A Collaborative Access Control Framework for Online Social Networks

Hanaa Alshareef[a], Raúl Pardo[b], Gerardo Schneider[a], Pablo Picazo-Sanchez[a]

[a] *Chalmers | University of Gothenburg. Gothenburg, Sweden*
[b] *Univ Lyon, Inria, INSA Lyon, CITI, F-69621 Villeurbanne, France*

Abstract

Most Online Social Networks allow users to set their privacy settings concerning posting information, but current implementations do not allow a fine grained enforcement in case the posted item concerns other users. In this paper we propose a new collaborative access control framework that takes into account the relation of multiple users for viewing as well as for sharing items, eventually solving conflicts in the privacy settings of the users involved. Our solution relies on two algorithms, one for viewing and another one for sharing items. We provide an evaluation of these algorithms where we demonstrate how varying some of the parameters directly influences the decision of viewing or sharing an item. Last but not least, we present a proof-of-concept implementation of our approach in an open source social network called Diaspora.

Keywords: Collaborative Access Control, Privacy, Social Networks.

1. Introduction

Most Online Social Networks (OSNs) today have privacy settings that allow users to define their preferences in what concerns the use of their data. This usually includes aspects related to whom can have access to which information but it is limited in a number of ways. For example, in OSNs like Facebook or Twitter users may only describe who is the direct audience of a given item (post, message, picture, etc.), meaning that it only concerns who has access to the item based on the explicit relationships the user has previously defined. In many cases it involves only one level in the relationship order or two levels, e.g., friends or friends of friends. This is a limitation since users might be interested in defining their privacy settings in order to limit the access of other users not directly connected with them beyond two levels. This is the case, for instance, whenever somebody who originally got access to the information, wants to share it with other users unrelated to the original source of the item.

In the majority of OSNs, the audience of a piece of information uploaded to the system is solely defined by a single user. Typically, the user defining the audience is the one uploading the data

Email addresses: hanaa@chalmers.se (Hanaa Alshareef), raul.pardo-jimenez@inria.fr (Raúl Pardo), gersch@chalmers.se (Gerardo Schneider), pablop@chalmers.se (Pablo Picazo-Sanchez)

to her own *space*, or somewhere else.[1] However, many other users may also be concerned with the posted data, so they should also have a say in who may access or not. Ideally, there should be a mechanism allowing all the involved users to take a decision collaboratively.

Current implementations of social networks rely on the so-called Relationship-based Access Control (ReBAC) model [1] where the social relationships between users are used to express access control policies. Though ReBAC has been shown to have many advantages with respect to other access control models in OSNs [1, 2], it does not allow a fine grained enforcement in case the posted item concerns many users, and the privacy settings usually do not allow for setting limits when a user wants to share the item she got access to. This lack of collaborative policies for access control may violate the privacy of the users who are part of the uploaded content, since they cannot decide who should access it: only the uploader of the data can decide that.

Additionally, apart from the aforementioned problem, ReBAC does not properly address users' policy conflicts.[2] It is possible that, within an access control policy, a user is permitted and denied access at the same time, thus creating a conflict in the policy. Thus, there is a need to solve the conflicts before deciding who has access to the shared object [3, 4, 5].

A promising line of work for collaborative access control is the so-called *aggregation-based models* [5]. Using this approach the individual privacy preferences of all users related to an item are aggregated to decide, for instance, whether the item can be accessed and shared. However, the main drawback is that existing models (such as [6, 7]) are too coarse grained to cover all cases, and, in some cases, rely again on the data owner to choose a conflict resolution strategy.

We propose a *Viewing* and a *Sharing* aggregation-based algorithms which take a decision by solving potential conflicts between the different privacy settings of all the concerned users. Our algorithms rely on four different components: the sensitivity level of the users with respect to the concerned item, a trust relationship between users, and different weights for both the *controller types* and *accessor types*.[3] In order to be as general as possible, trying to cover most of the existing OSNs nowadays, we include in our model *factors* to give (or take) importance to some components. This gives us the possibility to tune the decision policy, getting the outcome to range from very conservative (e.g., a strong denial from one party may overrule all the others) to more liberal (e.g., a majority granting access impose their decision). Thus, during the system set-up, the administrator can give different values to such factors to model her desired OSN.

We evaluate our proposed algorithms by generating all possible combinations of the components under consideration (for a given value of the factors). We carry out some experiments to show how the components influence the decision on who should, or should not, access or share the posted items. Additionally, we provide a proof-of-concept implementation into the open source OSN Diaspora [8].

In summary, our **contributions** are:

- A collaborative access control framework for OSNs taking into account: a trust relationship

[1] In this work, we use the term *space* to refer both to the user's profile and her interactive arena. A user *profile* is a collection of settings and information associated with the user. It may be defined to be the explicit digital representation of the user's identity in the context of the given (OSN) environment. It includes information such as age, location, and interests. The user *interactive arena* is the arena for both public interaction and communication with others (e.g., the wall in Facebook, the Home timeline in Twitter, stream in Diaspora, etc.).

[2] In its standard meaning a privacy policy is understood as a statement by an organization on how personal data collected from individuals should be handled during their storage and processing. In our paper we use the expressions "privacy policy" and "privacy setting" interchangeably making the difference only when confusing may arise.

[3] *Controller* and *accessor types* will be defined in Section 2; for the time being it suffices to know that they represent all the different users concerned with the item under consideration.

between users, sensitivity level of the users with respect to the concerned item, and different weights for both the controller types and accessor types (Section 2);

- An algorithm for collaboratively deciding who has access to an item (Section 2.3.1), and an algorithm to take such decision in case of sharing the item (Section 2.3.2);
- An evaluation of the behaviour of our algorithms based on an analysis of how the different components affect the decision to grant or deny access and sharing (Section 2.5);
- A proof of concept implementation of our framework in Diaspora (Section 3).

We compare our approach with previous work in Section 4, and we conclude in the last section.

2. A Collaborative Access Control Framework for OSNs

Our framework consists of three components: 1) an OSN model (Section 2.1); 2) access control policies (Section 2.2), and; 3) the collaborative access control mechanism (Section 2.3).

2.1. OSN Model

OSNs are typically structured as graphs, where vertices represent users and items whereas the edges of the graph represent connections between nodes. Concretely, vertices in the model are split into *actors*, *items*, and *groups*. Actors represent the real users of OSNs.[4] Each actor has a *space*, which includes the user's profile and interactive arena. An *item* is a digital representation of the physical object (e.g.,picture, text) to be posted, shared, etc. A *group* represents a spot that connects a collection of users who have the same beliefs, interests, behaviours, etc. We use *Relationship types* to represent connections between vertices in the graph. In what follows we formally describe the OSN model that we use throughout this paper.

Definition 1 (OSN Model). *Let $\mathcal{A}$ be a set of actors, $\mathcal{I}$ a set of items and $\mathcal{G}$ a set of groups. Consider also a set of relationship types $\mathcal{RT}$. An* OSN model *is a graph $SG = (\mathcal{A} \cup \mathcal{I} \cup \mathcal{G}, \{\mathcal{R}_i\}_{i \in \mathcal{RT}})$ where the vertices in the graph are elements of one of the sets $\mathcal{A}$, $\mathcal{I}$ or $\mathcal{G}$, and each $\mathcal{R}_i \subseteq (\mathcal{A} \cup \mathcal{I} \cup \mathcal{G}) \times (\mathcal{A} \cup \mathcal{I} \cup \mathcal{G})$ is a binary relation representing the edges of the graph.*

Figure 1 shows an example of an OSN. In this example, there are four actors, a post (p) and a group (g). The post is uploaded by Alice who *mentions* her family members Bob and Carol. Also, the group g was created by Carol, and Alice is a member of it. As it can be seen, the social network has relationships between actors and the item, e.g., *owner* (Alice *owns* the post p) and *mentioned* (Bob and Carol are mentioned in the post p); relationships between actors and groups, e.g., *owner* (Carol *owns* the group g) and *member* (Alice is a member of the group g); and relationships between actors[5], e.g., *family* (both Bob and Carol are in a family relationship with Alice) and *friend* (David is a friend of both Carol and Alice).

Trust Model. Trust becomes a crucial concept in OSNs for improving privacy mechanisms and reducing concerns about disclosing personal information [9]. Different techniques have been proposed

[4]For us it is not important whether the user is a physical individual, or an institution or corporation.
[5]For simplicity, in Figure 1 we use lines without arrows to represent symmetric connections.

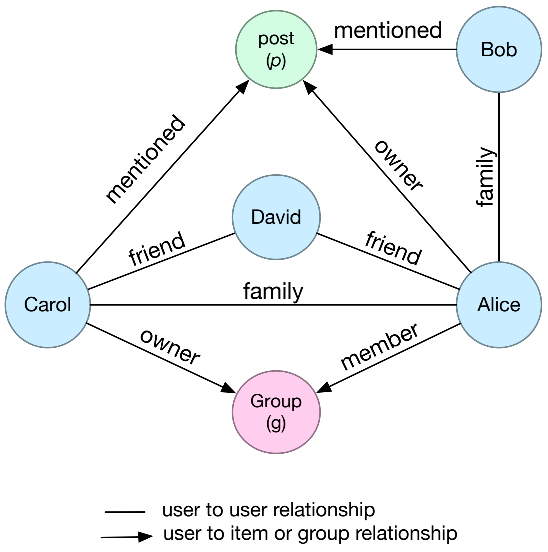


Figure 1: A Sample OSN Model

in the literature (e.g., [10, 11, 12, 13]) to determine the optimum path and the trust value among users in OSNs. In real life, trust is often expressed as a gradual phenomenon, meaning that people do not solely reason in terms of "trusting" and "distrusting", but rather trusting someone "very high" or "low". Fuzzy logic seems very well-suited for modelling such expressions which represent vague intervals rather than precise numerical values[10]. For instance, the *FuzzyTrust* algorithm [10] employs fuzzy linguistic terms to state trust levels and infers trust relationships among users. Compared to a purely numerical view of trust (e.g., [12, 13]), linguistic expressions are much easier to understand. FuzzyTrust requires a *trust graph* where edges are labelled with the following set of *Trust Linguistic Terms* ($\mathcal{TLT}$): $\{none, low, medium, high, highest\}$.

We assign to each of the elements of $\mathcal{TLT}$ a numerical value (see Table 1), used in our algorithms (Section 2.3). Hence, we define a trust graph as $TG = (\mathcal{A},\mathcal{TR})$ where vertices are actors ($\mathcal{A}$), which represent the OSN users, and edges $\mathcal{TR} \subseteq \mathcal{A} \times \mathcal{TLT} \times \mathcal{A}$ are defined as a set of triples (a, tv, v), indicating that a's trust level for v is equal to tv. The function $infer : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ computes the numerical trust value between two actors in a trust graph (note that $(1 - infer)$ corresponds to the *distrust value*). Given a trust graph TG, we use the notation $TG.infer$ to retrieve trust values among the actors in TG. When actors are directly connected in the trust graph, *infer* simply returns the numerical value corresponding to the label between the actors. On the contrary, if actors are not directly connected, *infer* applies the FuzzyTrust algorithm (see [10] for details). For instance, Alice may assign a high trust level for all her friends and a medium trust level to all her family except Bob who is assigned to a low level. The *infer* function computes the numerical value for trust between Alice and each of her friends, her family and Bob as following: 0.75, 0.50 and 0.25,respectively.

Associated Controllers. Similarly to [6], for each item we consider a set of *associated controllers*, representing all the actors of the OSN concerned in the decision of who should have access to a given

Table 1: Trust level weights

Linguistic term	Numerical value
none	0
low	0.25
medium	0.50
high	0.75
highest	1

item. We define a set of *controllers types* $\mathcal{CT} \subseteq \mathcal{RT}$ as $\{owner, stakeholder, contributor, originator\}$. Though the elements in $\mathcal{RT}$ may differ depending on the concrete OSN, we require that the elements in $\mathcal{CT}$ are always included in $\mathcal{RT}$. We define the set of *associated controllers* to item i as $C_i = \{c \mid (c, i) \in \mathcal{R}_j \text{ where } c \in \mathcal{A}, i \in \mathcal{I} \textit{ and } j \in \mathcal{CT}\}$ described in what follows.

Owner. All the items in the actor space are *owned* by that actor. We say that the actor is the *owner* of all those items. In our model we include an owner relation between an actor and an item every time an item is posted in her space. We use the relationship *owner* to indicate when an actor is the owner of an item.

Stakeholder. A *stakeholder* is an actor who is tagged or mentioned in an item. We use the relationship *stakeholder* to indicate when an actor is a stakeholder of an item.

Contributor. A *contributor* is an actor who posts an item in a space different than hers, e.g., Alice posting in Bob's space, making Alice to be a contributor for that post and Bob is the owner. We use the relationship *contributor* to indicate when an actor is a contributor of an item.

Originator. An actor is considered to be an *originator* when an item is shared from her space. For instance, if Alice shares an item from Bob's space to her space, then Bob is the originator of the item and she is the owner. For the item in Alice's space, if David shares the item from Alice's space to his space, then Alice is the originator of the item and he is the owner. We use the relationship *originator* to indicate when an actor is an originator of an item.

Note that for each item there is exactly one owner, at most one contributor, at most one originator, and zero or more stakeholders.

Controllers Types Weight. In our framework the associated controllers do not necessarily have the same importance. We use the principle that close people tend to be similar [14, 15], and thus we give the stakeholders the same importance as the owner for two reasons: 1) they are explicitly mentioned (or tagged) without their approval; and 2) stakeholders who are conventionally related to the content of the item, by one way or another, have the right to manage the disclosure of their sensitive information.

Regarding the contributor and the originator controllers, and contrarily to other proposals (e.g., [16, 17, 18]), we get them involved into the collaborative decision. The main difference with respect to the owner and stakeholder(s) is that we take into account the distance between them and the owner in the OSN model, and we give them different values. The *distance* between the owner and

Table 2: Controllers types' weights. Column 1 represents the types of associated controllers; column 2 shows the algorithms for producing the collaborative decisions; column 3 is the minimum distance among all the social relationships between the owner and the associated controller; column 4 shows the weight we assign to the associated controllers

Controller type	Algorithm	Distance	Weight
Owner	Viewing and Sharing	-	1
Stakeholder	Viewing and Sharing	-	1
Contributor	Viewing and Sharing	1	0.50
Contributor	Viewing and Sharing	≥ 2	0.25
Originator	Viewing	1	0.50
Originator	Viewing	≥ 2	0.25
Originator	Sharing	-	$\begin{cases} 0.25 \text{ if } TG.infer(originator, owner) \geq 0.75 \\ 0.75 \text{ otherwise} \end{cases}$

each associated controller is defined as the shortest path among all the relationships that connect them.

We propose two algorithms, *Viewing* and *Sharing*, corresponding to the different *actions* that an actor (associated controller) may do concerning an item. The algorithms depend on several parameters, among others the *weight* of the associated controllers.

Each associated controller is weighted based on whether she is involved in the process of making the collaborative decision regarding viewing or sharing an item, and her distance from the owner. Table 2 shows how this weight is defined for each associated controller in both algorithms. Note that for the owner and stakeholder types, the weight is always one and it does not depend on the distance since both are equally involved in the process of making a collaborative decision. In the case of the contributor, she is weighted differently based on her distance from the owner in both algorithms (see rows 3 and 4).

For the originator, the weight is different for each algorithm: for Viewing the distance is calculated in the same way as for the contributor, whereas for Sharing the distance is not relevant as she is weighted based on the trust level between herself and the owner. The trust level is used to indicate how much influence an originator's opinion will have on the aggregated decision. In this case, if the originator highly trusts the owner (with a value ≥ 0.75) then the weight is only 0.25, otherwise it will be 0.75. In other words, when the originator highly trusts the owner, the task of deciding whether to share or not is delegated to the trustee.

In what follows, we use a function $wct : \mathcal{CT} \rightarrow \mathbb{R}$ to retrieve the weight of a controller type.

2.2. Access Control Policies

In this section, we introduce the access control policies that our collaborative algorithms use. Before explaining what an access control policy is and how it is represented, we introduce two concepts: accessor specifications and sensitivity levels of data items.

Accessor Specification. In our framework each associated controller can identify a set of actors who can access her data and who cannot, the so-called *accessors*. Associated controllers can specify their permitted and denied accessors using the following accessor types: *actor names*, *group names* and *relationship names*. Formally, we define the set of accessor types as $\mathcal{AT} = \{an, gn, rn\}$. Many OSNs allow users to specify who can access their information using these accessor types; they have

Table 3: Accessor types weights

Accessor Type	Numerical value
an	1
gn	0.75
rn	0.50

also been used in other collaborative access control frameworks like in [6]. These accessor types help the controllers to customize their access control policies.

Definition 2 (Accessor specification). *We define an accessor specification as pair* $(\mathcal{A} \cup \mathcal{G} \cup \mathcal{RT}) \times \mathcal{AT}$*. We use* $\mathcal{AS}$ *to denote the set of accessor specifcations.*

For example, the accessor specification $\langle Alice, an \rangle$ indicates that Alice is specified as actor, while $\langle friends, rn \rangle$ denotes that the controller determines her friends relationship either to access her data or not. We denote the universe of accessor specifications as, $\mathcal{U}_{\mathcal{AS}} \subseteq 2^{\mathcal{AS}}$. Accessor types are organized hierarchically forming a total order: $rn > gn > an$ where $y > x$ means that x is *more specific than* y (or that y is *more general than* x). As we define below, denying or permitting an actor by means of a policy that uses a specific accessor type contributes more to the final decision than policies using less specific accessor types.

Accessor Type Weight. In our framework, not all accessor types are equal. For example, we consider that directly denying an actor ("Alice is denied") should have a "stronger" effect on a collaborative decision than indirectly denying an actor because it belongs to a relationship ("My friends are denied", where Alice is one of the friends). Thus, we weight the accessor types based on the *most-specific-takes-precedence* principle [19, 20]. We define the function $wat : \mathcal{AT} \rightarrow \mathbb{R}$ to retrieve the weight of an accessor type, e.g., $wat(getAccessorType(a, i))$ where $getAccessorType(a, i) \in \{an, gn, rn\}$ according to the definition in Table 3.

Sensitivity levels of data items. The actor's space, relationships and items, embody the actor's data in OSNs. Using *sensitivity levels*, the associated controllers of an item indicate how much a disclosure of the item would harm them. In what follows we define a set of sensitivity levels that associated controllers can add in their access control policies (see Definition 3). Let $\mathcal{SL} = \{none, low, medium, high\}$ be the set of *sensitivity level* linguistic terms. The *sensitivity levels* are shown in Table 4: the linguistic terms, which are the inputs that are assigned by the associated controllers, correspond to numerical values. We use the function $wsl : \mathcal{SL} \rightarrow \mathbb{R}$ to acquire the numerical value associated to the sensitivity level linguistic term.

Access Control Policies. The associated controllers can define their privacy preferences, where the policy of each controller affects the collaborative decision of viewing and sharing an item. We define an access control policy as follows.

Definition 3. *An* access control policy *is a tuple* $\langle i, c, ct, sl, PER, DEN \rangle$ *where: i)* $i \in \mathcal{I}$ *is the item to which this policy applies; ii)* $c \in$ *is the associated controller who defines the policy over the considered item; iii)* $ct \in \mathcal{CT}$ *is the type of the associated controller—automatically extracted from the corresponding relation in the OSN model; iv)* $sl \in \mathcal{SL}$ *is the sensitivity level of the considered*

Table 4: Sensitivity levels

Linguistic term	Numerical value
none	0
low	0.25
medium	0.50
high	1

item; v) PER and DEN $\in \mathcal{U_{AS}}$ are two accessor specification sets indicating the actors permitted and denied to view the item, respectively.

We denote the universe of access control policies as $\mathcal{U}_{ACP} \subseteq 2^{\mathcal{I}\times\mathcal{A}\times\mathcal{CT}\times\mathcal{SL}\times\mathcal{U_{AS}}\times\mathcal{U_{AS}}}$. For every item $i \in \mathcal{I}$, we use $ACP_i \in \mathcal{U_{ACP}}$ to denote the set of access control policies of the associated controllers—note that each associated controller must specify exactly one policy per item. We denote the access control policy of each associated controller as acp_c, where $c \in \mathcal{A}$. Given ACP_i we use $ACP_i.acp_c.e$ to refer to an element e of the access control policy tuple. Let us illustrate an access control policy with an example as follows: "Alice performs a post (p) (she is the owner) and grants all actors who have a family relationship with her to view the p and denies all her friends to view her post p, with high sensitivity level". Such access control policy for that post p is expressed as: $acp = \langle p, Alice, owner, high, \{\langle family, rn\rangle\}, \{\langle friends, rn\rangle\}\rangle$.

Note that we provide support to explicitly specify permitted and denied actors. This feature is present in OSNs such as Facebook, where actors can, for instance, share an item with their friends, and additionally, explicitly exclude other actors. For instance, consider a policy defined by Bob with: $PER = \{\langle friends, rn\rangle\}$ and $DEN = \{\langle Alice, an\rangle\}$—assuming that the item is shared with Bob's friends and Alice is the actor to be excluded. Note that Alice may, or may not, be friend with Bob. The potential set of actors that may be granted viewing permission is the union of all the permitted actors of all policies for the item, formally, $\bigcup_{acp\in ACP_i} acp.PER$. Every time that an actor in DEN is included in the previous union, a conflict may arise, as it will be the case in the example if Alice is in the permitted set of other associated controller. (We describe the conflict resolution algorithm in Section 2.3).

Note that specifying only permitted actors and marking the other actors in the OSN as denied, or vice versa, is a strictly less expressive choice—in particular the policy above would not be possible to express. So, expressing such cases in our framework is possible, but it may be tedious—the denied and permitted sets may contain a large number of actors. In order to model this in a compact manner, we use a special element $\perp$ that can be included in DEN and represents "all actors not in PER". Likewise, we use the element $\top$ in PER to denote "all actors not in DEN". For instance, a policy with $PER = \{\langle Alice, an\rangle\}$ and $DEN = \{\perp\}$ means "Alice is permitted and anybody else is denied", and with $PER = \{\top\}$ and $DEN = \{\langle Alice, an\rangle\}$ means "Alice is denied and anybody else is permitted". Note that PER and DEN can be empty. Intuitively, polices where PER is not empty and DEN is empty specify only permitted actors, e.g., $PER = \{Alice\}$ and $DEN = \emptyset$ means Alice is permitted and nobody is denied. Note the difference with $DEN = \{\perp\}$ where everybody except for Alice is denied. Not specifying denied actors does not mean that everyone can access the item. In particular, in the example above, Alice is the only actor who may be permitted to access the item. Defining $DEN = \emptyset$ simply imposes no restrictions in the set of permitted actors that other associated controllers may allow (in their respective PER sets). For example, consider a policy with

$PER = \{Alice\}$ and $DEN = \emptyset$, and a different policy for the same item with $PER = \{Bob\}$ and $DEN = \emptyset$, then the audience of the item is *only* $\{Alice, Bob\}$ (see Section 2.3.1). The intuition behind policies with $PER = \emptyset$ and $DEN \neq \emptyset$ is the inverse of the previous explanation, i.e., they specify only denied actors and leave unspecified the permitted set of actors.

Normalization of Accessor Specifications. It is possible that, within a single access control policy, an actor is permitted and denied access at the same time, thus creating a conflict in the policy. These conflicts may arise explicitly or implicitly. An *explicit conflict* occurs when a concrete actor, group or relationship type is explicitly included in the permitted and denied accessors sets at the same time. *Implicit conflicts* may occur, for instance, if an actor appears explicitly in the permitted accessors set but the denied accessors set includes a group or relationship where the actor is a member of—e.g., if Alice is a friend of Bob and we have $PER = \{\langle friends, rn\rangle\}$ and $DEN = \{\langle Alice, an\rangle\}$. Also, when an actor is a member of two different groups that appear in the denied and permitted accessor sets, respectively—i.e., imagine that Alice belongs to the groups engineers and mathematicians, and we have $PER = \{\langle mathematicians, gn\rangle\}$ and $DEN = \{\langle engineers, gn\rangle\}$. Similarly, if an actor is a member of two different relationship types that appear as permitted and denied, a conflict occurs. Finally, if an actor belongs to a relationship type and a group that appear in different sets, it will cause a conflict, e.g., $PER = \{\langle mathematicians, gn\rangle\}$ and $DEN = \{\langle friends, rn\rangle\}$ would cause a conflict since Alice belongs to both.

To resolve explicit conflicts we check that the sets of permitted and denied accessors are mutually exclusive, i.e., $PER \cap DEN = \emptyset$.

Resolving implicit conflicts requires looking into the following cases: i) Actors permitted and denied at different hierarchical levels, and; ii) Actors permitted and denied in different groups or relationship types.

Before handling the specific kinds of implicit conflicts, we apply a pre-processing step where we replace the pairs of type group name, $\langle g, gn\rangle$, in the accessors specification sets PER and DEN with a pair for each member of the group $\langle m_1, gn\rangle$, $\langle m_2, gn\rangle, \ldots$ where $m_j \in \{m \mid (m, g) \in R_{member}\}$. Likewise, we replace relationship types pairs such as $\langle \mathcal{R}, rn\rangle$ with their members. As a result we obtain the multisets M_{PER} and M_{DEN}. We use multisets because it is necessary for our conflict resolution strategies to count how many times a pair appears. Note that the pairs in these sets have the type $\mathcal{A} \times \mathcal{AT}$ since we replaced every group and relationship type with their members. Therefore, we can now syntactically identify conflicts by checking the first element of the pairs.

In order to resolve conflicts between actors at different hierarchical levels we apply the *most-specific-takes-precedence* principle [20, 19]. It states that the accessor specification that is more specific should remain. For example, if Bob is in the permitted set as an actor an, and in the denied set because he belongs to a group gn, the strategy removes Bob from the denied set. Formally, we apply the following principle: "If $\langle a, at_x\rangle \in X$ and $\langle a, at_y\rangle \in Y$ and at_x *is more specific than* at_y, then $X := X \setminus \{\langle a, at_y\rangle\}$ where $X, Y \in \{M_{PER}, M_{DEN}\}$" where the operation $\setminus$ over multisets discards all occurrences of the elements to remove.

Once we apply the previous step, there might still exist conflicts among groups or relationship types at the same hierarchical level—e.g., if Alice belongs to the groups co-workers and family, and she has permitted one but not the other. To resolve this type of conflict, we apply the *many-takes-precedence* principle [20], i.e., the higher number of positive/negative policies prevail. Formally, given the multisets $X, Y \in \{M_{PER}, M_{DEN}\}$ this principle updates them as follows: $X = X \setminus \{\langle a, at\rangle\}$ if $count(\langle u, at\rangle, Y) > count(\langle a, at\rangle, X)$ where $at \in \{gn, rn\}$, and $count(e, S)$ returns the number of appearances of element e in a multiset S. Note that in the previous strategy we require that the number of elements in one set must be strictly greater than in the other.

Finally, there can still be conflicts if there is the same number of appearances in both multisets. To solve these conflicts we use the *denial-takes-precedence* principle [20]. It simply keeps the pair appearing in the denied accessors set and removes it from the permitted accessors set. Formally, given $at \in \{gn, rn\}$, $M_{PER} = M_{PER} \setminus \{\langle a, at \rangle\}$ if $\langle a, at \rangle \in (M_{PER} \cap M_{DEN})$.

After applying the previous steps in the described order, we add all the elements from M_{PER} to PER and M_{DEN} to DEN to remove any remaining duplicate pairs. The resulting PER and DEN sets are not in conflict. Hence, for the rest of the paper, we assume that the sets PER and DEN are conflict-free.

2.3. Collaborative Access Control

In this section, we introduce our collaborative access control algorithms. These algorithms correspond to two actions that users may perform in the OSN which can potentially involve the controller's types in our model: viewing and sharing. In a nutshell, viewing corresponds to the event of accessing (*view*) an existing data item. Sharing, on the other hand, consists in selecting an existing item and sharing a copy in another space.

2.3.1. Collaborative Access Control: Viewing

Here we present Algorithm 1, an algorithm that produces the list of actors that can view an item. It takes as input three parameters: 1) a set of access control policies ACP_i for the item i; 2) the set of associated controllers for this item C_i, and; 3) the trust graph (TG). As output, it returns a set of *viewers*, i.e., actors that can view item i. Before running the algorithm, we apply the function *normalize* on the first parameter, $normalize(ACP_i)$, to resolve internal conflicts within each individual policy as described in the previous section. In the algorithm, first we include the set of associated controllers C_i in the set of viewers. Thus, modeling that all associated controllers will always be able to view the item. Second, we use an external procedure named $generateAccessors(ACP_i)$ to create a set of all possible viewers from the access control policies (ACP_i) of the item. Concretely, $generateAccessors(ACP_i)$ computes the union of all the actors that appear in the sets of accessor specifications PER and DEN of the policies in ACP_i. Formally, $generateAccessors(ACP_i) =$

$$\{a \mid (a, at) \in acp.PER, acp \in ACP_i\} \cup \\ \{a \mid (a, at) \in acp.DEN, acp \in ACP_i\}$$

The set is created by adding the actors specified in the sets PER and DEN of each policy to ACP_i. The algorithm aggregates the weights of the controller types, accessor types, trust level and sensitivity level indicated in the access control policies defined by the associated controllers according to Equations (1) and (2) below. Concretely, we use Equation (1) for accessor specifications in the PER set, and Equation (2) for accessor specifications in the DEN set. Additionally, we include support for a *veto right* when the following conditions are satisfied: 1) there is a high sensitivity level for the item i ; 2) the accessor is specified by the most specific accessor type (i.e., an), and; 3) there is a "highest" distrust value between the associated controller and the accessor. If all these conditions are met, the accessor cannot access to the item regardless of other associated controllers' policies.

$$\texttt{decision\_permit} = \phi_{ct} \cdot wct(acp.ct) + \phi_{at} \cdot wat(getAccessorType(a, acp.i)) + \\ \phi_{tr} \cdot TG.infer(acp.c, a) + \phi_{sl} \cdot wsl(acp.sl) \quad (1)$$

Table 5: Factors Impact on Collaborative Access Control(CAC) Framework.

Components	Facebook	Diaspora	CAC Framework
Controllers Types	*owner*	*owner*	*owner*, *stakeholder*, *contributor*, *originator*
Acessors Types	*rn*, *gn*, *an*	*rn*	*rn*, *gn*, *an*
Sensitivity Levels	$\emptyset$	$\emptyset$	*none*, *low*, *medium*, *high*
Trust	$\emptyset$	$\emptyset$	*none*, *low*, *medium*, *high*, *highest*

$$\begin{aligned}\texttt{decision\_deny} = \phi_{ct} \cdot wct(acp.ct) + \phi_{at} \cdot wat(getAccessorType(a, acp.i)) + \\ \phi_{tr} \cdot (1 - TG.infer(acp.c, a)) + \phi_{sl} \cdot wsl(acp.sl)\end{aligned} \tag{2}$$

The equations depend, among other things, on four *factors*: $\phi_{ct}, \phi_{at}, \phi_{tr}, \phi_{sl}$ (where each $\phi_i \in [0,1]$) representing the importance we give to each one of the different components of the equation. In particular, ϕ_{ct} affects the weight of controller types, ϕ_{at} affects the weight of accessor types, ϕ_{tr} affects the trust and ϕ_{sl} affects the sensitivity level of the item. By introducing these factors, our framework can model other OSNs like Facebook and Diaspora (see Table 5). For instance, we can model Facebook's privacy settings by setting the accessor type (ϕ_{at}), the sensitivity level (ϕ_{sl}) and the trust (ϕ_{tr}) to 0 whereas the weight of all the associated controllers is 0 except for the owner whose weight is 1. In the rest of the paper we omit the factors in our examples for sake of simplifying the presentation.

Note that the `decision_permit` and `decision_deny` equations are present in the algorithm as part of the computation of the variable `decision`. Their value must be computed n times where n ranges from 1 (the owner must always exist) to len (the total amount of associated controllers involved in the decision). The final result is given as $\texttt{decision} = \sum_{n=1}^{len} \texttt{decision\_permit}_n - \texttt{decision\_deny}_n$. If $\texttt{decision} > 0$ then the accessor can access the item, otherwise she cannot. Note that by using our proposed collaborative access control framework, each associated controller of an item has the ability to affect the final decision.

Example 1. *Consider that Alice performs a post p and mentions her family members Bob and Carol. Alice is the owner whereas Bob and Carol are the stakeholders. Their access control policies are,* $ACP_p =$

$$\begin{aligned}\{\langle p, Alice, owner, low, \{\langle family, rn\rangle\}, \{\langle friends, rn\rangle\}\rangle, \\ \langle p, Bob, stakeholder, medium, \{\langle co\text{-}worker, rn\rangle\}, \emptyset\rangle, \\ \langle p, Carol, stakeholder, low, \{\langle friends, rn\rangle\}, \emptyset\rangle\}.\end{aligned}$$

Consider now an actor (David), who is a friend of Alice and Carol. In ACP_p *Alice denies her friends whereas Carol allows her friends. In this scenario, the positive and negative authorizations about David's access create a conflict. The permitted decision value (*`decision_permit`*) is aggregated from Carol's acp which affects David's access as he is a friend of her. Carol has a low sensitivity level (i.e.,0.25) for the post p. On the other hand, the algorithm computes the value of a denied decision from acp of Alice which affects David's access as he is her friend. Alice also has a low sensitivity level (i.e.,0.25) for the post p. Owner and stakeholder are weighted 1 as defined in Table 2. For the purpose of this example, let Alice define a trust value of 0.75 for David whereas Carol sets a trust value of 0.5 for David. According to these trust values and the associated controllers' privacy policies, the value of* $\texttt{decision\_deny} = 2$ *based on Equation* (2)*, where* $wct(acp.owner) = 1$*,*

```
input  : $ACP_i$, $C_i$ and $TG$
output: viewers
viewers $\leftarrow C_i$
set_accessors $\leftarrow$ $generateAccessors(ACP_i)$
foreach $a \in$ set_accessors do
    decision $\leftarrow 0$
    foreach $acp \in ACP_i$ do
        if $a \in acp.PER$ then
            decision $\leftarrow$ decision + $\phi_{ct} \cdot wct(acp.ct)+$
             $\phi_{at} \cdot wat(getAccessorType(a, acp.i))+$ $\phi_{tr} \cdot TG.infer(acp.c, a)+$
             $\phi_{sl} \cdot wsl(acp.sl)$
        end
        else if $a \in acp.DEN$ then
            if (
                $wat(getAccessorType(a, acp.i)) = 1$ and
                $wsl(acp.sl) = 1$ and
                $(1 - TG.infer(acp.c, a)) = 1$
            ) then
                decision $\leftarrow 0$
                break
            end
            else
                decision $\leftarrow$ decision $-$ $\phi_{ct} \cdot wct(acp.ct)+$
                 $\phi_{at} \cdot wat(getAccessorType(a, acp.i))+$ $\phi_{tr} \cdot (1 - TG.infer(acp.c, a))+$
                 $\phi_{sl} \cdot wsl(acp.sl)$
            end
        end
    end
    if decision $> 0$ then
        viewers $\leftarrow$ viewers $\cup \{a\}$
    end
end
```

Algorithm 1: Viewing

$wat(getAccessorType(David, acp.p))$= 0.50, $wsl(acp.sl)$=0.25 and $(1-TG.infer(acp.Alice, David))$ =0.25. The value of `decision_permit` *equals* 2.25 *based on Equation* (1)*, where $wct(acp.stakeholder)$ = 1, $wat(getAccessorType(David, acp.p))$= 0.50, $wsl(acp.sl)$=0.25 and $TG.infer(acp.Carol, David)$ =0.50. So, David will have access to view the post.*

2.3.2. Collaborative Access Control: Sharing

Our second algorithm produces a set of actors that can share an item which has been previously posted (see Algorithm 2). We call *disseminators* the actors that have the right to share an item. The event of sharing an item consists in copying an already posted item and placing it in the disseminator's space. Consequently, the shared item has exactly one owner, one originator and zero or more stakeholders. This algorithm, as opposed to Algorithm 1, includes two phases: 1) filtering viewers in potentially allowed disseminators and potentially denied disseminators, based on the trust that the associated controllers have for each viewer (specified sharing policies, Definition 4 below), and; 2) an aggregation-based method similar to that of Algorithm 1 to decide whether the conflicting viewer—i.e., a viewer permitted by some associated controllers and denied by other associated controllers—might become a disseminator.

Sharing Policies. For sharing, each associated controller specifies a trust threshold that determines how much the minimum value of trust between her and the viewer has to be in order to allow the sharing action.

Definition 4 (Sharing Policies). *We define a* sharing policy *as a tuple* $\langle i, c, trc \rangle$ *where: 1)* $i \in \mathcal{I}$ *is the item to which the policy applies; 2)* $c \in \mathcal{A}$ *is the associated controller who defines the sharing policy, and; 3)* $trc \in \mathbb{R}$ *is a numerical value specifying the trust threshold for which sharing the item* i *is permitted by* c*.*

For every item $i \in \mathcal{I}$, $SP_i \in \mathcal{U_{SP}}$ is the set of sharing policies (with $\mathcal{U_{SP}} \subseteq 2^{\mathcal{I} \times \mathcal{A} \times \mathbb{R}}$). The numerical value in *trc* is obtained from a $\mathcal{TLT}$ that the associated controller selects when defining the policy—Table 1 contains the equivalences between TLTs and their corresponding numerical value. Given a sharing policy *sp* we use *sp.e* to refer to an element *e* of the sharing policy tuple, e.g., *sp.trc* refers to the trust threshold of the sharing policy *sp*.

Algorithm 2 takes as input four parameters: 1) a set of access control policies ACP_i for the given item i; 2) the trust graph (TG); 3) a set of `viewers` (computed using Algorithm 1), and; 4) the sharing policies associated to the item i, SP_i. As output, it returns the set of actors that can share the item. In what follows we explain the two phases of the algorithm in detail.

Phase 1). As mentioned earlier, SP_i contains the sharing policies for item i. These policies determine the minimum value of trust between the associated controller and the actor who might share the item. We use an external procedure named *filterActors* (i,TG,SP_i,`viewers`) to split the potential disseminators into actors who do (`permit_sharing`) and do not (`deny_sharing`) fulfill the sharing policies SP_i set in advance by each associated controller. The pseudo-code of the procedure is shown below:

```
foreach $a \in$ viewers do
    foreach $sp \in SP_i$ do
        $tr \leftarrow TG.infer(sp.c, a)$;
        if $tr \geq sp.trc$ then
            permit_sharing $\leftarrow$ permit_sharing $\cup \{c\}$;
        else
            deny_sharing $\leftarrow$ deny_sharing $\cup \{c\}$;
        end
    end
end
```

Procedure filterActors(i,TG,SP_i,`viewers`)

Given an item i, the above procedure includes a viewer v in the set of potentially permitted disseminators (`permit_sharing`), if an associated controller c has specified a sharing policy *sp* that includes a trust level lower than the trust the associated controller defined for the viewer, i.e., $TG.infer(sp.c, a) \geq sp.trc$. Otherwise the actor is included in the set of potentially denied disseminators (`deny_sharing`). Note that, since `permit_sharing` and `deny_sharing` are sets, each viewer can appear at most once in each set.

Phase 2). When a conflict arises among the associated controllers to allow or refuse the sharing action of the item i, for all involved associated controllers we have to differentiate between two cases: 1) when the trust of the viewer who might share the item is equal or higher than the sharing threshold (Equation (3)); and 2) when the trust of the viewer who might share the item is lower than the sharing policy (Equation (4)).

$$\texttt{decision\_permit} = \phi_{ct} \cdot wct(acp.ct) + \phi_{sl} \cdot wsl(acp.sl) \tag{3}$$

$$\texttt{decision\_deny} = \phi_{ct} \cdot wct(acp.ct) + \phi_{sl} \cdot wsl(acp.sl) \tag{4}$$

Contrarily to the Viewing algorithm, both Equations (3) and (4) have only two components, i.e., the weight of the controllers types and the sensitivity level of the associated controllers with respect to the item to be shared. This is because the set of actors that can share an item already had privileges to access it (viewers). This difference directly impacts the structure of the decision formula in such a way that the trust is used as a filter to know in advance if an actor is willing to share an item (there is no reason to include the accessor's weight as there are no accessors involved in the algorithm). As for the first two equations, we also introduce factors, two in this case: ϕ_{ct} and ϕ_{sl} (where each $\phi_i \in [0, 1]$) representing the importance we give to each one of the components of the formula. Concretely, ϕ_{ct} affects the weight of the controller types whereas ϕ_{sl} affects the sensitivity level of the item. Such factors may be used in a fine grained manner to prioritize one component over the other, in the same way as for the first two equations.

As mentioned earlier, the set of viewers always includes the associated controllers C_i. So, they are considered by the Sharing algorithm. As opposed to the Viewing algorithm, where the associated controllers are always part of the permitted actors to view the item, the Sharing algorithm treats the associated controllers as any other viewer. Therefore, it is not guaranteed that an associated controller can share an item unless the sharing policies specified by the rest of the associated controllers permit her.

Example 2. *The result of running Algorithm 1 in Example 1 was that David is in the* `viewers` *set. We run the Sharing algorithm in order to determine whether David can be a disseminator or not. The sharing policies of the associated controllers are,* $SP_p = \{\langle p, Alice, 1\rangle, \langle p, Bob, 0.50\rangle, \langle p, Carol, 0.25\rangle\}$*. Remember that in Example 1, Carol defined a trust value of 0.5 for David, and Alice defined a trust value of 0.75 for David. We assume 0.25 is the returned value of the infer function due to the indirect connection between Bob and David. In this scenario, David fulfills Carol's sharing policy but he does not satisfy the minimum value of trust set by Alice and Bob generating then a conflict and executing the Sharing algorithm. According to the associated controllers' privacy policies, the value of* `decision_permit` *is equal to 1.25 based on Equation* (3)*, where* $wct(acp.stakeholder)$*=1 and* $wsl(acp.sl)$*=0.25. On the other hand, the value of* `decision_deny` *of Alice is equal to 1.25 based on Equation* (4)*, where* $wct(acp.owner)$*=1 and* $wsl(acp.sl)$*=0.25. The value of* `decision_deny` *of Bob is equal to 1.50 based on Equation* (4)*, where* $wct(acp.stakeholder)$*=1 and* $wsl(acp.sl)$*=0.50. So the final result of denied decisions is 2.75, which means the final decision is to deny David to share the post.*

2.4. Computational Complexity

The algorithms presented here have linear time complexity. The Viewing algorithm with respect to the size of the set of accessors (potential viewers), and the Sharing algorithm with respect to the

```
input  : $ACP_i$, $TG$, viewers and $SP_i$
output: disseminators
permit_sharing, deny_sharing $\leftarrow$ $\mathit{filterActors}(TG, SP_i, \texttt{viewers})$
foreach $a \in$ viewers do
    decision $\leftarrow 0$
    foreach $acp \in ACP_i$ do
        if $a \in$ permit_sharing then
            decision $\leftarrow$ decision $+ \phi_{ct} \cdot wct(acp.ct) + \phi_{sl} \cdot wsl(acp.sl)$
        end
        else if $a \in$ deny_sharing then
            decision $\leftarrow$ decision $- \phi_{ct} \cdot wct(acp.ct) + \phi_{sl} \cdot wsl(acp.sl)$
        end
    end
    if decision $> 0$ then
        disseminators $\leftarrow$ disseminators $\cup \{a\}$
    end
end
```

Algorithm 2: Sharing

size of the input set of viewers. In practice, these boundaries are never large enough to noticeably affect the performance. We explain the complexity of both algorithms in more detail below.

Algorithm 1 Let n be the size of set of input access control policies $|ACP_i|$ and m an upper bound in the size of the set of accesors—i.e., the sum of the sizes of the sets *PER* and *DEN*. The complexity of Algorithm 1 is $O(n \times m)$. This result trivially follows from the fact that, for each policy, it is necessary to go through all the actors included in *PER* and *DEN*. Very often the set of policies n is not very large [6] (it can thus be regarded as constant). Therefore, the time complexity of the algorithm is linear with respect to m.

For a practical implementation, the main drawback of this result is that the upper bound on the size of the sets *PER* and *DEN* might be large. In particular, actors with many friends, or who belong to vastly populated groups, may include in their policy sets *PER* and *DEN* many actors. For instance, some studies show that, on average, Facebook users have around 300 friends [21]. Therefore, optimizations that avoid checking all the actors in *PER* and *DEN* can have a great impact in the performance of the algorithm.

Algorithm 2 Let be j the size of the viewers set |`viewers`|, k the size of the set of input access control policies $|ACP_i|$, s the size of the set of input sharing policies $|SP_i|$, v the number of vertices in the social graph $|V|$, and e the number of edges in the social graph $|E|$, the time complexity of Algorithm 2 is $O((j \times s) + (j \times (k \times (k' \times v + e))))$. The term $(k' \times n + e)$ is worst-case complexity of *FuzzyTrust* which is executed for each policy in ACP_i. As before, the time complexity is linear with respect to j because the number of access control policies k, sharing policies s, and complexity of *FuzzyTrust* are regarded as constant.

Regarding the complexity of *FuzzyTrust*, it is important to note that it is bounded to the degree of separation between the origin and target vertices—i.e., how many connections (friends, friends of friends, friends of friends of friends, etc.) separate two users in an OSN. The term $O(v+e)$ amounts for a breath-first search from the origin vertex to the target and k' for the inferring process once the path has been found (see [10] for details). In the worst case it requires traversing all users, but in practice some studies show that it is rare to have a large degree of separation—for instance, in [22] the authors show that in Twitter the degree of separation between any two random users is ≈ 3.43. Consequently, the algorithm will very often only process 4 vertices and 3 edges. Thus, the algorithm can run efficiently even in OSNs with a large user base.

Though the set of viewers may be large, this algorithm allows for a simple optimization. In many OSNs, it is unnecessary to compute *a priori* the set of users that can share a post. It is possible to execute the algorithm for a single actor on demand. For instance, when the actor is about to view the item, the algorithm may be run for this particular actor. As a consequence the factor j in the previous complexity is reduced to a constant size of 1.

2.5. Implementation and Analysis of Viewing and Sharing Algorithms

We implemented our algorithms in Python and executed it on a MacBook Air with 2.2 GHz Intel Core i7 CPU and 8 Gb of RAM. Concretely, for the Viewing algorithm we computed all possible combinations of trust, sensitivity level, accessor types and the associated controllers types, whereas for the Sharing one, we used the sensitivity level and the associated controllers.

In the Viewing algorithm, each one of these associated controllers might have 5 possible trust values, 4 sensitivity levels and 3 different accessors types for the accessors. For the Sharing algorithm, the associated controllers might only have 4 sensitivity levels defined. Finally, each one of the associated controllers may permit or deny the access or the sharing action for an item.

We split the analysis into two main parts corresponding to Viewing and Sharing. We omit the factors as the objective is to see the interplay of our components and not how the factors affects the outcome.[6] Additionally, and for the sake of simplicity, in what follows we will focus on the cases where the distance between associated controllers is 1 (see Table 2). Though the remaining cases are not explicitly presented, they can be found in our implementation. All the source code of our implementation are publicly accessible online [23].

Finally, as a consequence of the collaborative nature of our proposal, both Viewing and Sharing algorithms generate a so-called *flipping point*. This point represents the number of associated controllers who are needed in order to revoke an action. For example, let assume that Alice (the owner of an item) does not want Bob to access to the item, but other associated controllers want to let him access to it. The question is then how many associated controllers are needed in order to revoke Alice's policy and let Bob access to the item. We calculated that point for both algorithms as explained later in this section assuming all involved factors to be 1.

Viewing. In all the figures related to Viewing, i.e., Figures 2 to 5, the root node denotes the associated controllers for whom the experiment is running. The first level means the possible trust values. The second level symbolizes the possible values for the sensitivity levels. The third level stands for the accessor types possibilities whereas finally, the leaves represent the output of the equation being executed according to the experiment.

Figure 2 shows all the outputs when there is only one associated controller, either an owner or a stakeholder—both associated controllers generate the same values. For this experiment we created

[6]The factors may be considered as parameters that fine tune our decision algorithms, providing a range of decisions from more conservative to more liberal.

an access control policy containing an arbitrary accessor when she is in the permitted set (PER) of the owner/stakeholder.

Similarly, we run the same experiment for the contributor/originator as both generate the same values when the distance is 1, and the outputs can be seen in Figure 3. As expected, the values are the same as in Figure 2 with a difference of -0.5—which is the only difference between owners/stakeholder and contributor/originator (see Table 2).

We generated all possible values for Equation (2), i.e., when the accessor is in the denied set, and the results are shown in Figures 4 and 5 for the owner/stakeholder and contributor/originator, respectively. We can see that when the owner/stakeholder or the contributor/originator define the accessor in the DEN set and the values of the trust, sensitivity level and the access type are none, high and AN respectively, then we apply a veto right so the accessors will be automatically denied. Finally, we included in Figures A.7 and A.8 corresponding to the combinations where the contributor distance $\geq$2 and the viewer is in the PER and DEN respectively.

As an example, let us assume a scenario where there are two associated controllers: an owner and an originator. The owner has the accessor in the permitted set whereas the originator has she in the denied one. On the one hand, the owner defines her privacy policy as follows: trust=*highest*; sensitivity_level=*low*; accessor_type=*an*. On the other hand, the originator defines it as: trust=*none*; sensitivity_level=*medium*; access_type=*gn*. Then, the output of Equation (1) is 2.75 whereas the output of Equation (2) is 3.25. The final decision is that the accessor can access to the item (since $3.25 - 2.75 > 0$).

Sharing. In Figures 6a and 6b, the root node denotes the associated controllers for whom the experiment is running. The first level denotes the possible values for the sensitivity levels and the leaves represent the output of Equations (3) and (4) respectively.

To evaluate the Equations (3) and (4), we calculated all the possible values that these equations can generate when the viewer is in either `permit_sharing` or `deny_sharing` sets generated by the external procedure *filterActors*.

In particular, Figures 6a and 6b depict the decision values when the viewer is in the `permit_sharing` for the owner/stakeholder (Figure 6a) and for the contributor when the distance is equal to 1 (see Figure 6b). As expected, the value of the leaves only differ by 0.5 (see Table 2). It is worth mentioning that we have not included Figures when the viewer is in the `deny_sharing` because in the Sharing algorithm they produce the same ones. In Appendix A, we run the same experiments and generated the same figures for the rest of the cases, i.e., when the distance of the originator is greater than 2, and all the remaining cases for the originator, i.e., when the *infer* function returns either 0.25 or 0.5.

As an example, let us assume a scenario where there are two associated controllers: an owner and a contributor. The owner has the accessor in the permitted set whereas the contributor has she in the denied one. The owner defines her sensitivity level as low whereas the contributor defines the sensitivity level as medium. In both cases, the viewer fulfills the sharing threshold defined by the associated controllers. With these values, the output of Equation (3) is 1.25 whereas the output of Equation (4) is 1. The decision is that the viewer can finally share the item, since $1.25 - 1 > 0$.

Flipping Point. We computed the *flipping point* to determine how many associated controllers are needed in order to revoke the decision taken by another (set of) associated controller(s). Note that this is a combinatorics problem since there can exist a large number of stakeholders—the upper bound is given by the number of actors in the OSN.

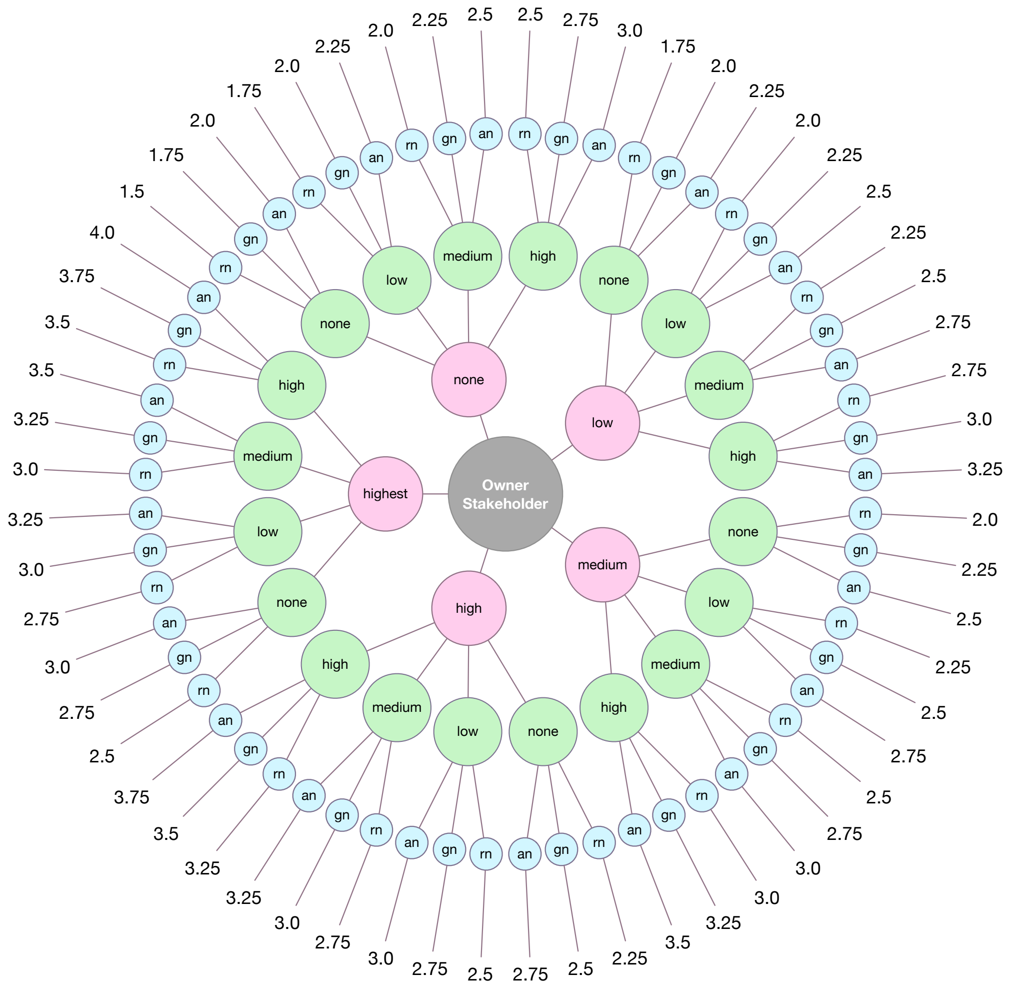


Figure 2: Viewing: Decision values for the owner/stakeholder with $(\phi_{ct}, \phi_{at}, \phi_{tr}, \phi_{sl}) = 1$ and the accessor $\in PER$

We first focus on the Viewing algorithm with one of the simplest scenarios, i.e., the owner and a stakeholder, who wants to revoke the owner's decision. In this case, the accessor is in the permitted set of the owner. On the contrary, the accessor is the denied set of the stakeholder. We
550 computed the frequency of all the possible outputs (second row of Table 6). The third row shows how many different ways a stakeholder can revoke the initial decision, i.e., how many possible ways the stakeholder can get a number greater than the output given in the second row. Note that, the stakeholder can use the veto right and revoke the owner's decision when the components are: trust=*none*; sensitivity_level=*high*; access_type=*an*. In the last row, we computed the probability
555 that an owner's decision might be revoked by one stakeholder. For example, when the owner sets the trust=*highest*; sensitivity_level=*high* and access_type=*rn*, there is a 11.6% of probability that the stakeholder revokes her decision.

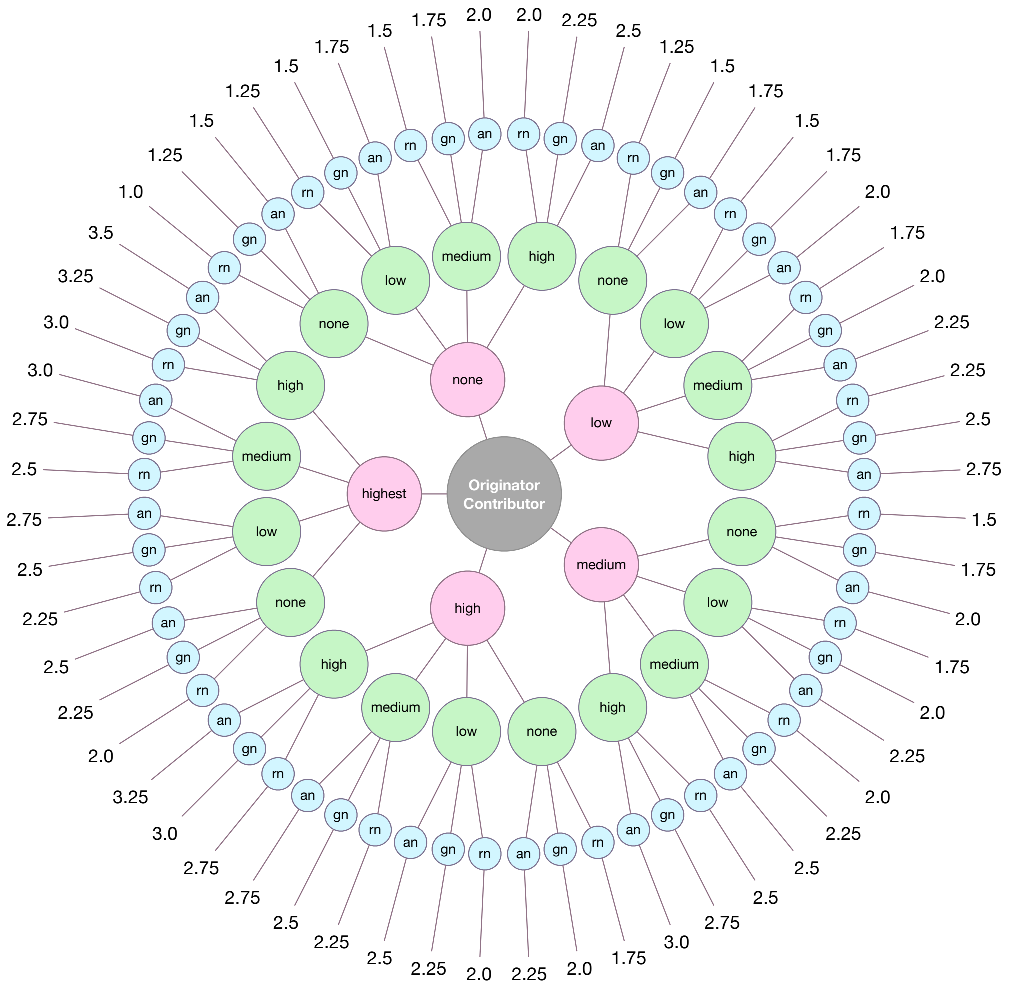


Figure 3: Viewing: Decision values for the contributor/originator with $(\phi_{ct}, \phi_{at}, \phi_{tr}, \phi_{sl}) = 1$; accessor $\in PER$, and; distance=1

In order to compute the baseline probability for any number of associated controllers, we have two cases depending whether the accessor is: 1) in the PER set, or; 2) in the DEN set of the associated controllers. The PER set directly affects the output (row 1) and the frequency (row 2), whereas the DEN set only affects the revocation number (row 3). Therefore, we should first generate the associated controllers who have the intended accessor in their PER set and then in their DEN set.

On the one hand, if the accessor is in the PER set, the range of the outputs is computed by multiplying all the values of the first row by the max (4.0) and min (1.5) values by the number of stakeholders plus the owner. Finally, both the frequency and the revocation number should be recalculated to obtain the probability. On the other hand, if the accessor is in the DEN set, both

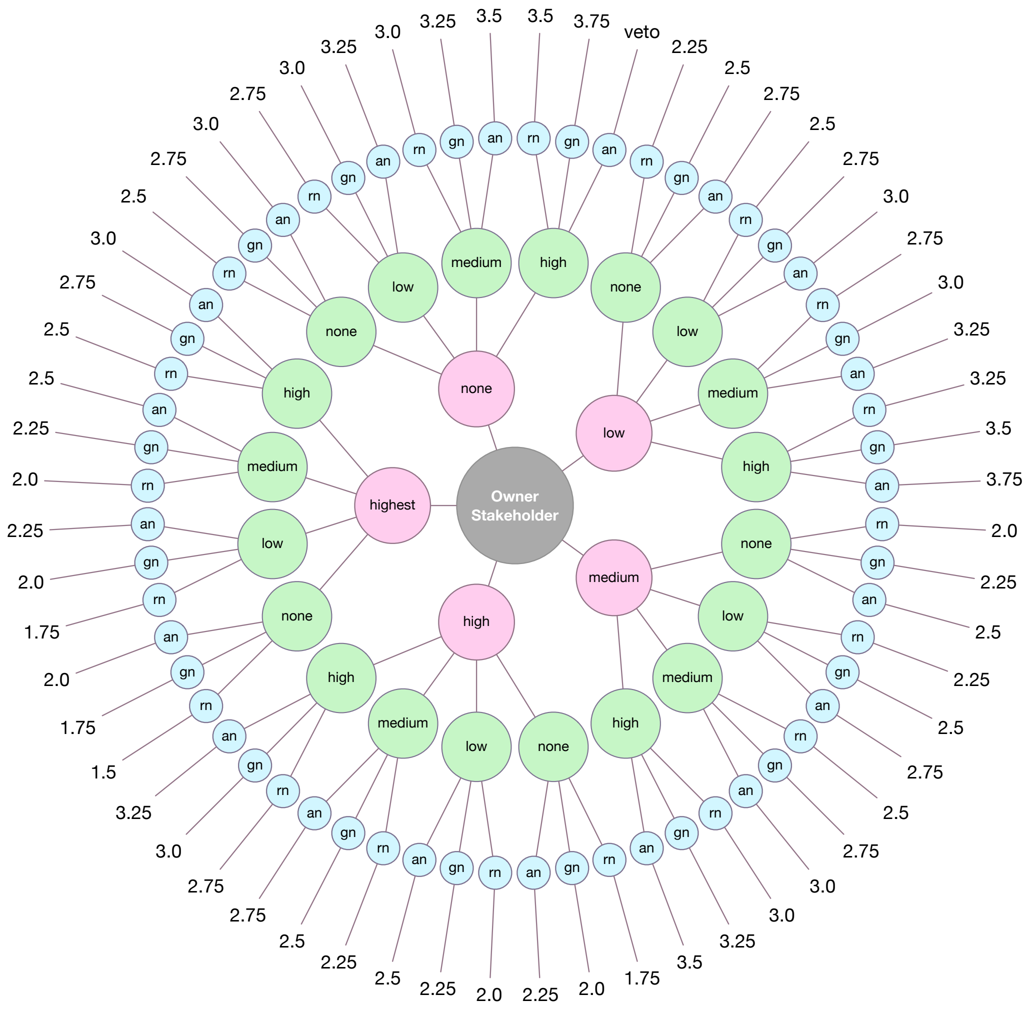


Figure 4: Viewing: Decision values for the owner/stakeholder with $(\phi_{ct}, \phi_{at}, \phi_{tr}, \phi_{sl}) = 1$ and the accessor $\in DEN$

the range of the outputs and the frequency remain the same. However, the revocation number must be recalculated according to the frequency row, i.e., computing the number possibilities for the stakeholder to get a number greater than the output (row 1). 570

As an example, let us suppose that there are 3 stakeholders plus the owner having all of them the accessor in the *PER* set. Then the range of the output will go from 6 to 16 by steps of 0.25. Under these circumstances, and without using the veto right, it is impossible for only one stakeholder to revoke the decision (note that the maximum value that a stakeholder can achieve is 3.75).

575 We also computed the same baseline probability when there is an owner and one contributor/originator and the distance is 1. The results can be seen in Table 7. The same strategy regarding the *PER* and *DEN* sets explained above is also applied in this table. However, since our framework can only have one contributor and one originator, the possibilities are simplified to

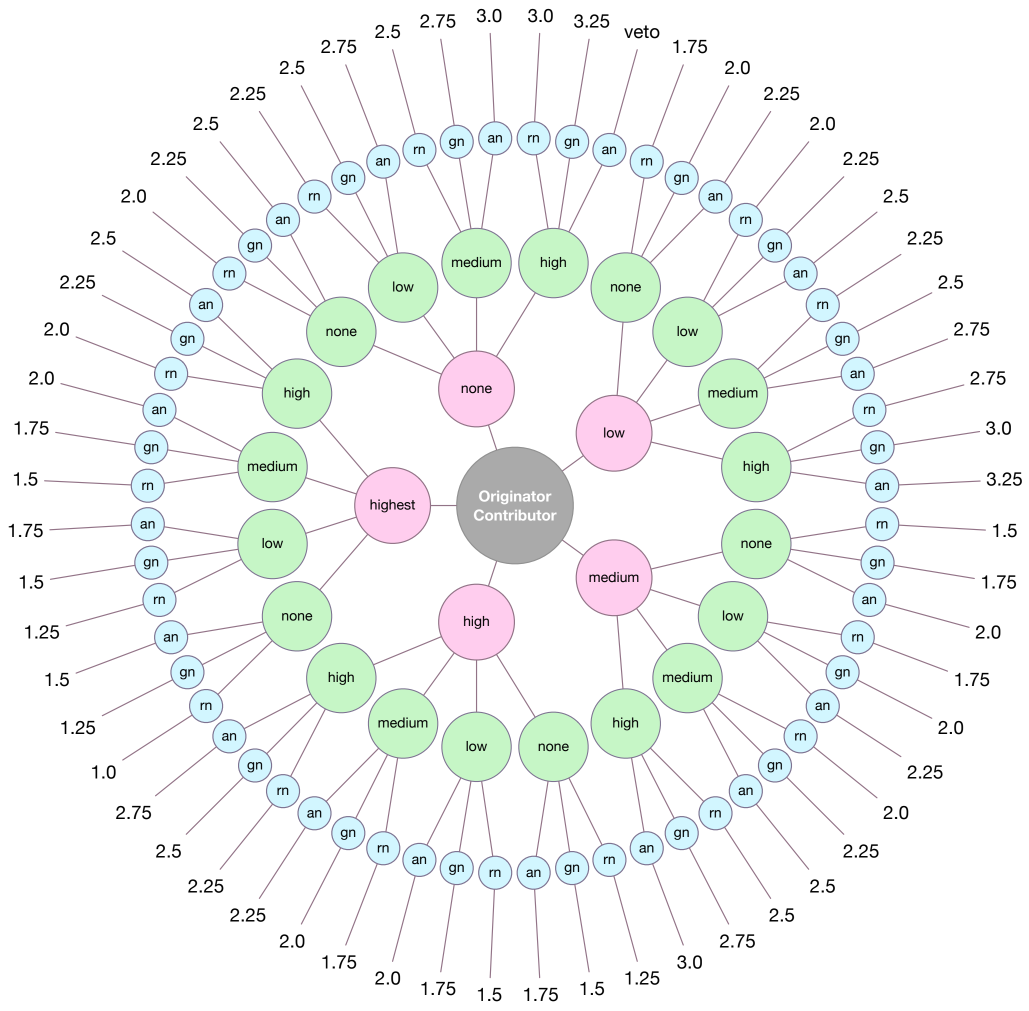
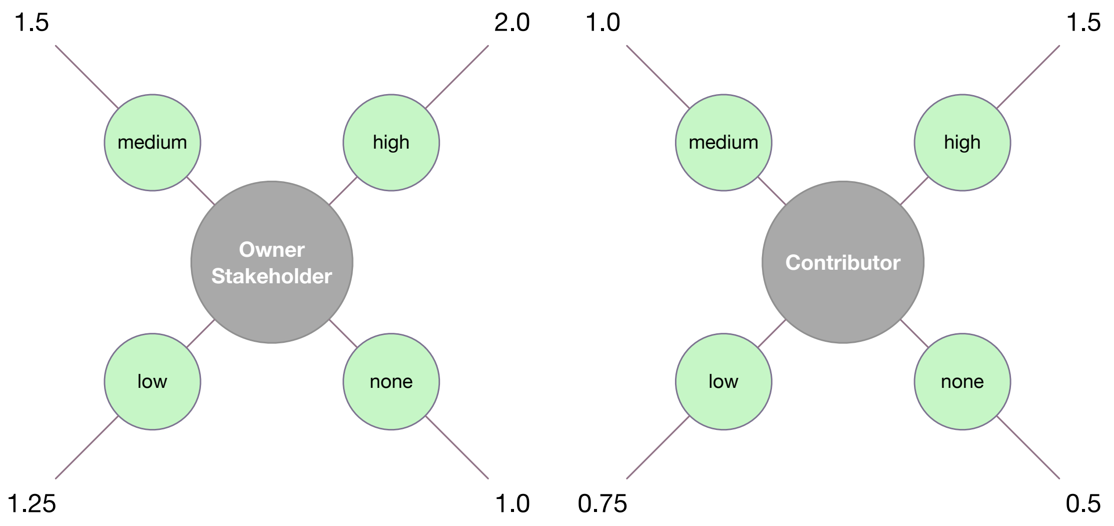


Figure 5: Viewing: Decision values for the contributor/originator with $(\phi_{ct}, \phi_{at}, \phi_{tr}, \phi_{sl}) = 1$; accessor $\in DEN$, and; distance=1

the combinations of 3 elements, i.e., if the accessor is in the PER or DEN sets of the owner, the contributor or the originator. In Appendix B we carried the same experiment fixing the distance ≥ 2 (see Table B.12).

Without having the veto right into consideration, we conclude that whenever there are 5 associated controllers (remember that the owner is mandatory), if 4 of them have a different opinion than the other one, the decision will be definitely revoked in the Viewing algorithm. Note that the veto right will automatically deny an accessor to access to the item.

We carried out the same analysis for the Sharing algorithm. The results can be seen in Table 8 for the stakeholder/owner, Table 9 for the contributor when the distance is 1, and Table 10 for an originator when her weight is 0.75. For completeness, we included in Appendix B the base-



(a) owner/stakeholder (b) contributor with distance=1

Figure 6: Sharing: Decision values with $(\phi_{ct}, \phi_{sl}) = 1$ and viewer $\in$ `permit_sharing`

Table 6: Viewing baseline probability of revoking the owner or a stakeholder's initial decision for only one stakeholder or the owner respectively

Viewing — Stakeholder											
Output	4	3.75	3.5	3.25	3.0	2.75	2.5	2.25	2.0	1.75	1.5
Frequency	1	2	4	6	9	10	10	8	6	3	1
Revocation number	1	3	7	13	22	32	42	50	56	59	60
Baseline Probability	1.6%	5%	11.6%	21.6%	36.6%	53.3%	70%	83.3%	93.3%	98.3%	100%

line probability computation when the contributor's distance is ≥ 2 (Table B.13) and when the originator's weight is 0.25 (see Table B.14).

It is interesting to see that if an owner achieves the maximum value (2), then only a stakeholder plus one more associated controllers are needed to revoke the owner's decision. In addition to that, if there are 3 associated controllers, then the owner's decision will always be revoked. Finally, while contributors and originators are the less powerful associated controllers—they can only achieve a 1.5 and 1.75 respectively, they can be crucial when there only are a few stakeholders involved in the sharing decision.

3. Proof-Of-Concept: Diaspora

Diaspora belongs to the family of decentralized OSNs. In such OSNs there is no single entity where all information is stored. Instead, they consist of independent nodes which host all the

Table 7: Viewing baseline probability of revoking the owner's initial decision for one contributor/originator when distance is 1

Viewing — Contributor/Originator													
Output	4	3.75	3.5	3.25	3.0	2.75	2.5	2.25	2.0	1.75	1.5	1.25	1.0
Frequency	1	2	4	6	9	10	10	8	6	3	1	0	0
Revocation number	1	1	1	3	7	13	22	32	42	50	56	59	60
Baseline Probability	1.6%	1.6%	1.6%	5%	11.6%	21.6%	36.6%	53.3%	70%	83.3%	93.3%	0%	0%

Table 8: Sharing baseline probability of revoking the owner or a stakeholder's initial decision for only one stakeholder or the owner respectively

Sharing — Stakeholder				
Output	2.0	1.5	1.25	1.0
Frequency	1	1	1	1
Revocation number	0	1	2	3
Baseline Probability	0%	25%	50%	75%

Table 9: Sharing baseline probability of revoking the owner's initial decision for one contributor when distance is 1

Sharing — Contributor						
Output	2	1.5	1.25	1.0	0.75	0.5
Frequency	1	1	1	1	0	0
Revocation number	0	0	1	1	2	3
Baseline Probability	0%	0%	25%	25%	0%	0%

Table 10: Sharing baseline probability of revoking the owner's initial decision for one originator when her weight is 0.75

Sharing — Originator					
Output	2	1.5	1.25	1.0	0.75
Frequency	1	1	1	1	0
Revocation number	0	1	1	2	3
Baseline Probability	0%	25%	25%	50%	0%

information of the social network. Diaspora nodes are called *pods*. A pod is basically a server which host an instance of Diaspora's source code and its own database.

We deployed a particular instance of our approach in Diaspora [23]. Since our main goal is to demonstrate that our approach can be deployed in a real world application, and for the sake of simplicity, we did not use the decentralized architecture that Diaspora provides. Instead, we deployed our own pod on a MacBook Pro with 2.9 GHz Intel Core i5 CPU and 8 Gb of RAM.

Table 11: Comparison between Diaspora and our Diaspora implementation.

Components	Diaspora	Proof-of-Concept
Controllers Types	*owner*	*owner*, *stakeholder*
Acessors Types	*rn*	*rn*
Sensitivity Levels	$\emptyset$	*none*, *low*, *medium*, *high*
Trust	$\emptyset$	*none*, *low*, *medium*, *high*, *highest*

Diaspora does not allow to define sensitivity levels with respect to an item, nor trust between users, we extended Diaspora in order to include such notions (see Table 11). Due to the way Diaspora is implemented, our proof-of-concept implementation only includes the *owner* and *stakeholder* controllers types and accessor type *rn*. Implementing *contributor* and *originator* controller types is not possible because Diaspora does not provide the feature of posting in someone else's profile. Moreover, the accessor types *gn* and *an* are not performed due to other Diaspora constraints: 1) no distinction between relationships and groups, and; 2) social activities such as posting, sharing, etc., are defined over relationships.

Usability. In our implementation we provide a user-friendly system to define privacy settings, having default values that favour the privacy of the users. In Appendix C we included some screenshots corresponding to the different interfaces of our proof-of-concept implementation. We allow users to define both the permitted and denied set of accessors either by using either the defined relationships, e.g., "family" or "friends", or more general relationships like "Everyone" and "Nobody" (see Figure C.11). The associated controllers can also determine: i) the sensitivity level of shared items based on their contents; ii) the trust level of each relationship, and; iii) the trust level for users who do not belong to any relationship. We also implemented such options as can be seen in Figure C.12.

Regarding how users interact with the sharing action, we implemented in our Diaspora instance a system that allows the associated controllers to specify their privacy settings (see Figure C.12). Having in mind that not all the associated controllers and accessors who have privileges to view an item have the rights to share it, the share button only appears to those associated controllers and viewers who can do it.

A limitation of our algorithms is that the resulting audience of an item is not known to stakeholders until the item has been posted. Unfortunately, this is an inherent limitation of all aggregation based collaborative access control models [3, 24, 6]. Typically, this problem is mitigated by giving one of the stakeholders the authority of deciding whether the result of the algorithm is satisfactory—e.g., in [6] this authority is given to the owner. Instead, we opt for keeping this choice open for future research. For example, we think that a possible solution would be to introduce a voting phase after the algorithm computes the audience of the item. In this voting phase, all stakeholder must confirm that audience computed by our algorithm satisfies the preferences. As of today, we believe that there is no silver bullet and different options must be investigated.

Validation. To validate our proof-of-concept, we conducted a battery of test cases that contained fourteen users and forty-eight relationships between them. We considered a few scenarios where users were in permitted and denied sets at the same time, as well as the typical operations of adding and removing users with different privacy settings and posting items. Also, we tested cases where some of the associated controllers do not have any privacy preferences regarding the given item. We ran tests regarding the sharing action for the viewers and the associated controllers. Privacy policy elements such as sensitivity level, permitted accessors, denied accessors and trust levels were changed in each action in order to check whether we got the expected outcome under different settings. Table D.15 in the appendix shows the fourteen tests we executed.

4. Related Work

In access control models for OSNs, we can distinguish between two main approaches: 1) mechanisms which assume that the information is governed by a single user, e.g., [25, 26, 27], and; 2) mechanisms where a collaborative decision regarding the information is made, e.g., [18, 16, 28, 29, 17, 6, 7, 30, 31, 32]. In the following we only focus on the second approach and we analyze the most relevant proposals published on this topic.

Squicciarini et al. [29, 17] provide a novel collective privacy mechanism for content sharing among users in OSNs. Authors consider that the privacy control of the shared content is co-owned by multiple users. Each user may separately specify her own privacy settings for the shared data and thus, a voting algorithm to enable collective enforcement for shared data is used. However, in their algorithm only the winners of the voting algorithm control who can access the data, instead of harmonizing all users' privacy preferences.

Carminati and Ferrari [24] introduce a collaborative access control mechanism in OSNs that integrates the topology of social networks in policy-making. They improve topology-based access control taking into account a set of collaborative users by giving a new class of security policies, called collaborative security policies, which indicate the set of users who should contribute to the collaboration. In contrast, our work proposes a formal collaborative model to manage viewing and sharing of shared items in OSNs as well as a fine-grained policy specification scheme.

Similarly, Such and Criado [18] propose a mechanism to resolve multiparty conflicts based on the willingness of each associated controller to give access. However, if one user has high willingness and another one low willingness, only the former will determine the final access.

A policy-based approach to control access to shared data in OSNs is proposed by Wishart et al. [33]. In this case, the owner of the content is allowed to specify policies for the content she uploads and other users (called trusted co-owners, who previously must be invited by the owner) can edit such a policy. In our proposal we use the same concept in the sense that the owner has to mention users (stakeholders) to be part of the collaborative decision. However, in our work stakeholders do not directly edit the owners policy, and instead we consider their access policies to calculate the decision.

Xu et al. [7] propose a collaborative privacy management mechanism where the collective decision is made by the user who wants to post data (the owner, who is responsible for gathering feedback from other involved users). Though trust values are used to indicate how much influence a user's opinion will have on the aggregated decision, the owner has full decision power on who should access the item.

The approach proposed by Xu et al. [34] is similar to our work in the sense that it offers a systematic solution for detecting and resolving privacy conflicts for collaborative data sharing in OSNs. However, their approach needs a negotiation mechanism to solve the privacy policies conflicts before the access privileges are computed. In order to fix that issue, they improved their work by enhancing a policy specification scheme and a voting-based conflict resolution mechanism [6]. Nevertheless, the conflict resolution strategy presented in such work is selected by the data owner which leads to a unilateral decision, i.e., without considering the privacy preferences of other associated controllers involved. Our work could be seen as an extension as the one presented in [6]: all the associated controllers are taken into consideration for the collaborative decision, so we are indeed giving a truly collaborative access control framework.

Based on the multiparty access control model presented in [6], Vishwamitra et al. [35] introduced a model that allows each involved user in a photo to independently decide whether some personally identifiable information in the photo is shown or blurred. In our scenario, the collaborative decision protects all the items, including photos, from being viewed or shared. Controlling the content of the item is outside the scope of this article.

Ilia et al. [16] proposed a multi-party access control for OSNs based on a cryptographic scheme called secret sharing [36]. Essentially, data owners and stakeholders encrypt the items and the shares needed to reconstruct the encryption key are shared with their trusted friends. A user decrypts an item if she collects enough shares of the key. The penalty of using secret sharing is the need of a trusted party—a key management service—to generate both the keys and the secret shares. In our model, we assume that the OSN provider a is trusted party, i.e., it runs our algorithms correctly and there is a access control system in place that correctly enforces each policy. Consequently, the use of encryption is not required.

Gay et al. [37] provide a fine-grained privacy mechanism in decentralized OSNs. Similar to our work, they also base their enforcement of privacy policies on ReBAC. Despite this apparent similarity, our access control policy has more fine-grained features such as the possibility to define an explicit denied set and accessor specification policies. In their work, only trust is used to determine which usersare allowed to propagate the item. In contrast, we use trust in the process of computing the collaborative decision regarding who can view or share an item.

5. Discussion and Conclusions

We presented a collaborative access control framework for OSNs that collectively achieves a decision about who should (not) access, or (not) share, an item. The decision is based on the privacy settings of all concerned associated controllers, i.e., owner, originator, contributor and stakeholder(s). This is done by taking into account the following four aspects: trust relationship between users, sensitivity level of the users with respect to the concerned item as well as different weights for both the controller types and accessor types. We proposed a Viewing and a Sharing algorithms for taking such a decision about the items. We evaluated the different outputs of our algorithms based on all combinations of inputs to better understand what the consequences of choosing a policy are (e.g., how likely it is that a policy will be revoked). Finally, we developed a proof-of-concept on Diaspora, and tested it against some predefined scenarios.

The theoretical complexity analysis of our algorithms, together with the tests of our proof-of-concept, constitute a preliminary evaluation on the performance of our access control model. However, in order to determine the feasibility of this approach in a real system, a thorough evaluation must be carried out, i.e., an evaluation in a system including real data coming from real users. High workload combined with large amounts of data may reveal weaknesses that need to be addressed in order to use our access control mechanism in production. We leave this evaluation as future work.

Concerning the correctness of our solution, different decisions could have been taken depending on whether one might want to privilege privacy over utility or vice-versa. This trade-off between privacy and utility is stretched or relaxed depending on the policies and their expressiveness(delimited by the factors). We plan to study the impact of different values for the factors on the privacy/utility trade-off in future work.

Acknowledgements

This research has been partially supported by the Swedish Research Council (*Vetenskapsrådet*) under grant Nr. 2015-04154 (*PolUser: Rich User-Controlled Privacy Policies*) and the Cultural Office of the Saudi Embassy in Berlin, Germany.

References

References

[1] C. Gates, Access control requirements for web 2.0 security and privacy, IEEE Web 2.0 privacy and security workshop (W2SP'07) 2 (0).

[2] P. W. Fong, M. Anwar, Z. Zhao, A privacy preservation model for facebook-style social network systems, in: European Symposium on Research in Computer SecurityESORICS, Springer, 2009, pp. 303–320.

[3] K. Thomas, C. Grier, D. M. Nicol, unfriendly: Multi-party privacy risks in social networks, in: International Symposium on Privacy Enhancing Technologies Symposium, Springer, 2010, pp. 236–252.

[4] J. M. Such, J. Porter, S. Preibusch, A. Joinson, Photo privacy conflicts in social media: A large-scale empirical study, in: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, ACM, 2017, pp. 3821–3832.

[5] J. M. Such, N. Criado, Multiparty privacy in social media, Communications of the ACM 61 (8) (2018) 74–81.

[6] H. Hu, G.-J. Ahn, J. Jorgensen, Multiparty access control for online social networks: model and mechanisms, IEEE Transactions on Knowledge and Data Engineering 25 (7) (2013) 1614–1627.

[7] L. Xu, C. Jiang, N. He, Z. Han, A. Benslimane, Trust-based collaborative privacy management in online social networks, Forensics and Security.

[8] Diaspora, Diaspora, `https://joindiaspora.com`, [Available Online] (2016).

[9] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decision support systems 43 (2) (2007) 618–644.

[10] M. Lesani, S. Bagheri, Applying and inferring fuzzy trust in semantic web social networks, in: Canadian Semantic Web, 2006, pp. 23–43.

[11] J. Caverlee, L. Liu, S. Webb, Socialtrust: tamper-resilient trust establishment in online communities, in: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital librariesJCDL, 2008, pp. 104–114.

[12] J. A. Golbeck, Computing and applying trust in web-based social networks, Ph.D. thesis (2005).

[13] U. Kuter, J. Golbeck, Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models, in: AAAI, Vol. 7, 2007, pp. 1377–1382.

[14] K. Carley, A theory of group stability, American sociological review (1991) 331–354.

[15] S. L. Feld, The focused organization of social ties, American journal of sociology 86 (5) (1981) 1015–1035.

[16] P. Ilia, B. Carminati, E. Ferrari, P. Fragopoulou, S. Ioannidis, Sampac: socially-aware collaborative multi-party access control, in: CODASPY, 2017, pp. 71–82.

[17] A. C. Squicciarini, M. Shehab, J. Wede, Privacy policies for shared content in social network sites, The VLDB Journal 19 (6) (2010) 777–796.

[18] J. M. Such, N. Criado, Resolving multi-party privacy conflicts in social media, IEEE Transactions on Knowledge and Data Engineering 28 (7) (2016) 1851–1863.

[19] S. De Capitani Di Vimercati, S. Foresti, P. Samarati, S. Jajodia, Access control policies and languages, International Journal of Computational Science and Engineering 3 (2) (2007) 94–102.

[20] S. D. C. di Vimercati, P. Samarati, S. Jajodia, Policies, models, and languages for access control, in: International Workshop on Databases in Networked Information Systems, Springer, 2005, pp. 225–237.

[21] J. Ugander, B. Karrer, L. Backstrom, C. Marlow, The anatomy of the facebook social graph, CoRR abs/1111.4503.

[22] R. Bakhshandeh, M. Samadi, Z. Azimifar, J. Schaeffer, Degrees of separation in social networks, in: Proceedings of the Fourth Annual Symposium on Combinatorial Search, SOCS 2011, Castell de Cardona, Barcelona, Spain, July 15.16, 2011, 2011.

[23] H. Alshareef, R. Pardo, G. Schneider, P. Picazo, A Collaborative Access Control Framework for Online Social Networks (Feb. 2019). `doi:10.5281/zenodo.3570319`.
URL `https://doi.org/10.5281/zenodo.3570319`

[24] B. Carminati, E. Ferrari, Collaborative access control in on-line social networks, in: CollaborateCom, 2011, pp. 231–240.

[25] J. Grossklags, N. Christin, J. Chuang, Secure or insure?: a game-theoretic analysis of information security games, in: WWW, 2008, pp. 209–218.

[26] Y. Cheng, J. Park, R. Sandhu, Relationship-based access control for online social networks: Beyond user-to-user relationships, in: 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing, IEEE, 2012, pp. 646–655.

[27] P. W. Fong, I. Siahaan, Relationship-based access control policies and their policy languages, in: Proceedings of the 16th ACM symposium on Access control models and technologiesSACMAT, ACM, 2011, pp. 51–60.

[28] S. Rajtmajer, A. Squicciarini, C. Griffin, S. Karumanchi, A. Tyagi, Constrained social-energy minimization for multi-party sharing in online social networks, in: AAMAS, 2016, pp. 680–688.

[29] A. C. Squicciarini, M. Shehab, F. Paci, Collective privacy management in social networks, in: WWW, 2009, pp. 521–530.

[30] W. Sherchan, S. Nepal, C. Paris, A survey of trust in social networks, ACM Computing Surveys (CSUR) 45 (4) (2013) 47.

[31] F. Paci, A. Squicciarini, N. Zannone, Survey on access control for community-centered collaborative systems, ACM Computing Surveys (CSUR) 51 (1) (2018) 6:1–6:38.

[32] A. C. Squicciarini, S. M. Rajtmajer, N. Zannone, Multi-party access control: Requirements, state of the art and open challenges, in: Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, ACM, 2018, pp. 49–49.

[33] R. Wishart, D. Corapi, S. Marinovic, M. Sloman, Collaborative privacy policy authoring in a social networking context, in: POLICY, 2010, pp. 1–8.

[34] H. Hu, G.-J. Ahn, J. Jorgensen, Detecting and resolving privacy conflicts for collaborative data sharing in online social networks, in: ACSAC, 2011, pp. 103–112.

[35] N. Vishwamitra, Y. Li, K. Wang, H. Hu, K. Caine, G.-J. Ahn, Towards pii-based multiparty access control for photo sharing in online social networks, in: Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies, ACM, 2017, pp. 155–166.

[36] B. Schneier, Applied cryptography: protocols, algorithms, and source code in C, john wiley & sons, 2007.

[37] R. Gay, J. Hu, H. Mantel, S. Mazaheri, Relationship-based access control for resharing in decentralized online social networks, in: International Symposium on Foundations and Practice of Security, Springer, 2017, pp. 18–34.

Appendix A.

For completeness, we included the results of the experiments presented in Section 2.5 for the
825 remaining cases in the Viewing algorithm, that is, when the originator/contributor are not directly connected to the owner, i.e., distance ≥2, and the viewer is in the PER set (see Figure A.7) and when she is not (see Figure A.8).

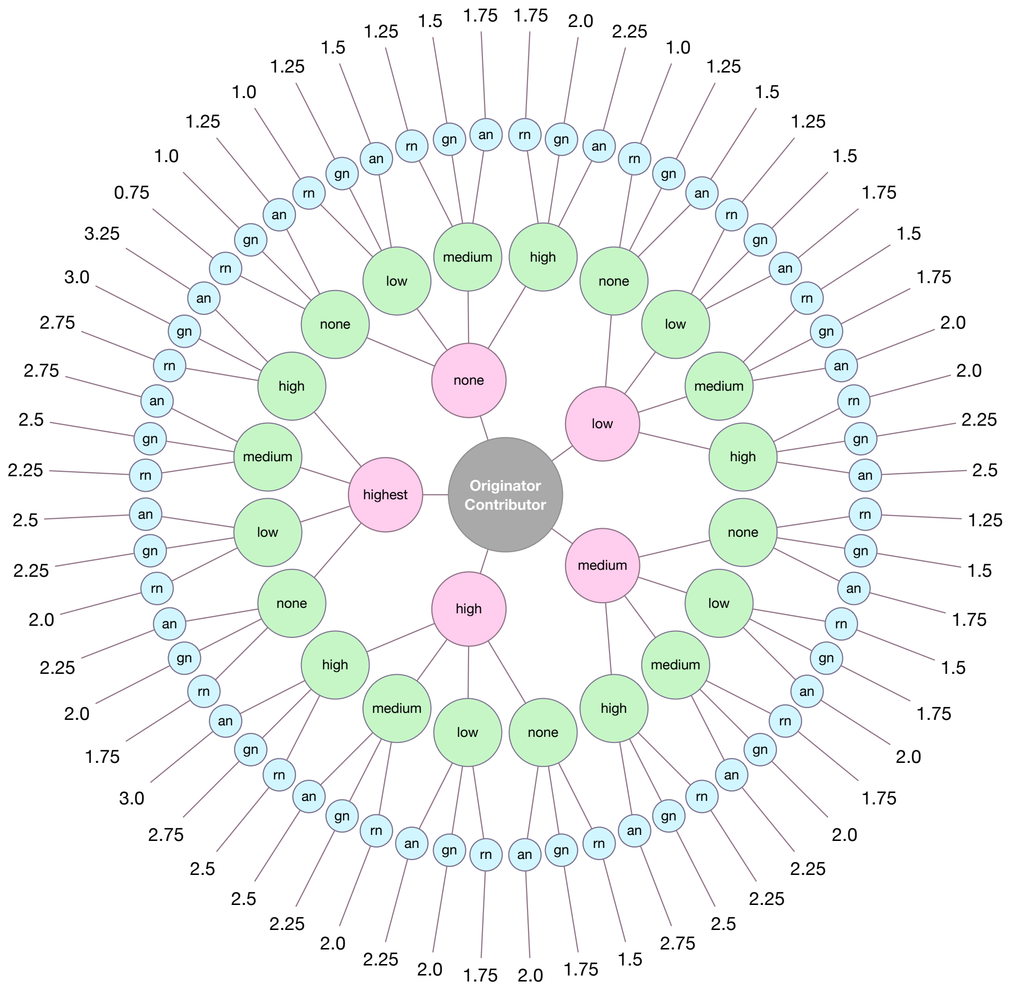


Figure A.7: Viewing: Decision values for the contributor/originator with $(\phi_{ct}, \phi_{at}, \phi_{tr}, \phi_{sl}) = 1$; accessor $\in PER$, and; distance≥2

Regarding the Sharing algorithm, we generated the combinations for a contributor when the distance ≥2 and the viewer is in the `permit_sharing` set (see Figure A.9). Additionally, we run
830 the experiments for the originator when the viewer is in the `permit_sharing` set. We generated two figures according to the trust level, i.e., when the $TG.infer$ $(originator, owner)$ returns 0.25 (see Figure A.10a) and when it returns 0.75 (see Figure A.10b).

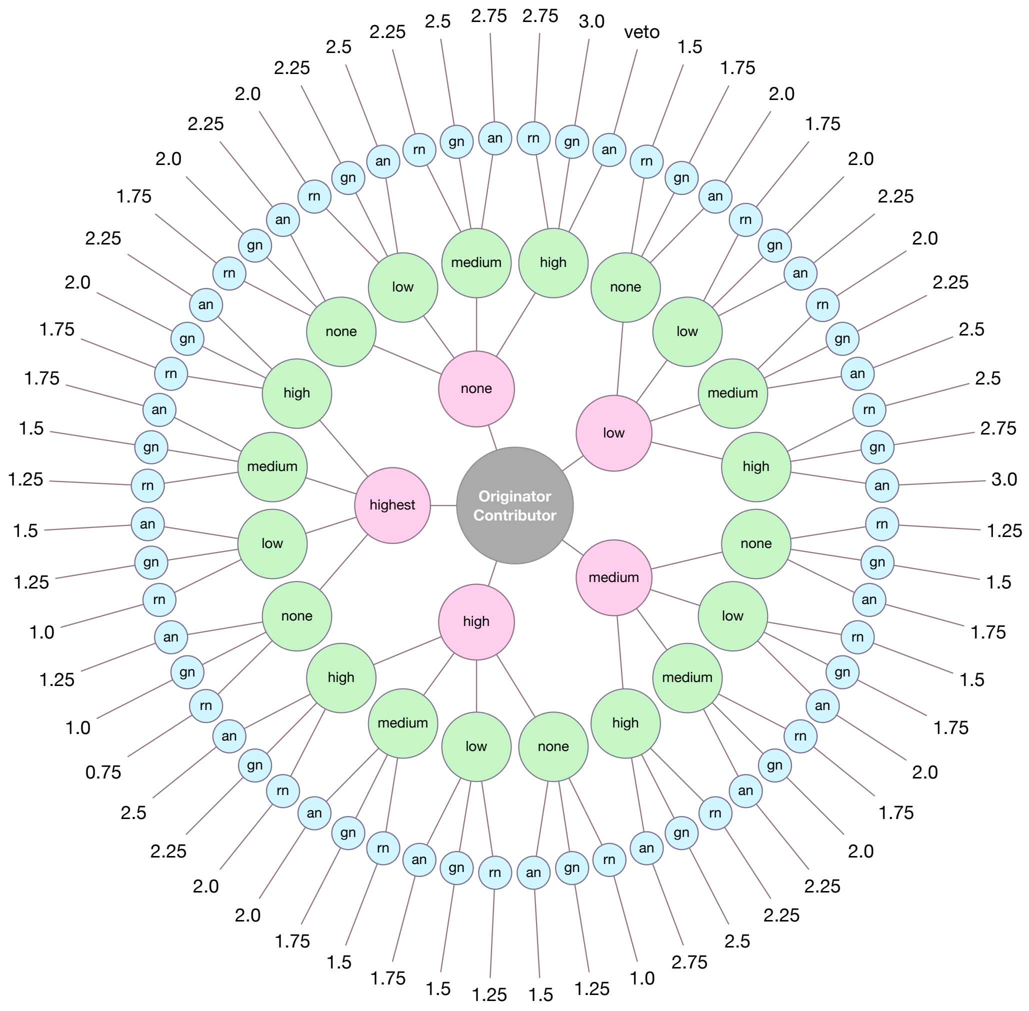


Figure A.8: Viewing: Decision values for the contributor/originator with $(\phi_{ct}, \phi_{at}, \phi_{tr}, \phi_{sl}) = 1$; accessor $\in DEN$, and; distance≥2

Appendix B.

We executed the experiments in order to compute all the baseline probabilities for the Viewing algorithm when the distance of both the contributor and the originator is ≥2 (see Table B.12). 835

For the Sharing algorithm, we included tables corresponding to the contributor when the distance ≥2 (see Table B.13) and to the originator when the weight is 0.25 (see Table B.14).

Appendix C.

Here we show different screenshots of the UI of the collaborative access control prototype that we implemented in Diaspora. In Figure C.11, associated controllers can specify their privacy preferences 840

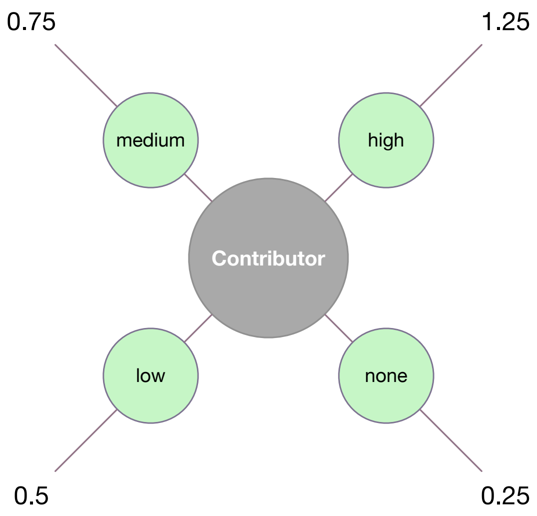
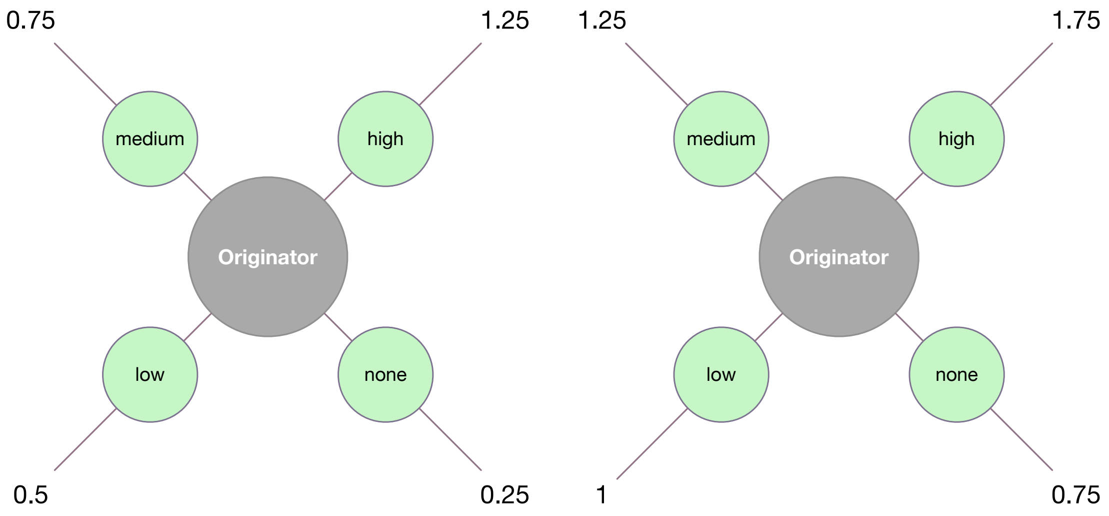


Figure A.9: Sharing: Decision values for the contributor with $(\phi_{ct}, \phi_{sl}) = 1$; viewer $\in$ `permit_sharing`, and; distance ≥ 2



(a) $TG.infer\ (originator, owner)$=0.25 (b) $TG.infer\ (originator, owner)$=0.75

Figure A.10: Sharing: Decision values for the originator with $(\phi_{ct}, \phi_{sl}) = 1$; viewer $\in$ `permit_sharing`

regarding who—accessors—are allowed to access the item and who are not. Figure C.12 shows the settings of the sensitivity level, trust level and sharing policy that associated controllers can assign.

Table B.12: Viewing baseline probability of revoking the owner's initial decision for one contributor/originator when distance is ≥ 2

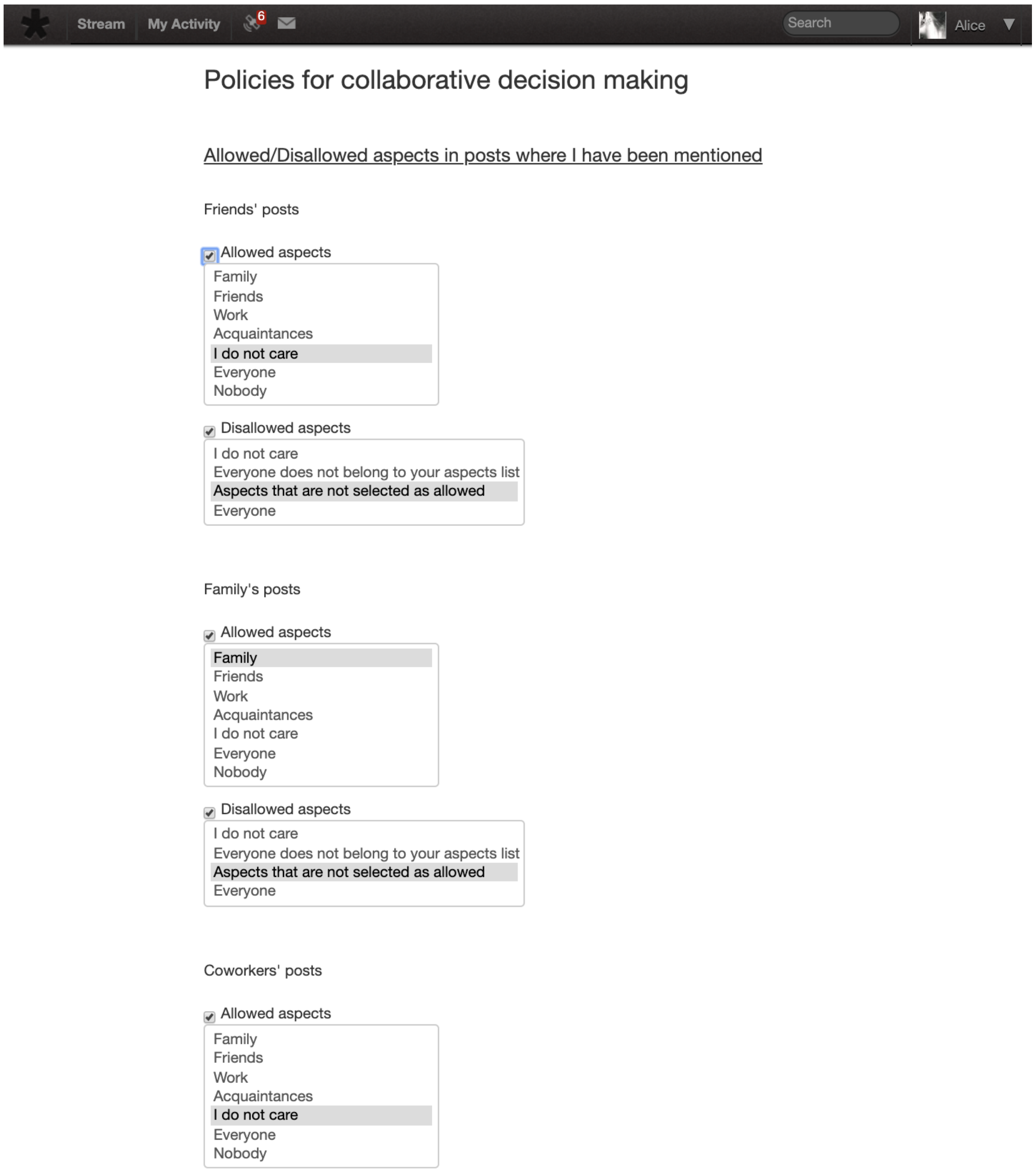
Viewing — Contributor/Originator														
Output	4	3.75	3.5	3.25	3.0	2.75	2.5	2.25	2.0	1.75	1.5	1.25	1.0	0.75
Frequency	1	2	4	6	9	10	10	8	6	3	1	0	0	0
Revocation number	0	0	0	0	1	3	7	13	22	32	42	50	56	59
Baseline Probability	0%	0%	0%	0%	1.6%	5%	11.6%	21.6%	36.6%	53.3%	70%	0%	0%	0%

Table B.13: Sharing baseline probability of revoking the owner's initial decision for one contributor when distance ≥ 2

Sharing — Contributor						
Output	2	1.5	1.25	1.0	0.75	0.5
Frequency	1	1	1	1	0	0
Revocation number	0	0	0	1	2	3
Baseline Probability	0%	0%	0%	25%	0%	0%

Table B.14: Sharing baseline probability of revoking the owner's initial decision for one originator when her weight is 0.25

Sharing — Originator							
Output	2	1.5	1.25	1.0	0.75	0.5	0.25
Frequency	1	1	1	1	0	0	0
Revocation number	0	0	0	1	1	2	3
Baseline Probability	0%	0%	0%	25%	0%	0%	0%



Policies for collaborative decision making

Allowed/Disallowed aspects in posts where I have been mentioned

Friends' posts

[✓] Allowed aspects

- Family
- Friends
- Work
- Acquaintances
- [✓ I do not care]
- Everyone
- Nobody

[✓] Disallowed aspects

- I do not care
- Everyone does not belong to your aspects list
- [✓ Aspects that are not selected as allowed]
- Everyone

Family's posts

[✓] Allowed aspects

- [✓ Family]
- Friends
- Work
- Acquaintances
- I do not care
- Everyone
- Nobody

[✓] Disallowed aspects

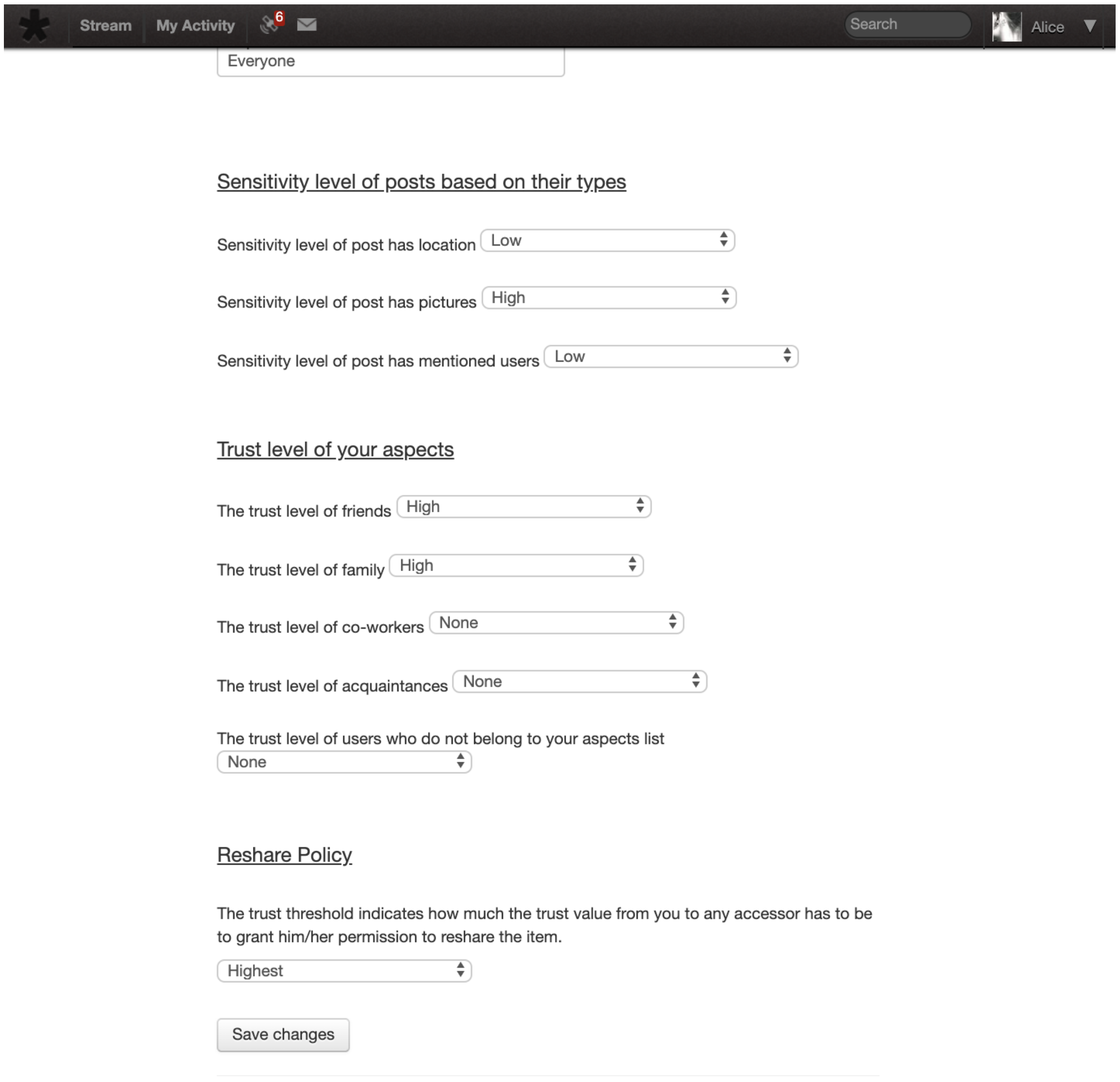
- I do not care
- Everyone does not belong to your aspects list
- [✓ Aspects that are not selected as allowed]
- Everyone

Coworkers' posts

[✓] Allowed aspects

- Family
- Friends
- Work
- Acquaintances
- [✓ I do not care]
- Everyone
- Nobody

Figure C.11: User interface to specify allowed and disallowed users



Everyone

Sensitivity level of posts based on their types

Sensitivity level of post has location: Low

Sensitivity level of post has pictures: High

Sensitivity level of post has mentioned users: Low

Trust level of your aspects

The trust level of friends: High

The trust level of family: High

The trust level of co-workers: None

The trust level of acquaintances: None

The trust level of users who do not belong to your aspects list: None

Reshare Policy

The trust threshold indicates how much the trust value from you to any accessor has to be to grant him/her permission to reshare the item.

Highest

Figure C.12: User interface to assign sensitivity levels to types of items and trust levels on other users

Appendix D.

Table D.15 present some of the test cases we have executed: the first column shows the performed 845 action, the second column what was expected as outcome, and the last one whether the test passed or failed, or a clarifying comment.

Table D.15: Some of the test cases performed to validate our prototype.

Action	Expected Outcome	Result
Alice posted a picture and tagged her friend Bob. Alice permitted her family aspect and has low sensitivity level for a post that contains a picture and high trust level for her family members. Bob, who is the stakeholder, permitted his friends' aspects and has high sensitivity level for post that contains a picture. He has medium level of trust with his friends. In our social network, we have Charlie who is a family member of Alice and a friend of Bob.	The collaborative decision regarding Charlie's access is aggregated from Alice's and Bob's privacy policies. Charlie will access the picture since he is a family member of Alice and a friend of Bob.	Passed
Bob displayed a post that contains a picture and mentions his friend Alice. Bob's privacy settings are permitting only his family aspects and denying everyone else such as friends and coworkers. In addition, he specified a high sensitivity level for posts that contain pictures; and medium trust level for his family and friends and low trust level for his coworkers. Alice, who is the stakeholder, denied her coworkers as well and has no trust with her coworkers. In our social network, we have Frank who is one of Alice's and Bob's coworkers.	The collaborative decision regarding Frank's access is aggregated from Alice's and Bob's privacy policies. Frank will not access the post since he is one of Alice's and Bob's coworkers.	Passed

Table D.15 – continued from previous page

Action	Expected Outcome	Result
Bob uploaded a post that contains a picture and mentions his friend Alice. Bob's privacy settings are permitting only his family aspects and denied everyone else such as friends and coworkers. In addition, he specified a high sensitivity level for posts that contain pictures; and medium trust level for his family and friends. Alice, who is the stakeholder, permitted her friends and family aspects and denied her coworkers. Regarding trust level setting, she assigned medium level of trust to her friends and family aspects. In our social network, we have Charlie who is a friend of Bob and a family member of Alice.	The collaborative decision regarding Charlie's access is aggregated from Bob as denier and Alice as authorizer. Charlie will not access the post since the permitted decision value is equal to denied decision value.	Passed
Frank posted a post that contains a location and mentions his coworker Bob. Frank's privacy settings are allowing his coworkers and friends aspects to view the post and denied his family. Frank assigned the medium sensitivity level for any post includes a location and the high trust level for his friend aspect. Bob, who is the stakeholder, allowed his coworkers to view his coworker's post and denied everyone else from his aspects list such as family and friends. Bob assigned the medium sensitivity level for any post includes a location and the high trust level for his family aspect. In our social network, we have Grace who is a friend of Frank and a family member of Bob.	The collaborative decision regarding Grace's access is aggregated from Bob as denier and Frank as authorizer. Grace will access the post since the permitted decision is greater than the denied decision.	Passed

Table D.15 – continued from previous page

Action	Expected Outcome	Result
Judy posted a family picture and tagged her family member Charlie. Judy wants to give access to this picture only to her family aspect and denies other aspects such friends and coworkers. The rest of her privacy settings are as follows: high sensitivity level for any post that contains a picture and medium trust level for her friends. Charlie, who is stakeholder, wants to allow all his family members and friends to view the family picture. He has high trust level for his friends aspect and medium sensitivity level for any post that contains pictures. In our social network, we have Eve who is a friend of both Judy and Charlie.	The collaborative decision regarding Eve's access is aggregated from Judy as the denier and Charlie as the authorizer. Eve will not access the family picture since the denied decision is greater than the permitted decision.	Passed
Niaj posted a picture of a work event and tagged her coworkers Mike and David. Niaj wants her coworkers and friends to access this post and has low sensitivity level for such a type of post. She highly trusts her friends aspect. The first stakeholder Mike wants his coworkers and friends to view his work pictures. Regarding sensitivity and trust settings, he has medium level for any post containing pictures and medium trust level for his friends aspect. The second stakeholder David gives access to his coworkers to view the post and denies his friends aspect. His privacy setting is high sensitivity level and no trust for his friends aspect. In our social network, we have Ivan who is a friend of David, Niaj and Mike.	The collaborative decision regarding Ivan's access is aggregated from David as the denier and Niaj and Mike as the authorizers. Since David has maximum level of sensitivity and no trust with Ivan then we apply veto rights to deny to access the post. Ivan will not access the post.	Passed

Table D.15 – continued from previous page

Action	Expected Outcome	Result
Mike and Judy became friends. Adding this friendship while we are in the same social and privacy settings of action 6. As a result, Judy is a friend of Mike.	The decision regarding Judy's access is aggregated from Mike as the authorizer. Judy will access the working event picture that was posted by Niaj since she is permitted by Mike and no one denied her.	Passed
Bob posted a picture and tagged his friend Alice and cousin Judy. He would like his family members and friends to access this picture and medium level of sensitivity has been set to any post that contains pictures. Bob has high trust for his friends. Alice, who is a stockholder, wants only her friends to access her friends' pictures and denied other aspects such as family and coworkers. The sensitivity level and trust level of friends have been set to medium level and low, respectively. On the other hand, Judy wants her family and friends to view the picture. She has no sensitivity level for her family picture and specified medium trust level for her family aspect. Charlie is a friend of Alice and a family member of Judy and Bob.	The collaborative decision regarding Charlie's access is aggregated from Alice as denier and Judy and Bob as authorizers. Charlie will access the picture since the permitted decision is greater than denied decision.	Passed
Niaj posted a message in her stream showing her current location and mentioned her friend Ivan and close colleague David. For this post, she wants only her friends to view; and assigned a low sensitivity level for a such type of post. She assigned low trust level for her coworkers' aspect. The stakeholders Ivan and David have no privacy preferences regarding this post. In our social network, we have Mike who is a friend of Ivan and a colleague of Niaj and David.	Since the stakeholders David and Ivan have no privacy preferences, the decision regarding Mike's access is aggregated from Niaj as the denier. As a result, Mike will not be able to view the post.	Passed

Table D.15 – continued from previous page

Action	Expected Outcome	Result
Heidi posted a picture and tagged her friend Grace and close colleague Ivan. She wants only her family to view this picture; and assigned a high sensitivity level for it. Also, she assigned a high trust level for her family aspect. The stakeholders Ivan wants to give an access to her friends. The sensitivity level and trust level of friends have been set to high and medium, respectively. The second stakeholder Grace, has no privacy preferences regarding this picture. In our social network, we have Judy who is a family member of Heidi; and Mike, David and Niaj who are friends of Ivan.	The decision regarding Judy's access is aggregated only from Heidi as authorizer, since Judy has not been mentioned in other associated controllers' privacy settings. On the other hand, the decision regarding Ivan's friends (Mike, David and Niaj) access is aggregated only from Ivan as the authorizer, since they have not been mentioned in other associated controllers' privacy settings. As a result, Judy, Mike, David and Niaj will be able to access the picture.	Passed
Oscar became a friend with Alice, so he is now a member in friends aspect. Consequently, if we recall action 1 and remain the privacy settings as they are.	The decision regarding Oscar access is aggregated from Alice as the authorizer. Oscar will access the post.	Oscar saw Alice's posts that are performed after the friendship is created between them.

Table D.15 – continued from previous page

Action	Expected Outcome	Result
Judy preformed a post and mentioned her family members Charlie and Bob. She wants only her family members to access this post. The sensitivity level, trust level of family and trust level of friends are set to medium, high and high respectively. The stakeholder Charlie, who wants to give an access to his friends only and denied everyone else (i.e., every user who do not belong to his aspects list), assigned high sensitivity level for any post containing a picture; medium trust level for his friends aspect; and no trust level for users who do not belong to his aspects list (i.e., users who have no direct relationship with him). Bob, as the stakeholder, wants his friends to view his family's posts. The sensitivity level and trust level of friends have been set to low and medium, respectively. In our social network, we have the following: 1) Heidi who is family member of Judy and has no relationship with Charlie; 2) Eve who is a friend of Charlie and Judy; 3) David who is a friend of Charlie; 4) Alice who is a friend of Bob and a family member of Charlie.	The collaborative decisions regarding Heidi, Eve, David and Alice are aggregated as follows: • Heidi will be denied since her denied decision, which is computed from Charlie's policy, is greater than permitted decision, which is computed from Judy's policy. • Eve will be allowed to view the post since her permitted decision, which is computed from Charlie's policy, is greater than denied decision, which is computed from Judy's policy. • David will be allowed to access the post since he is permitted by Charlie and no one denied him. • Alice will be denied since her denied decision, which is computed from Charlie's policy, is greater than permitted decision, which is computed from Bob's policy.	Passed

Table D.15 – continued from previous page

Action	Expected Outcome	Result
Alice posted a picture and tagged her friend Bob and her family member Charlie. She wants only her family members to access this picture and prevents her friends. The sensitivity level, trust level of friends are low and high, respectively. Her reshare policy, which is a trust threshold indicates how much the trust value from her to any user has to be to grant him/her permission to reshare the post, is highest. Charlie would like his friends to access the post and assigned low sensitivity level for any post contains picture. Regarding the trust, she specified for her friends aspect a medium level and for reshare policy a low level. Bob as the stakeholder, would like his coworkers to view his friends' post and assigned medium sensitivity level for a such type of post. The trust level for users who do not belong to his aspects list has been set to low level; and reshare policy has been placed to medium level. In our social network, David is a friend of Alice and Charlie; and does not have any direct relationship with Bob (i.e., does not belong to Bob aspects list).	The collaborative decision regarding David's access is aggregated from Charlie as authorizer and Alice as denier. David will access the picture since the permitted decision is greater than the denied decision. However, he will not be able to reshare the post. The decision regarding resharing is aggregated form Alice, Charlie and Bob reshare policies.	Passed

Table D.15 – continued from previous page

Action	Expected Outcome	Result
Mike pasted a picture in his stream and tagged his coworkers Niaj and David and his friends Judy and Ivan. In this action, we focus on the reshare action for the stakeholders (i.e., which one of the stakeholders will be allowed to reshare the picture). Starting with the owner's policy, the sensitivity and reshare policy have been set to medium level. The trust levels for both his coworkers and friends aspects have been set to a medium level. David assigned high level for both the sensitivity and the reshare policy. Moreover, he has no trust for users who do not belong to his aspects list. The stakeholder Niaj, defined low level for both the sensitivity level and reshare policy. From Ivan's side, the sensitivity level and reshare policy have been set to a high and low, respectively. Finally, Judy specified a low level of trust for reshare policy and medium level of sensitivity for a such post type. In addition, she assigned a high trust level for her friends' aspect and medium level for users who do not belong to her aspects list. In our social network, Judy is a friend of Mike but does not have any relationship with Ivan, David or Niaj.	The collaborative decision regarding Judy's resharing is aggregated form Mike, Ivan, David and Niaj. She will be to view the picture as stakeholder but will not be allowed to reshare it.	Passed