message if and only if $SIR_{new} = SIR \cup \{ir_{new}\}$ is a correct state.*

We note that changes to labels, either FCLs or OSLs, by a user does not affect the stability of the system, as they concern how the user's data flows to her friends only. That is, if Alice changes the OSL of her photo, this will only result in making it available to a larger audience from her friends, as will be imposed by Algorithm 1. Likewise, changing the FCL of her friend Bob, will result in changing the set of her available objects to Bob, for which the access will always be controlled by Algorithm 1.

[5] Related proofs are reported in [1].

Algorithm 1 **Access control enforcement**

```
Input: An IR, $ir = (x, p, b)$
Output: $granted$ or $denied$
Switch $p$ do
    Case $read$                                                    ▷ x is an object
        if $x.copyof \neq null$ then
            $ir \leftarrow$ NoHorDisclosure($ir$);
        if SatisfyFSP($ir$) then
            propagate($ir$);
        else
            send $denied$;
    Case $write$                                                ▷ x is a target user
        $root_x \leftarrow$ GetRootOf($x$);
        $ir_{allowed} \leftarrow (root_x, read, b)$;
        if SatisfyFSP($ir_{allowed}$) $\wedge$ SatisfyWHP($ir$) then
            send $granted$;
        else
            send $denied$;
    Case $share$                                                   ▷ x is an object
        if SatisfyFSP($ir$) $\wedge$ SatisfySHP($ir$) then
            send $granted$;
        else
            send $denied$;
    Case add-tag                                          ▷ x is a pair (usr, obj)
        $ir_{allowed} \leftarrow (obj, read, b)$;
        $ir \leftarrow (usr, p, b)$;
        if SatisfyFSP($ir_{allowed}$) $\wedge$ SatisfyWHP($ir$) then
            send $granted$;
        else
            send $denied$;
    Case add-like $\vee$ add-comment                               ▷ x is an object
        $ir_{read} \leftarrow (x, read, b)$;
        if SatisfyFSP($ir_{read}$) $\wedge$ SatisfyFSP($ir$) then
            send $granted$;
        else
            send $denied$;

function NoHorDisclosure($ir_{a,o}$)
    $IR \leftarrow ir_{a,o}$;
    while $o.copyof \neq null$ do
        if isFriend(a, o.copyof.owner) then
            $IR \leftarrow ir_{a,o.copyof}$;         ▷ rewrite the IR to target the copied object
        $ir_{a,o} \leftarrow ir_{a,o.copyof}$;
    return $IR$;

procedure propagate($ir_{a,o}$)
    send $granted$;
    while $ch \leftarrow o.nextChild \neq null$ do
        $IR \leftarrow (ch, read, a)$;
        if SatisfyFSP($IR$) then
            propagate($IR$);                         ▷ propagate to next in-depth child
        else
            send $denied$;                  ▷ deny and iterate to next in-breadth child
```

5 Performance Experiments

In our experiments, we study the performance of LAMP under access scenarios simulated on a real OSN graph. We have implemented a prototype of LAMP as a web application that we interfaced with Facebook graph API. The prototype has been implemented using PHP and the MySQL DBMS. We have been running a usability study, for which preliminary satisfactory results have been obtained. For space limitations, we report these in a technical report [1]. Our experiments have been conducted on a standard PC with a dual core processor of 2.99 GHZ each and 8 GB of RAM.

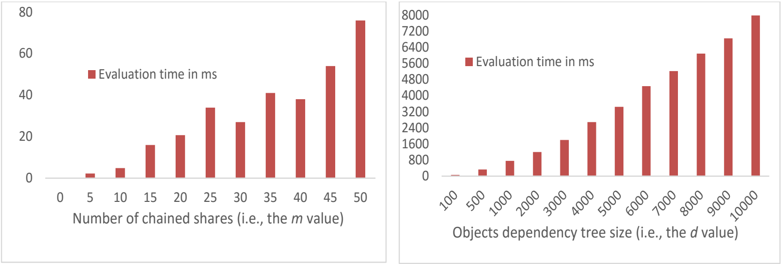
For performance testing, we exploited the public dataset available for the Slovakian OSN, Pokec.[6] The dataset represents a friendship graph of 1.63 million nodes and 30.6 million edges with a diameter equal to 11; however, it does not contain any data object. As we discuss in [1], the performance is regulated by the number of children in an object dependency tree (i.e., d), and by the number of chained shares an object is subject to (i.e., m). For this, we performed two experiments, by varying both m and d independently. As such, we first simulated share operations along friendship paths of different lengths, and we computed the time required to evaluate an access request on each of the shared copies. Figure 3.a reports the achieved results. The x-axis refers to the number of chained shares an object has been subject to, whereas the y-axis refers to the average time in milliseconds required to evaluate an access request on the last shared copy. All the reported numbers are the result of averaging 40 individual instances. As we can see on the figure, the time required to evaluate an access request on a shared copy is directly proportional to the distance between the target shared copy and the original object it refers to (i.e., the number of chained shares). However, the case of a chained share of length 50 requires no more than 80 ms. We can also see that the evaluation time does not follow a perfect linear increase with the number of chained shares. This is because the friendship sub-graph surrounding the sharing path also affects the convergence of the evaluation, as the *NoHorDisclosure* function in Algorithm 1 makes recursive calls as long as a common friendship is found to be between the requestor and the owner of the copied object.

We have also simulated object dependency trees of different sizes (d) varying both their breadth and depth. We considered the worst case scenario whereby the access evaluation on all children of a tree is positive. Figure 3.b reports the achieved results by plotting the time in milliseconds (the y-axis) for tree sizes varying from 100 to 10K children.[7] For each reported tree size, the evaluation time has been averaged across 10 different trees by varying their breadth and depth sizes. For all considered trees, the breadth made at least 80% of the

[6] http://snap.stanford.edu/data/soc-pokec.html

[7] Based on a 2015 study on Facebook, the average likes for personal content is below 100. It is about 100K for business and fans pages: http://www.adweek.com/socialtimes/infographic-quintly-average-like-totals-pages-february-2015/617303

tree's size. This is because on Facebook for instance, and according to the same study mentioned above, most of the interactions are one level likes or one level comments.



a. Evaluation time (ms) for access requests on shared copies

b. Evaluation time (ms) by objects dependency tree size

Fig. 3: Performance results on LAMP

As it can be seen on the figure, the amount of time required to evaluate access requests on all the children in an objects dependency tree increases, as expected, almost linearly with the size of the tree. We can see that the evaluation time still remains below 8k ms for trees of 10k children in size.

6 Conclusion and Future Work

In this paper, we presented a label-based access control model for OSNs. The proposed LAMP model considers relationships between objects in an OSN and allows their control at an atomic ownership level, simplifying as such the complexity of addressing data co-ownership of integral elements over which conflictual access policies may be specified. LAMP ensures a flexible and rich set of privacy settings, and empowers data owners to have control over the objects they create at finer granularity levels (e.g., comment, like, etc.). Our method achieves practical performance results and it is also designed to fit within different types of OSNs. Indeed, although we assume an undirected graph model for OSNs, our proposed solution can naturally fit the directed graph model as well, as the FCLs are directional (i.e., $FCL_{A,B}$ and $FCL_{B,A}$ are independently governing the data flow from A to B and from B to A, respectively.)

We are aware of the importance of ensuring usability and friendliness to average OSN users. Our preliminary usability experiments results are satisfactory [1]; however, they remain limited in terms of the number of participants and their representation of average users, but most importantly in terms of the amount of time these participants have been using our app. However, we believe that our model provides a simpler solution compared to other proposals for the management of co-owned data. As such, the first planned extension of this work is on usability experiments that need to be enriched in terms of the representativeness of participants, the statistical relevance of usage time, and the completeness of the interactions available on the app. Moreover, we plan to design accompanying mechanisms for the automatic assignment of friend and object labels based on general policies that the users can set, or based on automated learning strategies.

References

1. Bahri, L., Carminati, B., Ferrari, E., Lucia, W.: Technical report: Lamp - label-based access control for more privacy in online social networks. https://drive.google.com/file/d/0B9hyKuTMyMBzeTUxbGZBWHlQSHc/view (2016)
2. Carminati, B., Ferrari, E., Perego, A.: Enforcing access control in web-based social networks. ACM Transactions on Information and System Security (TISSEC) (2009)
3. Caviglione, L., Coccoli, M., Merlo, A.: A taxonomy-based model of security and privacy in online social networks. International Journal of Computational Science and Engineering 9(4), 325–338 (2014)
4. Cheng, Y., Park, J., Sandhu, R.: Relationship-based access control for online social networks: Beyond user-to-user relationships. In: Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom). pp. 646–655. IEEE (2012)
5. Ferrari, E.: Access Control in Data Management Systems. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2010), http://dx.doi.org/10.2200/S00281ED1V01Y201005DTM004
6. Fong, P.W., Siahaan, I.: Relationship-based access control policies and their policy languages. In: Proceedings of the 16th ACM symposium on Access control models and technologies. pp. 51–60. ACM (2011)
7. Gao, H., Hu, J., Huang, T., Wang, J., Chen, Y.: Security issues in online social networks. Internet Computing, IEEE (2011)
8. Hu, H., Ahn, G.J., Jorgensen, J.: Multiparty access control for online social networks: model and mechanisms. Knowledge and Data Engineering, IEEE Transactions on 25(7), 1614–1627 (2013)
9. Hu, H., Ahn, G.J., Zhao, Z., Yang, D.: Game theoretic analysis of multiparty access control in online social networks. In: Proceedings of the 19th ACM symposium on Access control models and technologies. pp. 93–102. ACM (2014)
10. Madejski, M., Johnson, M.L., Bellovin, S.M.: The failure of online social network privacy settings. Columbia University Academic Commons (2011)
11. Masoumzadeh, A., Joshi, J.: Osnac: An ontology-based access control model for social networking systems. In: Social Computing (SocialCom), 2010 IEEE Second International Conference on. pp. 751–759. IEEE (2010)
12. Mehregan, P., Fong, P.W.: Design patterns for multiple stakeholders in social computing. In: Data and Applications Security and Privacy XXVIII, pp. 163–178. Springer (2014)
13. ORACLE: Label security administrator's guide. http://docs.oracle.com/cd/B19306_01/network.102/b14267/intro.htm, accessed: 2015-05-29
14. Pang, J., Zhang, Y.: A new access control scheme for facebook-style social networks. Computers & Security 54, 44–59 (2015)
15. Squicciarini, A.C., Xu, H., Zhang, X.L.: Cope: Enabling collaborative privacy management in online social networks. Journal of the American Society for Information Science and Technology 62(3), 521–534 (2011)
16. Such, J.M., Rovatsos, M.: Privacy policy negotiation in social media. arXiv preprint arXiv:1412.5278 (2014)