Preserving Privacy as Social Responsibility in Online Social Networks

DILARA KEKULLUOGLU, Bogazici University
NADIN KOKCIYAN, King's College London
PINAR YOLUM, Bogazici University & Utrecht University

Online social networks provide an environment for their users to share content with others, where the user who shares a content item is put in charge, generally ignoring others that might be affected by it. However, a content that is shared by one user can very well violate the privacy of other users. To remedy this, ideally, all users who are related to a content should get a say in how the content should be shared. Recent approaches advocate the use of agreement technologies to enable stakeholders of a post to discuss the privacy configurations of a post. This allows related individuals to express concerns so that various privacy violations are avoided up front. Existing techniques try to establish an agreement on a single post. However, most of the time, agreement should be established over multiple posts such that the user can tolerate slight breaches of privacy in return of a right to share posts themselves in future interactions. As a result, users can help each other preserve their privacy, viewing this as their social responsibility. This article develops a reciprocity-based negotiation for reaching privacy agreements among users and introduces a negotiation architecture that combines semantic privacy rules with utility functions. We evaluate our approach over multiagent simulations with software agents that mimic users based on a user study.

CCS Concepts: • **Security and privacy → Social network security and privacy**; • **Computing methodologies → Multi-agent systems**;

Additional Key Words and Phrases: Privacy, agreement, negotiation

ACM Reference format:
Dilara Kekulluoglu, Nadin Kokciyan, and Pınar Yolum. 2018. Preserving Privacy as Social Responsibility in Online Social Networks. *ACM Trans. Internet Technol.* 18, 4, Article 42 (April 2018), 22 pages.
https://doi.org/10.1145/3158373

1 INTRODUCTION

Online social networks are providing an immediate platform for users to share content online. A user can share content about herself, about third-party entities, and about friends in the online social network. Although this capability creates an effective mechanism to interact, it also allows tremendous privacy violations to take place. A friend shares a picture of you on a beach, and all

Preliminary ideas for this work were presented as a short paper at ECAI 2016 and at the ECAI Workshop on Artificial Intelligence for Privacy and Security (PrAISe'16) [10].

Authors' addresses: D. Kekulluoglu, Bogazici University, Department of Computer Engineering, Istanbul, 34342, Turkey; email: dilara.kekulluoglu@boun.edu.tr; N. Kokciyan, King's College London, London, WC2B 4BG, United Kingdom; email: nadin.kokciyan@kcl.ac.uk; P. Yolum, Bogazici University, Department of Computer Engineering, Istanbul, 34342, Turkey, and Utrecht University, Utrecht, Netherlands; email: p.yolum@uu.nl.

https://doi.org/10.1145/3158373

of your colleagues learn where you were for your holiday. A parent shares a childhood video of you, and all of your friends learn how shy you were as a kid. The information that the posts reveal may very well be private and are shared without your own consent, creating a privacy breach. In current online social networks, such as Facebook, a common way to deal with this is for the user to complain to the social network administration and ask the content to be removed [26]. However, by the time the content is removed (if at all), many people might have seen it already. Ideally, it would be best if such content was not shared in the first place.

To facilitate this, the users of the online social network need to collaborate to help each other preserve their privacy. This reflects a paradigm shift in terms of who is responsible for preserving the privacy of a user. In traditional systems, the responsibility lies with the system administration; in online social networks, the responsibility lies with the society (i.e., the users of the system). A user may help preserve a friend's privacy by not sharing some of the posts that she intends to share or sharing them with a limited set of people. Obviously, in that case, the user is left at a disadvantage because not all of the people that she would like to reach will see her post, but the benefit is that she is helping a friend. This is indeed the underlying idea of *social responsibility* [1], where a person takes action for the benefit of others in the society. Although social responsibility is a concept that lives mostly offline, there is substantial evidence to show that users in an online social network are willing to collaborate, such as to ensure that their friends' privacy is preserved [16, 26]. To realize the idea of social responsibility in an online world, computational models are needed to facilitate the collaboration. The proposed work here uses the idea of reciprocity [5] to realize social responsibility.

Recent work on privacy management has focused on applying agreement technologies to solve privacy problems before they take place. One line of work uses multiagent negotiation techniques to resolve privacy conflicts among users [19, 28]. The general idea is to enable users (or their software agents) to negotiate how to share the content before it is shared. Mester et al. [19] employ a semantic approach, where users' agents have privacy rules and negotiate based on the firing of the rules. Their semantic representation is powerful and gives insight on what each agent expects. However, their proposed decision-making scheme is simplistic and assumes that one agent will always accept a second agent's requests. Such and Rovatsos [28] employ a utility-based approach where each agent has a utility function that assigns a utility to a content based on its user's privacy expectations. Although their approach provides decision-making capability for many cases, they do not enable agents to reason on why their privacy is being violated. Further, both approaches assume that negotiation is being performed on a single content item and cannot account for ongoing interactions. However, preserving privacy is not a single state of affairs but a state that needs to be maintained throughout. It has been observed that users build reciprocal trust in online social networks and respect others as much as others respect them [16]. Hence, it is of utmost importance to consider repeated interactions to study privacy leakages.

Accordingly, this article develops a hybrid negotiation architecture where privacy domain and rules are represented semantically, but the agents can benefit from utility functions in reaching decisions. A key idea in the architecture is the use of a reciprocity mechanism to imitate social responsibility such that the agents are equipped with incentives to respect each other's privacy. This is done by keeping track of which agent is helpful using a credit system. When agents help others in preserving their privacy, their credit increases so that later they can ask others to help them. We evaluate this mechanism under various settings and in relation to existing strategies using multiagent simulations. Our results show that when social responsibility is in place, the society as a whole can preserve privacy better than one where a user tries to do it on her own.

The rest of this article is organized as follows. Section 2 explains our user study to understand various privacy constraints of users. Section 3 describes our hybrid negotiation architecture with an emphasis on semantic representation and decision making. Section 4 introduces various negotiation strategies for single interactions. Section 5 develops our reciprocal strategy that uses reciprocity as a basis to preserve privacy over interactions. Section 6 explains our user study and evaluates the proposed mechanism in multiagent simulations. Section 7 gives our results on preserving privacy using various strategies. Finally, Section 8 compares our work to related work in the literature.

2 UNDERSTANDING PRIVACY CONCERNS

To understand the privacy concerns of the online social network users, we conducted a user study. Understanding privacy concerns of real users is important because it enables us to develop realistic representations.

In the user study, we worked with 10 participants (five female and five male) that we chose from people that we know in person. Five participants are younger than 40 years, whereas the other five participants are older than 40 years. All participants are from different professional backgrounds (photography, education, etc.). All participants use social networking sites on a daily basis. However, they share posts at most once per week.

In conducting the user study, we are inspired by the work of Sleeper et al. [24]. In their work, the authors interviewed participants to understand the types of content that the participants choose to self-censor in Facebook. They report that users do not want to share content if it is personal (e.g., personal updates). Hence, we chose 10 pictures that describe personal situations in various contexts. All participants were shown five of these pictures and were asked questions about how they would like this picture to be shared if a designated individual in the picture was herself. The study was done as an interview by following a guideline for questions. The guideline consisted of three parts.[1] In the first part, we show a picture and ask each participant whether she would be happy to share this picture. We also want to understand cases where the participant would not prefer sharing the content. In the second part, we inform the participant that a friend of hers had shared this picture without consulting the participant. Here, we try to understand how the participant feels when another user shares a content about her. In the third part, we ask participants whether they would consider privacy concerns of other users. Hence, we aim to learn under which circumstances a user would consider the privacy of other users.

The privacy concerns of the participants are mostly influenced by the metainformation that comes along with the content. For example, some participants have privacy concerns that are based on the context of a content, whereas others prefer not sharing a content with people who are connected to these via various relationships. The privacy concerns that are derived from the user study are as follows:

(1) *Mood of a picture*: The mood of a picture plays an important role in deciding to share a picture. Some of the users were glad to share a picture if its mood is happy and festive. However, if the picture reminded people of a feeling of depression, then the users did not want that picture to be shared. <*C1: If the mood of the picture is depressed, then the user does not prefer others to see this picture.*>
(2) *Bar pictures*: There are two privacy concerns regarding contents taken in a bar context: <*C2.1: If the picture is taken in a bar, then the user does not prefer her family members to see it.*>, <*C2.2: If the picture is taken in a bar, the user does not prefer her friends who do not consume alcohol to see it.*>.

[1]The pictures and the guideline are available online at http://mas.cmpe.boun.edu.tr/negotiation.

(3) *Vacation pictures*: When we showed users beach pictures where they are in swimsuits, some of them did not want their work-related friends to see such pictures. Work-related friends include not only colleagues but whoever is connected to the user in a work context. For example, in the education context, a teacher, who was one of the participants, did not want her students and their parents to see her beach pictures. *<C3.1: If the picture shows the user in a beach, then the user does not prefer her work-related friends to see it.>* Another concern related to vacation pictures was that some of the users wanted to keep their location secret. One reason for this is when users go on a vacation in a city, they may not want to meet their friends who live in that city. *<C3.2: If the picture is taken in a city different from where the user lives, then the user does not prefer her friends who live in that city to see it.>*
(4) *Event pictures*: When we showed participants a wedding picture, they did not want people who were not invited to the event to see this picture. We generalized this concern to apply to any event organized by the user. *<C4: If the user is the organizer of an event, then the user does not prefer people who are not invited to this event to see event-related pictures.>*
(5) *Adult content*: Users did not want to be seen in pictures with subliminal sexual content. *<C5: If the user is shown in an adult content, then the user does not prefer others to see this picture.>*
(6) *Leisure pictures*: Some users do not want to share content that is in leisure context with their colleagues. *<C6: If a picture is in leisure context, then the user does not prefer her colleagues to see this content.>*
(7) *Protest pictures*: Some users did not want to reveal their protest pictures to their friends who hold an opposing view with them. The reason is that people do not want to damage their relationships with others as a result of their political differences. *<C7: If a picture shows the user in a protest for an issue, then the user does not prefer her friends that hold an opposing view on that issue to see it.>*
(8) *Minor rights*: Some users do not want to share pictures of minors without their parents' consent. In other words, people do not want to make a sharing decision on behalf of the children of others. *<C8: If the user is in a picture that shows a minor, then the user does not prefer others to see this picture unless the minor's parents give permission to share the picture.>*
(9) *Removal of specific people*: Independent of the context of a content, some users would like to block certain users from the audience. *<C9: The user does not prefer a content to be seen by some users, who are part of the audience of that content.>*

3 NEGOTIATION ARCHITECTURE

We use PriNego [19] as the basis for the semantic aspects of negotiation. PriNego proposes an agent-based negotiation framework for privacy. An agent is a software entity that can act on behalf of a user to preserve her privacy. Each agent has information about the social network, such as the friends of the user. This information is captured in an ontology that is represented in Web Ontology Language (OWL) [18]. In an ontology, a class defines a concept in a domain. For example, the `Agent` class denotes a user agent in the social network domain. Each class may have instances. For example, `:alice` is an instance of `Agent`, which is specified as `Agent(:alice)`. Properties are a way of connecting classes to other classes or data values (e.g., name of an agent). An instance of an object property relates two instances to each other by a specific relationship. For example, *isFriendOf*(`:alice`, `: bob`) represents that `:alice` and `: bob` are friends of each other. An instance of a data property can be used to describe instance attributes. For example, *isAdultContent*(`: pic`, true) represents that the instance `: pic` is an adult content. We denote a `Concept` with text in monospaced format, a *property* with italic text, and an `:instance` with a colon followed by text in monospaced format.

A social network consists of a set of users who are connected to each other via various relationships. *isConnectedTo* is a property that connects two agents. The subproperties of *isConnectedTo* (*isColleagueOf*, *isFriendOf*, and *isPartOfFamilyOf*) allow the agents to describe relations in more detail. An `Agent` sends a `PostRequest` to other agents before sharing a post. A `PostRequest` may contain some content such as `Text`, `Medium`, or `Location` information. Additionally, *hasText*, *hasMedium*, and *hasLocation* are used to relate corresponding concepts to `PostRequest`. A person may be mentioned in a text (*mentionsPerson*) or tagged in a medium (*includesPerson*). Each `PostRequest` is associated with an `Audience` concept via *hasAudience*. An audience is a group of agents, and *hasMember* describes agents that are members of an audience.

Most of the time, privacy concerns depend on the context of a post as shown in our user study. However, the factual information such as time and location is not enough to determine the context [22]. A post may be associated with many contexts that are represented as `Context` instances in the ontology. In this work, we assume that the each agent is able to decide for itself what the context of the post is, using its ontology or external sources. For example, an agent can infer that a picture is in `Learning` context by using the picture attributes. We use *isInContext* to associate context information to a postrequest.

3.1 Privacy Rules as Semantic Rules

The privacy concerns of the users should be defined semantically so that agents can reason about them in an automated way. In this work, each agent captures its user's privacy concerns as semantic rules represented with a Semantic Web Rule Language (SWRL) rule [8]. SWRL rules are encoded in the agent's ontology and contribute into the ontological reasoning. Each SWRL rule is of the form *Body* → *Head*, which means that if the body holds, then the head must also hold. The body and the head consist of conjunctions of atoms. Here, atoms are of the form `C`(x) and *P*(x,y). `C` is a class name (e.g., `Leisure`), and *P* is a property name (e.g., *isInContext*), which are defined in the ontology. x and y are either variables prefixed with a question mark (e.g., ?ctx), instance names, (e.g., `:alice`) or literals (e.g., true). Semantic rules may depend on a specific location, context, relationship, or any combination of these. Consider the scenario in Example 1, which we will use as our running example.

Example 1. Bob wants to share a picture of Alice with everyone. This picture is in an eating and drinking context. Alice does not want her colleagues to see her leisure pictures. Moreover, she does not want Errol to see any of her pictures.

When a user (e.g., Bob) wants to share a post, the user agent finds users that would be affected by that post (e.g., Alice) and contacts those users' agents with a postrequest. An agent may reject a postrequest according to the user's privacy concerns, which is described via a *rejects* property in its ontology. A privacy rule is violated when the conditions specified in the body of this rule hold—that is, the agent infers a *rejects* predicate in its knowledge base. An agent can provide rejection reasons through the use of *rejectedIn* and *rejectedBecauseOf* properties. For example, a user may reject a postrequest because of an audience, which includes undesired people or a medium where the user did not like herself. Conversely, the agent can choose not to provide a reason. In this case, *rejects* is the only predicate observed in the head of a privacy rule. However, if an agent would like to give reasons about a rejection, then the *rejectedIn* property is used to declare whether the rejection is caused by a medium, a posttext, or an audience. Furthermore, *rejectedBecauseOf* is used to specify more details about the rejection. For example, an audience can be rejected in a

Table 1. Privacy Rules (P) as SWRL Rules

$P^{6}_{A_1}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, ?m), `Leisure`(?ctx), *hasMedium*(?pr, ?med), *isInContext*(?med, ?ctx), *isColleagueOf*(?m, `:alice`) → *rejects*(`:alice`, ?pr), *rejectedIn*(?aud, ?pr), *rejectedBecauseOf*(?aud, ?m)
$P^{3}_{A_2}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, `: errol`) → *rejects*(`:alice`, ?pr), *rejectedIn*(?aud, ?pr), *rejectedBecauseOf*(?aud, `: errol`)
$P^{w_1}_{x_1}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, ?m), *isConnectedTo*(?m, `: x`), *livesIn*(?m, ?city), *hasMedium*(?pr, ?med), `Vacation`(?ctx), *inCity*(?ctx, ?city), *isInContext*(?med, ?ctx), *includesPerson*(?med, `: x`) → *rejects*(`: x`, ?pr), *rejectedBecauseOf*(?aud, ?m), *rejectedIn*(?aud, ?pr), *rejectedIn*(?med, ?pr), *rejectedBecauseOf*(?med, ?ctx)
$P^{w_2}_{x_2}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, ?m), *isConnectedTo*(?m, `: x`), *hasMedium*(?pr, ?med), `Event`(?e), *isTakenIn*(?med, ?e), *isOrganizedBy*(?e, `: x`), *notInvitedTo*(?m, ?e), *includesPerson*(?med, `: x`) → *rejects*(`: x`, ?pr), *rejectedBecauseOf*(?aud, ?m), *rejectedIn*(?aud, ?pr), *rejectedIn*(?med, ?pr), *rejectedBecauseOf*(?med, ?e)
$P^{w_3}_{x_3}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, ?m), *hasMedium*(?pr, ?med), *isAdultContent*(?med, true), *includesPerson*(?med, `: x`) → *rejects*(`: x`, ?pr), *rejectedIn*(?med, ?pr), *rejectedBecauseOf*(?aud, ?m), *rejectedIn*(?aud, ?pr)

postrequest because of an undesired person in the audience. Or a content item can be rejected in a postrequest because of the location where the content item was taken.

The user study shows that a user might have various privacy constraints, but these might not be equally important. To capture the fact that a rule is more important than a second rule, we associate a weight with each rule. Each rule is denoted as $P^{w}_{X_i}$, which is the ith rule of agent X and w is the weight of this rule. There could be two ways in which privacy rules and weights can be incorporated into the system. The first one is enabling users to enter their rules in a user-friendly interface that is independent of the underlying rule language. We have developed an Android application that asks questions to users about their preferences and generates the privacy rules for the users. The users choose the context, location, user groups, and such to create the privacy rules, and assign weights to them manually [19]. However, this approach assumes that users know their privacy preferences, and can describe and prioritize them. In another approach, the privacy rules can be learned over time. For example, by the use of machine learning techniques, an agent can analyze the previous posts of the user and extract privacy preferences automatically as has been successfully done in various works [3, 11]. In this approach, the human effort is minimized, as each agent learns the privacy rules from the user's actions in the online social network.

We represent a subset of the privacy concerns derived from the user study as SWRL rules as shown in Table 1. For example, Alice has two privacy concerns: P_{A_1} and P_{A_2}. P_{A_1} states that Alice's agent (`:alice`) rejects any postrequest if a colleague of her is in the audience of this postrequest, which is in leisure context. P_{A_2} states that if `: errol` is in the audience of a postrequest, then `:alice` rejects it. Here, we see that P_{A_1} is more important than P_{A_2} since the weight of P_{A_1} is higher. P_{A_1} is an example for the concern $C6$, whereas P_{A_2} represents the concern $C9$. In the first rule, the context of the medium and the relationship type are the conditions specified by Alice. The second does not depend on any attribute of the content being shared. Alice does not want to share any content with `: errol`. The rules $P^{w_1}_{x_1}$, $P^{w_2}_{x_2}$, and $P^{w_3}_{x_3}$ mimic the privacy concerns $C3.2$, $C4$, and $C5$, respectively. We have decided not to express some of the privacy concerns ($C2.2$, $C7$, and $C8$) in our model. In *C*2.2, it is hard to find whether a person consumes alcohol or not from the social media profile of that person. This requires using more techniques to have a user profile of each person in the social network of the user. In *C*7, even if it may be possible to understand the protest context, it would be difficult to find views of the users on the particular protest issue. This requires analyzing the user's posts and the interactions more deeply. In *C*8, one problem is how to detect the minor and then find out the parents of that minor. In all three cases, if we can provide the missing information from other sources, then it is possible to update the ontology and express these privacy concerns as well. We leave this as future work.

3.2 Decision Making

Following the running example, when Bob initiates a negotiation with Alice, Alice evaluates Bob's postrequest according to her rules and decides whether to accept or make a counteroffer. This is followed by a similar move from Bob. In other words, the negotiation continues in a turn-taking fashion. Hence, a negotiation requires many iterations so that the agents can reach a final decision together. The evaluations done to decide whether to accept a proposal and to create a new counteroffer constitutes the *negotiation strategy* of an agent. Here, we require each agent to have a utility function that it can use to make this decision. The utility functions help the initiator and negotiator agents decide how much a postrequest supports their privacy concerns. The utility functions should be in line with the privacy requirements of their users. These functions take a value between 0 and 1, where larger numbers are preferred. The agent that initiates the negotiation (i.e., the initiator) will have a different utility function than an agent that negotiates for her privacy (i.e., the negotiator), as both agents have different capabilities.

3.2.1 The Initiator Agent. An initiator agent (in) is responsible for initializing the negotiation with other agents, collecting responses, updating a postrequest, and making a decision about it. It can choose to share the post, continue, or terminate the ongoing negotiation according to its utility function ($u^{in}_{p_i}$). During negotiation, if all agents agree on sharing the postrequest, then the initiator agent shares the post. Otherwise, it will try to update the postrequest according to rejection reasons of others. If the computed utility of the updated postrequest is above the threshold, the initiator agent will send the updated postrequest to relevant agents to have their consent. Otherwise, the initiator agent will not share the post, as the agents cannot reach an agreement. We define such an evaluation function in Definition 3.1.

Definition 3.1 (Evaluation of the Initiator Agent). Given a postrequest p_i, an initiator agent in makes a decision about p_i regarding its threshold t^{in}.

$$eval^{in}(p_i) = \begin{cases} share & \text{if agents accept } p_i \\ continue\ negotiation & \text{if } t^{in} \leq u^{in}_{p_i} \\ not\ share & \text{otherwise} \end{cases}$$

Before starting a negotiation, the initiator agent wants to share a content with an initial audience (a_0). However, this audience can only become smaller since other agents can choose to remove some users from the audience according to their privacy rules. Hence, the utility of the initiator agent should be high if it can share the content with most of the users included in a_0. We propose the following utility function for the initiator agent. Note that the initiator agent creates a postrequest that does not violate its privacy rules. Hence, the privacy rules are not included in the utility calculation.

$$u^{in}_{p_i} = 1 - \frac{|a_0 - a_i|}{|a_0|} \tag{1}$$

In Equation (1), the initiator agent considers how many users are removed from the audience of the original postrequest (a_0). The ith time the negotiator agent sends a rejection reason (i.e., ith iteration), in updates the postrequest such that it removes users who violate the privacy of the negotiator agents from a_0. This updated audience is called a_i, where $a_i \subseteq a_0$. Thus, the initiator agent's utility will decrease if users are removed from a_0 regarding rejection reasons of others.

3.2.2 The Negotiator Agent. A negotiator agent (ng) is responsible for evaluating a postrequest (p_i) and making a decision about this postrequest based on its utility ($u^{ng}_{p_i}$). For this, a negotiator agent checks if its utility value is above her acceptable threshold. In case it wants to reject it, it also provides rejection reason(s) depending on the strategy that it follows. We define such an evaluation function in Definition 3.2.

Definition 3.2 (Evaluation of the Negotiator Agent). Given a postrequest p_i, a negotiator agent ng makes a decision about p_i regarding its threshold t^{ng}.

$$eval^{ng}(p_i) = \begin{cases} accept & \text{if } t^{ng} \leq u^{ng}_{p_i} \\ reject\ with\ reasons & \text{otherwise} \end{cases}$$

Intuitively, it is best for the negotiator agent if the privacy rules with higher importance are violated by a small number of users. To capture this in a utility function, two principles are adopted. First, the violated privacy rules with higher weights decrease the utility more than the rules with lower weights. Second, if the number of unwanted people accessing the post is high, then the utility should decrease more. To realize these two principles, we propose the following utility function.

$$u^{ng}_{p_i} = u_{max} - \left(\frac{\sum_{k=1}^{K} u_{r_k}}{w_{max} \times v_r} \right) \tag{2}$$

$$u_{r_k} = w_{r_k} \times v_{r_k} \tag{3}$$

In Equation (2), we show how a negotiator agent (ng) computes a utility upon receiving a postrequest (p_i) at the ith iteration. u_{max} is the maximum utility that can be computed for a postrequest. w_{max} is the maximum weight of a privacy rule. v_r is the total number of users who violate privacy rules initially. In this work, u_{max} and w_{max} are set to 1 and 10, respectively. K is the total number of privacy rules of the negotiator agent. u_{r_k} is the utility value of kth rule, which is calculated as shown in Equation (3). w_{r_k} is the weight of the rule r_k, which takes a value between 1 and 10, with 10 being higher importance. The user sets this value regarding the importance of her privacy concerns. Aside from the weight of the rule, it is also important to consider the number of users (v_{r_k}) violating a rule. If a privacy rule is not violated, then the utility value of such a rule will be 0. This utility function reduces the utility according to the weights of the violated privacy rules and the number of people violating the privacy of the negotiator agent's user.

4 NEGOTIATION STRATEGIES FOR PRIVACY

In PriNego, there is only one strategy for the agents to use. The negotiator agent evaluates a postrequest and sends all rejection reasons in case the postrequest is not acceptable. The initiator agent then revises the postrequest according to the rejection reasons provided by the negotiator agent. Although this is an advantageous outcome for the negotiator agent, it puts the initiator agent at a disadvantage. For example, if the initiator agent wants to show a picture to 10 people but the audience is reduced to 3 people as the result of the negotiation with other agents, then this result clearly contradicts what the initiator agent wanted in the first place. Hence, strategies that take the initiator into account, as well as the negotiator, are needed. We created two new strategies with this purpose in mind. According to the strategy that they follow, the agents share their rejection reasons. In the first strategy, an agent provides one rejection reason per iteration, whereas the second strategy allows an agent to send multiple rejection reasons per iteration. If users cannot reach an agreement, then the postrequest is not shared by the initiator agent. However, the initiator agent can always choose to share the post by disregarding the result of the negotiation.

4.1 Good-Enough-Privacy

An agent that uses a good-enough-privacy (GEP) strategy rejects a postrequest by providing a single rejection reason per iteration, which explains why the most important rule was violated. The agent evaluates the postrequest in its ontology and finds a set of privacy rules that violate the privacy of its user. For each such rule, it computes a utility value (Equation (3)). The rule with the highest utility value is the most important rule, as the violation of this rule decreases the agent's utility most. If there is more than one important rule (i.e., rules with the highest utility value), then the agent can choose one rule arbitrarily. However, the users can choose one important rule from the set of important rules.

Recall Example 1 where Bob wants to share a picture, which is also about Alice. Assume that agents use GEP to provide rejection reasons. When Alice evaluates this postrequest, two of her privacy rules are fired. P_{A_1} is fired because (i) the context of the postrequest is inferred as `Leisure` context (i.e., the Eat & Drink concept is a subconcept of Leisure in the ontology) and (ii) `:irene` and `:david` (colleagues of Alice) are in the audience of this postrequest. P_{A_2} is fired because `:errol` is in the audience of the postrequest. Alice computes her utility with the help of Equations (2) and (3). To compute the utility, the agent needs to calculate the individual importance of the fired rules. A rule's importance is the production of its weight with the number of people violating the rule. P_{A_1} has a weight of 6 and affects David and Irene, as they are colleagues of Alice. Hence, the importance of P_{A_1} is 12. Similarly, P_{A_2} has a weight of 3 and is only violated by Errol, so the importance of P_{A_2} is 3. If the agent applies these values to Equation (2), the utility of the postrequest is $1 - ((12 + 3)/(10 * 3)) = 0.5$. This is below the utility threshold of Alice, which is 0.8, so she rejects this postrequest. In this example, P_{A_1} is the most important rule for Alice as calculated earlier. Hence, Alice wants Bob to remove `:david` and `:irene` from the audience of the postrequest. Bob revises the postrequest by removing these people from the audience. Bob calculates the utility of the revised postrequest by using Equation (1). Originally, there were 10 people in the audience, and the revised postrequest has 8 people remaining. Hence, the utility of this revised postrequest is $1 - ((10 - 8)/10) = 0.8$, which is higher than the utility threshold of Bob (0.7). Therefore, the revised postrequest is still acceptable for Bob. For Alice, the revised postrequest has the utility of $1 - (3/30) = 0.9$, which is acceptable as well. Therefore, Bob shares the revised postrequest. The utility thresholds of the users can be decided by them. For the purposes of this example, we chose 0.8 for Alice and 0.7 for Bob. We wanted both Alice and Bob to have reasonable utility thresholds that are not too high (i.e., higher than 0.9) or too low (i.e., lower than 0.5) so that we can show the steps of the negotiation better. However, a user can choose to have a utility threshold of 0.1 or 0.9 with no problems.

This strategy usually favors the initiator more since the negotiator sends one rejection reason at each iteration. The utility of the revised postrequests is always above the threshold for the initiator agent. When the utility threshold of the negotiator is met, it accepts the postrequest and sends an affirmative response to the initiator, which shares the post in its turn. This strategy finds an acceptable postrequest that usually has lower utility for the negotiator agent, as the negotiator agent could still have other privacy rules (e.g., less important ones) violating its user's privacy.

4.2 Maximal-Privacy

In the maximal-privacy (MP) strategy, our intuition is that the initiator agent may be willing to revise the postrequest by considering multiple rejection reasons at a time. Therefore, the negotiation could terminate in fewer iterations. The initiator agent revises the postrequest according to multiple rejection reasons. If the utility of the revised postrequest is not satisfactory for the initiator agent, then the negotiator agent will start narrowing the set of rejection reasons by removing rejection reasons that are less important than others. The tie-break strategy is the same as in GEP. For instance, when some rules have the same lowest utility, then the agent chooses one randomly. If the utility of a postrequest is satisfactory for both agents, then the negotiation terminates and the initiator agent shares the post.

In Example 1, when agents use MP, Alice sends all rejection reasons as a result of her postrequest evaluation. Recall that P_{A_1} and P_{A_2} are the two rules fired in Alice's ontology. Hence, Alice wants Bob to remove `:david`, `:irene,` and `:errol` from the audience of the postrequest. Bob revises the postrequest in the way Alice wants by removing these people from the audience and calculates the utility of the revised postrequest by using Equation (1). Originally, there were 10 people in the audience, and the revised postrequest has 7 people remaining. Hence, the utility of this revised postrequest is $1 - ((10 - 7)/10) = 0.7$, which is equal to the utility threshold of Bob (0.7). Therefore, the revised postrequest is still acceptable for Bob. For Alice, the revised postrequest has the utility of $1 - (0/30) = 1$, which is acceptable as well. Bob shares the revised postrequest.

In contrast to GEP, this strategy usually favors the negotiator more, as the negotiator sends all rejection reasons at first and removes them one by one if the initiator rejects. This strategy finds an acceptable postrequest that usually has lower utility for the initiator agent, as not all users in the initial audience will be able to see the content if the post is shared.

5 RECIPROCAL PRIVACY

In the previous strategies, the outcome of the negotiation was determined only by considering the current situation and ignoring the previous interactions. The outcome is beneficial for all of the negotiating agents; however, it is usually better for one party than for others. The difference may become disadvantageous for the other agent if one of them is favored most of the time. Hence, if one agent self-sacrifices privacy for a given post (i.e., it wants few people to be removed from the audience), then with another post, the other agent should be sacrificing privacy. Ideally, the sacrifices are done minimally at each negotiation, and when multiple negotiations are considered, the difference in the extent of sacrifice is little.

To realize this, the environment should hold agents accountable for their actions and promote agreement to take place. To facilitate this, we propose a trade-off mechanism based on reciprocity, which we call *reciprocal privacy* (RP). Reciprocity is a universal, powerful social norm that requires one to return kindness with kindness. According to the norm of reciprocity, there must be some "mutuality of gratification" for a social system to be stable. In other words, collective exchanges of gratifications strengthen a social system, hence reciprocity [6]. Therefore, one party feels obligated to return the act of kindness when she receives one, even from strangers. For example, when a person sends a postcard to a total stranger, this person is likely to receive one in return [15]. The norm of reciprocity is inherent in humans, showing influence in young children as young as 4 years old [20]. The mapping of reciprocity to privacy is that if an agent helps preserve the privacy of another agent, it is likely that the other agent will help preserve the initial agent's privacy in another setting. Additionally, we expect the sacrifices to be small so that the agents can tolerate them.

In RP, if one party is favored more in previous negotiations, then the mechanism tries to favor the other party in the coming negotiation. To keep track of the previous negotiations, we use a point-based system where both parties have the same amount of points in the initial state (e.g., each 5 points). For every negotiation, agents make point offers depending on who the initiator is and how much compromise the negotiator could make in that negotiation. The point offer corresponds to how many points an agent is willing to give to or request from the other agent if the postrequest is accepted. The agents consider point offers of each other while computing their utilities. The initiator agent decreases the utility for the postrequest according to the point offer of the negotiator, whereas the negotiator agent increases its utility according to the point offer of the initiator. At the end of a negotiation, the points are always transferred from the initiator agent to the negotiator agent. Further, the points are defined between every two agents, and points that one agent has against an agent cannot be used when negotiating with another agent. RP is a mechanism that works as a layer on the previous two strategies. We implement the point system on top of the negotiation principles of GEP and MP. The mechanism can also be used in conjunction with different negotiation strategies.

The execution steps in RP are as follows. Before sharing a post, the initiator agent sends the postrequest to the agents relevant to the postrequest as before. The negotiator agent evaluates the postrequest by computing its utility. If it decides to reject it, then it prepares a user list, which is ordered in a descending manner with respect to the damage the audience members make to the privacy of the negotiator agent. In the case of two people making the same damage, it can decide on the ordering randomly. Or it can choose to order all users according to some preferred criteria. For example, if the negotiator agent prepares this list as $\langle George, Filipo, Jill\rangle$, then this means that George seeing the post harms the negotiator agent more than Filipo or Jill seeing it. The negotiator agent sends this list to the initiator agent, which will consider this ordering to update postrequests by using RP over GEP (RGEP) or RP over MP (RMP), which will have the following results:

- In RGEP, the initiator agent will remove George from the audience first, if necessary. It will continue removing others from the audience if the revised postrequest is acceptable for the initiator agent but not for the negotiator.
- In RMP, the initiator agent will remove all three users from the audience. If the revised postrequest is not acceptable for the initiator agent, it will start adding people to the audience starting from Jill and so on.

5.1 Utilities in RP

As we introduce a point-based system to enable RP in negotiation, we update the utility functions in Equations (1) and (2). Each agent chooses a value between 0 and 1 to specify how important is to receive a point offer from other agents. w_P^{in} and w_P^{ng} represent these values for the initiator agent and the negotiator agent, respectively. They are taken as 0.5 as a default. P_0 is the amount of points received by both agents in the initial state, which is fixed at 5 in this strategy. We enable users to offer and take points for compensating privacy, so we need new utility functions to take these point transfers into consideration. A user who gets points should be able to compromise her privacy. To reflect this, the agent increases its utility with respect to the points received and the importance of points for a user.

5.1.1 The Initiator Agent. The initiator agent computes its point-based utility $(u_{p_i}^{in})'$ according to Equation (4). Note that we again refer to Equation (1) for the computation of $u_{p_i}^{in}$. Here, the initiator agent also considers points requested by the negotiator agent (P^{ng}). The calculation of

P^{ng} is shown in Equation (7). The initial utility will decrease at the expense of given points.

$$(u_{p_i}^{in})' = u_{p_i}^{in} - \left(\frac{w_P^{in} \times P^{ng}}{P_0}\right) \tag{4}$$

If the utility of the initiator agent is below its threshold, then the initiator agent will revise the postrequest according to the list of people sent by the negotiator agent in the first step. For this, it will make revisions to the postrequest according to the chosen strategy. If it cannot find a suitable postrequest, then it creates the revised postrequest p' and calculates the points (P^{in}) it needs to give to the negotiator agent for this request. P^{in} is the amount of points that the initiator agent can give to the negotiator agent if it accepts the updated postrequest p'. The utility $(u_{p_i}^{in})'$ needs to be at least equal to t^{in} so that the initiator agent accepts the postrequest. Therefore, the goal is to find P^{in} that satisfies $(u_{p_i}^{in})' = t^{in}$. Hence, P^{in} is calculated as shown in Equation (5). Note that $u_{p_i'}^{in}$ is the utility for the revised postrequest.

$$P^{in} = \frac{|u_{p_i'}^{in} - t^{in}| \times P_0}{w_P^{in}} \tag{5}$$

In case the initiator agent does not have sufficient points to offer, then it will revise the postrequest until it can find a suitable one for its needs. If there is no such post, then it will terminate the negotiation. If it can find such a postrequest, then it will send it together with a point offer to the negotiator agent. If none of the previous cases is possible, it will terminate the negotiation and will not share the post.

5.1.2 The Negotiator Agent. The negotiator agent computes its utility $(u_{p_i}^{ng})'$ according to Equation (6). Note that we again refer to Equation (2) for the computation of $u_{p_i}^{ng}$. The negotiator agent considers points offered by the initiator agent (P^{in}).

$$(u_{p_i}^{ng})' = u_{p_i}^{ng} + \left(\frac{w_P^{ng} \times P^{in}}{P_0}\right) \tag{6}$$

At evaluation time, the negotiator agent accepts or rejects a postrequest by also considering its current point. If the computed utility is not lower than the utility threshold, then the negotiator agent accepts the postrequest and receives the points offered by the initiator agent. If the negotiator agent rejects a postrequest, then it asks the initiator agent to give extra points (P^{ng}). In other words, the negotiator agent will accept the postrequest if the initiator agent is willing to give the specified amount of points. The negotiator agent wants its utility $(u_{p_i}^{ng})'$ to be at least equal to its threshold t^{ng} so that it can accept the postrequest. Hence, P^{ng} is calculated as shown in Equation (7).

$$P^{ng} = \frac{|t^{ng} - u_{p_i}^{ng}| \times P_0}{w_P^{ng}} \tag{7}$$

5.2 Revision Algorithm

We propose a revision algorithm that is used by the initiator agent that follows RP. The point offers are sent together with the postrequests. A `HistoryRequest` is a wrapper object that combines these. We explain the auxiliary functions used in the Revise algorithm in the following:

- initHR() creates a `HistoryRequest` with an empty postrequest and a point offer of 0.
- initPR() creates an empty `PostRequest`.
- initList() instantiates a new list.

- calculateUtility(*newP*, *firstP*) takes a postrequest and the initial postrequest, and calculates the utility of the new postrequest according to Equation (1).
- findANewPost(*iterList*) takes the previous interactions between the agents and finds a new postrequest that was not proposed by the initiator before.
- findANewPost(*iterList*, *newP*, *RList*) is used to find a new postrequest when the one returned by the previous function does not match the initiator's utility threshold or available points. This function takes previous interactions, the new postrequest, and *RList*, which is the list of previously recommended postrequests that were rejected by the initiator agent. It returns a new postrequest.
- calculatePointOffer(*utility*) takes the utility of a postrequest and calculates the amount of points that can be offered to the negotiator agent with the postrequest using Equation (5).

ALGORITHM 1: Revise (*P*,*iterList*)

```
   Input: $P$, the previous postrequest
   Input: $iterList$, the list of previous negotiation iterations
   Output: $hr$, a history request with a postrequest and a point offer
1  $hr \leftarrow$ initHR();
2  $newP \leftarrow$ initPR();
3  $R \leftarrow iterList.Last.R$;
4  if $iterList.size() = 1$ then
5      $hr.P \leftarrow P$, $hr.PointOffer \leftarrow 0$;
6  else
7      $utility \leftarrow$ calculateUtility($P$, $iterList.First.P$);
8      if $utility \geq utilityThreshold$ AND $points \geq R.PointOffer$ then
9          $hr.P \leftarrow P$, $hr.PointOffer \leftarrow R.PointOffer$;
10     else
11         $newP \leftarrow$ findANewPost($iterList$);
12         $utility \leftarrow$ calculateUtility($newP$, $iterList.First.P$);
13         $myPointOffer \leftarrow$ calculatePointOffer($utility$);
14         $RList \leftarrow$ initList();
15         $RList \leftarrow RList \cup \{newP\}$;
16         while $utility < utilityThreshold$ OR $myPointOffer \geq points$ do
17             $newP \leftarrow$ findANewPost($iterList$, $newP$, $RList$);
18             $utility \leftarrow$ calculateUtility($newP$, $iterList.First.P$);
19             $myPointOffer \leftarrow$ calculatePointOffer($utility$);
20             $RList \leftarrow RList \cup \{newP\}$;
21         $hr.P \leftarrow newP$, $hr.PointOffer \leftarrow myPointOffer$;
22 return $hr$;
```

This algorithm takes a postrequest *P* and a list of previous interactions *iterList* as inputs, and returns a history request *hr* that consists of the proposed postrequest and a point offer. First, the agent initializes *hr* (line 1). It assigns the last response of the negotiator agent to *R* (line 3). If this is the first interaction, the initiator agent adds *P* and a point offer of 0 to *hr* (line 5). Otherwise, the agent computes a utility for *P* regarding the initial postrequest (line 7). If the utility is not below *utilityThreshold*, which is the utility threshold of the initiator agent, and the initiator's *points* are enough, then *P* and *R.PointOffer* are accepted by the initiator (line 9). Otherwise, the initiator tries to find a new postrequest (line 11). The initiator calculates the utility of this postrequest *newP* (line 12) and the points that can be offered to the negotiator (line 13). The initiator also creates a recommendation list, *RList*, and adds *newP* to it (lines 14 and 15). This is useful in cases where a postrequest is not acceptable by the initiator, and a new postrequest should be created. It prevents agents from creating the same postrequests over and over. An agent tries to find a postrequest that is not below the utility threshold and affordable in terms of points. If the calculated point offer is bigger than its actual points, it means that the initiator does not have enough points to offer. Hence, the initiator agent updates *newP* until it finds an acceptable postrequest that is added to *RList* and a point offer. In the worst case, the while loop will terminate after considering all possible postrequests without finding an acceptable one (lines 17 through 20). *hr* is updated accordingly (line 21). The algorithm returns *hr*. Note that if the initiator cannot find a suitable postrequest, *hr* remains empty.

5.3 Negotiation Steps in RP

Recall Example 1, where Bob wants to share a picture of Alice. In the following, we show the negotiation steps when both agents use RGEP:

(1) Bob creates the postrequest, which is a PriNego entity that consists of the picture, audience, and context information. Bob sends this postrequest to Alice since she is tagged in the picture.
(2) Alice takes this postrequest and checks whether it conforms to her privacy concerns. David, Irene, and Errol are the ones she wants to remove from the audience. She puts these people in order of importance and sends it to Bob.
(3) Bob keeps the list of rejected people by Alice for revising the postrequest if necessary. He sends the same postrequest but with a point offer of 0.
(4) Alice receives the postrequest and evaluates it by computing her utility according to Equation (6), which is the sum of the regular utility of the postrequest $u^{ng}_{p_i}$ and a point utility. $u^{ng}_{p_i}$ is computed as 0.5 as shown in Section 4.1, and the point utility is $(w^{ng}_P * P^{in})/P_0 = (0.5 * 0)/5 = 0$. Since the sum is lower than the utility threshold, she does not accept it. She then calculates the points that she will request from Bob using Equation (7). This yields 3 points $(((0.8 - 0.5) * 5)/0.5 = 3)$, which is the points required for Alice to have utility above or equal to her threshold with the current postrequest.
(5) Bob receives the offer of Alice and calculates whether the new utility is above the threshold if he gives 3 points to Alice. Bob sees that Alice's offer is acceptable, so he sends the postrequest to Alice with the point offer given by Alice for confirmation.
(6) Alice receives the postrequest and point offer. She sees that the postrequest has not changed, and the point offer is what she has offered in the previous iteration. Alice agrees on sharing the content.
(7) Bob shares the postrequest, as they reach an agreement. Bob gives 3 points to Alice as promised.

When agents were using GEP, Bob had accepted to remove David and Irene from the audience since this modification was important to Alice. When agents were using MP, Bob had accepted to remove David, Irene, and Errol from the audience since Alice would be happy by the removal of these people from the audience. In RP, previous interactions affect the actual point of the agents. Recall that Bob has 2 points, whereas Alice has 8 points after the first negotiation. Consider the following case where Bob wants to share a new post of Alice with RGEP. The first four steps are identical to the preceding steps, and hence we omit them:

(5) Bob gets the offer of Alice and sees that he does not have 3 points. He removes David from the audience, then calculates that he can give 2 points to Alice by using Equation (5). Bob has the utility threshold of 0.7, and removing David will reduce the utility of the postrequest to 0.9. Hence, Bob can give $((0.9 - 0.7) * 5)/0.5 = 2$ points without going below his utility threshold. Bob sends this offer to Alice alongside with the new postrequest.
(6) Alice gets the postrequest and the point offer. She evaluates the postrequest by computing her utility considering the point offer. Alice agrees on sharing the content since the utility is above her utility threshold.
(7) Bob shares the postrequest, as they reach an agreement. Bob gives 2 points to Alice as promised.

If agents use RMP, then the first negotiation is the same as the RGEP since Bob's points are enough to compensate for the privacy violation. In the following, we show the second negotiation's steps when agents use RMP. Since first four steps are also identical to the previous negotiations, we omit them. Thus, we have the following:

(5) Bob gets the point offer request of Alice and sees that he does not have 3 points. He removes David, Irene, and Errol from the audience. The calculated utility is $1 - ((10 - 7)/10) = 0.7$, which is the utility threshold of Bob. By Equation (5), $((0.7 - 0.7) * 5)/0.5 = 0$, Bob calculates that he cannot give any points without going below his utility threshold. Instead, he revises the postrequest as requested by Alice.
(6) Alice gets the postrequest and the point offer. She evaluates the postrequest by computing her utility considering the point offer. Alice agrees on sharing the content since the utility is above her utility threshold.
(7) Bob shares the postrequest, as they reach an agreement. Bob does not give any points to Alice.

6 EVALUATION

We have implemented a Java-based Web application,[2] where agents communicate with each other through RESTful Web services. These Web services enable agents to start a negotiation, evaluate the incoming requests, and prepare postrequests together with point offers, if necessary. We use the OWL API [7] to work with ontologies. An agent evaluates a postrequest by adding it to its ontology and uses the Pellet reasoner [23] to infer new information from the existing knowledge.

It is important to measure if the proposed mechanism actually preserves users' privacy. For this, we evaluate the performance of our approach using multiagent simulations. Multiagent simulations enable us to mimic the user's behavior and carry out multiple negotiations. The agents evaluate postrequests according to their privacy rules. In multiagent simulations, each agent is associated with two or three privacy rules similar to the rules shown in Table 1. In other words, each agent works with the rules that are collected from real users during our user study (Section 2). The privacy rules assigned to each agent and their weights are chosen randomly.

6.1 Simulation Environment

The simulation environment is based on a real-life social network from the literature. We use a subset of the *ego-Facebook* network, which consists of 50 users and 563 relations [17]. Each agent has at least one connection in the network. The agents are connected to each other via three different relations: family, work, and friend. To observe the effects of the other variables better, we have fixed the utility thresholds (u_{p_i}) and the importance of points (w_P) to 0.7 and 0.5, respectively.

[2]Available online at https://github.com/mas-boun/prinego.

In RP, agents make use of points in evaluating postrequests. The points are defined pairwise, and hence we have created 25 agent pairs to examine their interactions. In each pair, agents are friends of each other, and every agent shares posts about the other agent. For each pair, two postrequests are generated so that each agent in the pair is an initiator in one postrequest and a negotiator in the other. These postrequests are created according to the pair's privacy rules to ensure that each postrequest conforms to the privacy rules of the initiator and violates those of the negotiator. Hence, we force agents to start a negotiation and observe the results of their interactions. Every privacy rule of the negotiator is violated to compare the negotiation strategies better. For example, the difference between GEP and MP cannot be shown if only one rule is violated by the postrequest. This is because in GEP, an agent sends rules one by one based on their importance to the initiator agent, whereas in MP, an agent sends them all in the first iteration. In other words, when there is only one violated rule, there is no difference between GEP and MP.

We have conducted the simulations in a way that both agents in a pair share posts one after another. To capture the features of RP, we need to have continuous posts. Therefore, we run these simulations repetitively and report the average results. We conduct these simulations for every negotiation strategy. For each RP-based strategy, every pair shares 10 posts and in total 250 posts are negotiated by the agents. For other strategies, the repetitions are not necessary, as previous interactions do not affect the ongoing negotiation.

6.2 Evaluation Metric

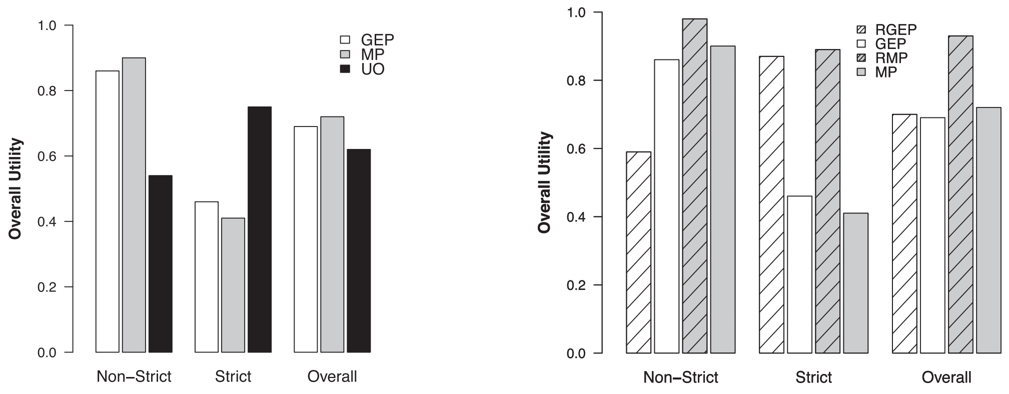
In negotiation, the aim is to find a common ground where one party does not gain a significant advantage than the other parties in the long run. We want to capture the idea of social responsibility by showing that each agent sacrifices a bit for the benefit of the other. Therefore, we would like to have cases where (i) utilities are high and (ii) utilities are close to each other so that no agent is way better off than the other one. Such and Rovatsos [28] use the product of the utilities as an evaluation metric, which is useful for evaluating whether the utilities are high. However, this metric cannot deal with utilities that are not close to each other. Compare these two cases among two agents. In the first case, Agent 1 has utility 0.7 and Agent 2 has utility 0.8. In the second case, Agent 1 has utility 1 and Agent 2 has utility 0.56. When the product is taken, both of these cases yield a utility of 0.56. However, intuitively, the first case yields a more fair setting, as it does not leave any agent at a disadvantage. To support this intuition, we propose a new evaluation metric to measure social responsibility called *preserved privacy* (PP) that gives a penalty when utilities are widely apart. We calculate this metric as shown in Equation (8). PP rewards utilities that are close to each other by considering the difference of utilities.

$$PP(u^{in}, u^{ng}) = u^{in} \times u^{ng} \times (1 - |(u^{in} - u^{ng})|) \tag{8}$$

In this equation, u^{in} represents the utility of the initiator agent, whereas u^{ng} represents the utility of the negotiator. We use the product of utilities, $u^{in} \times u^{ng}$, as a base for our metric. It means that the PP value of u^{in} and u^{ng} can be at most $u^{in} \times u^{ng}$; this happens when u^{in} and u^{ng} are equal. As the difference between u^{in} and u^{ng} increases, PP will decrease compared to the product of utilities. The maximum value that PP can have is 1, and this happens when both u^{in} and u^{ng} are 1.

7 RESULTS

First, we proposed two strategies: GEP and MP. However, these strategies were one-shot negotiations that would disregard previous interactions at negotiation time. Second, we proposed a trade-off mechanism RP that enables users to negotiate regarding previous interactions. RP is considered as a layer upon the GEP and MP. We represent these strategies as RGEP and RMP, respectively.



(a) Utility comparison of strategies (b) Effects of reciprocity

Fig. 1. Simulation results.

Currently, most online social networks allow a user to share content without considering the opinion of others that might be affected by this content. Uploader overrides (UO) is the default strategy that the social network operators offer. In addition to our proposed strategies, we also include UO in our comparison.

We use PP as the evaluation metric in our multiagent simulations. The maximum PP value cannot be reached with our scenarios, as every postrequest creates a conflict between agents. Comparing the PP values obtained in our strategies to 1 (no conflict situation) misrepresents their performance. That is why we found the best possible negotiation for every scenario according to PP. We generated the possible postrequests by removing people causing violations from the audience one by one and calculated a PP value. The maximum of these PPs is considered as the best possible outcome, and PPs of our strategies are adjusted according to this value. For example, if the best possible PP is 0.75 and the PP of our strategy is 0.6, then we consider 0.75 as 1 and adjust 0.6 to 0.8 to see more meaningful results.

Hypotheses. We formulate and test the following hypotheses regarding how well the strategies help preserve users' privacy using multiagent simulations. UO does not consider the preferences of all users, and a user can share a post without consulting others. GEP and MP try to find a common ground between users and aim to protect the privacy of all users, and hence they should perform better than UO. Hypothesis 1 captures this.

Hypothesis 1. *The users can protect their privacy significantly better when they use GEP and MP instead of UO.*

To support this hypothesis, we compared the results of GEP, MP, and UO from the simulation runs that we described in Section 6.1. In Figure 1(a), we show the average results for GEP, MP, and UO. In the overall category, GEP and MP perform better than UO. However, we would expect to have better results according to our hypothesis. To understand these results, we decided to check different factors that would affect the negotiations. We found that the most significant reason was the *strictness* of the users. The strictness is used in the sense of unwillingness to show the post to the majority of the audience. For instance, the user wants all users she suggests to be removed from the audience of the initial postrequest. We discuss the results in three categories, each of which depends on the strictness of the negotiating users. The *nonstrict* category represents the case where agents in a pair are not strict, whereas in the *strict* category, at least one of the agents in a pair is strict. The *overall* category shows the total average results of both categories. As a result of this, we noticed that GEP and MP perform really good compared to UO when the pair is nonstrict, which proves Hypothesis 1. When agents are willing to negotiate, we find an acceptable common ground. In the strict category, UO performs noticeably better than our strategies since GEP and MP try to satisfy utility thresholds in all situations. In both GEP and MP, if there is no postrequest that is acceptable for both agents, then the post is not shared, and hence the overall utility decreases.

RGEP and RMP make use of the reciprocity principle. They keep track of who is at a disadvantage and try to compensate them in subsequent negotiations. In GEP and MP, the negotiations are done without considering previous interactions, which may result in unfairness between the users. We define this in Hypothesis 2.

Hypothesis 2. *The users can have better outcomes if they prefer (RGEP or RMP) over (GEP or MP).*

In Figure 1(b), we show RP-based strategies compared to GEP and MP. RMP performs significantly better than other strategies considering the overall results. Moreover, in the case of strict pairs, RP-based strategies perform well. In RP, agents can trade off on current utility by considering future utilities. Hence, the post could be shared even if the agents are strict. In strict pairs, the maximum number of iterations was exceeded for the negotiator agents that want the majority of people to be removed from the audience, and the negotiation was terminated. In that case, the initiator agent shares the last proposed postrequest. This may result in low utility values for the negotiator agent. This was especially the problem with RGEP compared to RMP since a postrequest is revised differently. RMP has better utility values for both agents compared to RGEP if the negotiation halts halfway since the utility of the negotiator agent is higher. However, in RGEP, when one agent does not need to receive any points to share a post and its pair uses all of its points earlier, the latter agent has no points to use in future negotiations. Hence, agents could not reach an agreement in postrequests, which are not shared. This results in lower average utilities, as not sharing anything means that the negotiator will have a utility of 1, whereas the initiator will end up with a utility of 0. This is usually not the case with the RMP, because the MP strategy usually favors the negotiator agent. This is why the surplus of utility for the initiator agent is small, which results in minimal point transfer between the agents.

In UO, the initiator agent will not consider the privacy preferences of others. Hence, if a person is tagged frequently by her friends, the probability of facing a privacy violation for that person will be higher. This is captured in Hypothesis 3.

Hypothesis 3. *UO is the worst strategy to use when a user is mostly tagged in others' posts.*

According to our experiments, UO performs worse than all other proposed strategies when users share posts and get tagged in others' posts frequently. We decided to analyze the case where a user does not share posts so much but gets tagged in others' posts frequently. To observe the results, we conducted another set of simulation runs where an agent is consistently tagged by its pair without sharing posts itself. As a result of these experiments, UO performed even worse than other strategies compared to previous simulation runs. This is because only one agent in a pair shares a post with utility 1 without considering the tagged agent. Note that both agents in a pair share a post in our previous simulation runs.

Another important observation is that in the case of nonstrict agents, the average results of GEP and MP are better, whereas in the case of strict agents, RP-based strategies perform better. We then combine the best of the RGEP and RMP with GEP and MP, respectively. HybridG is the combination of GEP and RGEP, whereas HybridM combines MP and RMP. For these two strategies, we first run GEP or MP and see if agreement is reached. If there is no agreement, then RGEP or

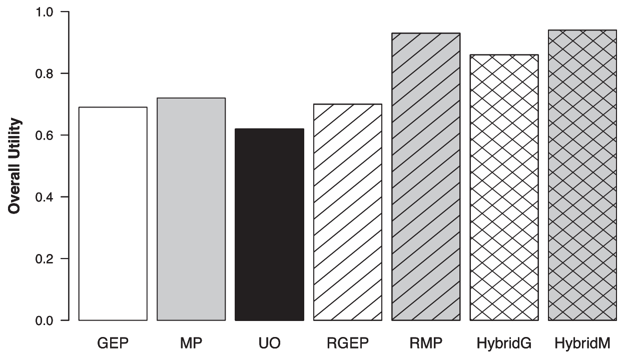


Fig. 2. Simulation results of all strategies.

RMP are respectively invoked to find common ground. If there is an agreement when we run GEP or MP, we compensate for the utility surplus of the initiator agent by transferring points to the negotiator agent if there is enough points. The amount of points transferred is calculated by Equation (5). If the points of the initiator are not enough, then it tries to find a new postrequest that is acceptable. If there are no other acceptable posts, then the post is shared as it is offered by the GEP or MP strategy and no point transfer takes place. In Figure 1, we already discussed the overall average results for five strategies (GEP, MP, UO, RGEP, RMP). Now, we compare these strategies with HybridG and HybridM in Figure 2. We observe that HybridG and HybridM perform better than RGEP and RMP, respectively. As discussed earlier, RGEP performs poorly compared to RMP when agents are strict. This also affects the performance of HybridG.

The results of the multiagent simulations show that our proposed negotiation strategies help users protect their privacy better. All of the proposed strategies perform better than UO, which is the default strategy being used in online social networks. Using RP-based strategies improved the performance of the privacy protection in the long run. We also see that through the use of social responsibility, we can go beyond negotiation.

Implementing this system as part of an online social network would require necessary networking systems to be in place. For example, it would be necessary to ensure that the agents can keep track of their points at any given time and that there are no discrepancies between their expected scores. This can be achieved in various ways, the simplest being that the scores are recorded with a trusted third party, such as the online social network administrator.

8 DISCUSSION

We have proposed a hybrid negotiation architecture that benefits from semantic knowledge and privacy rules, as well as utility functions for decision making. Based on human studies, we know that privacy is best preserved if agents collaborate in long-term relations. Hence, our proposed reciprocal strategies consider the fact that respecting another user's privacy will lead to preserving one's own privacy later. Our results show that RP is indeed successful in enabling agents to preserve their privacy over interactions.

Privacy negotiation is fairly new concept in the context of social networks. Such and Rovatsos [28] propose a negotiation mechanism that enable users to solve conflicts by agreeing on a compromise. Action vectors of 0's (deny) and 1's (accept) are created according to the privacy policies of the agents. Any mismatch in the action vectors denotes a conflict. After conflict detection, they use one-step negotiation, where each agent proposes a solution that will maximize the product of the utility values for both agents. In the end, the solution with the higher utility product is chosen. In their approach, utility values and calculations for each agent are known to all agents so that they can propose a solution that will maximize the product of the utilities. However, this is not always possible. Hence, our reciprocal strategy accommodates the fact that the utilities for each agent might be different and are hidden from each other.

Kökciyan et al. [12] propose a framework (PriArg) where the agents use argumentation to protect their users' privacy in online social networks. The arguments are derived from the user's ontology or other appropriate agents. In this work, the agents do not share the privacy constraints of their users as is done in PriArg. Hence, the privacy policies themselves are preserved. Different from PriArg, we keep track of interactions between agents and use a point system to mimic social responsibility. Overall, our proposed approach enables agents to reach a middle ground in terms of how and if a post will be shared.

Such and Criado [27] propose a computational mechanism to resolve privacy conflicts in social media. The conflict resolution considers the concessions that the users would be willing to make to achieve an agreement. In our work, there is no mediator agent that resolves the privacy conflicts. Our approach proposes a distributed approach where each agent acts on behalf of its user. The privacy rules of the users are kept private from other agents.

Squicciarini et al. [25] propose a method for collective privacy management using an incentive mechanism. The owner and co-owners of a post bid for each privacy setting separately, and the setting with the maximum total utility is chosen. Then the pivotal users whose bids affect the decision most get taxed. This strategy resembles our RP, where we use points to decide on a privacy setting. In their work, the credits are universal and can be used in every negotiation regardless of whom the agent is negotiating. However, in our work, there is a point system between every user pair to mimic reciprocity among users.

Another work that uses past interactions in the current negotiation is the work proposed by Ramchurn et al. [21]. They propose a general negotiation system with rewards that provides agents to trade off their present gain with future gains. Agents can persuade their opponents to accept their offer by giving or requesting rewards. That approach only considers negotiations with two games. When an agent accepts to concede in the first game by taking a reward, she does so to fulfill her utility target in the second game by making her opponent concede. In our approach, we do not limit the number of interactions and consider every one of them while conducting the current negotiation.

There are also interesting works that focus on detecting privacy violations in online social networks. Carminati and Ferrari [4] specify and enforce collaborative access control policies in decentralized systems where every user has its access rules, resources, and collaborative security policies in their server. Hu et al. [9] propose a multiparty access control model for online social networks. Kökciyan and Yolum [13] represent online social networks as agent-based social networks and capture interactions of agents as commitments. They show that privacy violations take place as a result of commitment violations.

The work opens up interesting directions for future work. It is worthwhile to incorporate trust relations into the utility functions such that agents are more willing to cooperate with those that they trust. The current approach and the strategies have been evaluated over multiagent simulations. An interesting follow-up evaluation could be to perform a user study to test how well the strategies mimic users' expected negotiations. Another important point is to understand when users are willing to give up their privacy so that negotiation strategies that benefit from that can be built. For example, a user who usually keeps her medical information private may prefer to reveal parts of it to a set of doctors, with the hope that a remedy can be recommended [2]. This depicts a trade-off between information privacy and information gain [14]. Being able to model such a trade-off will take us closer to building agents that negotiate better and serve users' interests the most.

REFERENCES

[1] Roland Benabou and Jean Tirole. 2010. Individual and corporate social responsibility. *Economica* 77, 305, 1–19.

[2] Igor Bilogrevic, Kévin Huguenin, Berker Agir, Murtuza Jadliwala, Maria Gazaki, and Jean-Pierre Hubaux. 2016. A machine-learning based approach to privacy-aware information-sharing in mobile social networks. *Pervasive and Mobile Computing* 25, 125–142.

[3] Gul Calikli, Mark Law, Arosha K. Bandara, Alessandra Russo, Luke Dickens, Blaine A. Price, Avelie Stuart, Mark Levine, and Bashar Nuseibeh. 2016. Privacy dynamics: Learning privacy norms for social software. In *Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems.* ACM, New York, NY, 47–56.

[4] Barbara Carminati and Elena Ferrari. 2011. Collaborative access control in on-line social networks. In *Proceedings of the 2011 7th International Conference on Collaborative Computing: Networking, Applications, and Worksharing (CollaborateCom '11).* 231–240.

[5] Armin Falk and Urs Fischbacher. 2006. A theory of reciprocity. *Games and Economic Behavior* 54, 2, 293–315.

[6] Alvin W. Gouldner. 1960. The norm of reciprocity: A preliminary statement. *American Sociological Review* 25, 2, 161–178.

[7] Matthew Horridge and Sean Bechhofer. 2011. The OWL API: A Java API for OWL ontologies. *Semantic Web* 2, 1, 11–21.

[8] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. 2004. SWRL: A Semantic Web rule language combining OWL and RuleML. *World Wide Web Consortium Member Submission* 21, 79.

[9] Hongxin Hu, Gail-Joon Ahn, and Jan Jorgensen. 2013. Multiparty access control for online social networks: Model and mechanisms. *IEEE Transactions on Knowledge and Data Engineering* 25, 7, 1614–1627.

[10] Dilara Keküllüoğlu, Nadin Kökciyan, and Pınar Yolum. 2016. Strategies for privacy negotiation in online social networks. In *Proceedings of the 1st International Workshop on Artificial Intelligence for Privacy and Security (PrAISe'16).* 2:1–2:8.

[11] Berkant Kepez and Pınar Yolum. 2016. Learning privacy rules cooperatively in online social networks. In *Proceedings of the 1st International Workshop on Artificial Intelligence for Privacy and Security (PrAISe@ECAI'16).*3:1–3:4.

[12] Nadin Kökciyan, Nefise Yaglikci, and Pınar Yolum. 2017. An argumentation approach for resolving privacy disputes in online social networks. *ACM Transactions on Internet Technology* 17, 3, 27:1–27:22.

[13] Nadin Kökciyan and Pınar Yolum. 2016. PriGuard: A semantic approach to detect privacy violations in online social networks. *IEEE Transactions on Knowledge and Data Engineering* 28, 10, 2724–2737.

[14] Nadin Kökciyan and Pınar Yolum. 2017. Context-based reasoning on privacy in Internet of Things. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI '17).* 4738–4744.

[15] Phillip R. Kunz and Michael Woolcott. 1976. Season's greetings: From my status to yours. *Social Science Research* 5, 3, 269–278.

[16] Airi Lampinen, Vilma Lehtinen, Asko Lehmuskallio, and Sakari Tamminen. 2011. We're in it together: Interpersonal management of disclosure in social network services. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, New York, NY, 3217–3226.

[17] Jure Leskovec and Julian J. McAuley. 2012. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems 25,* F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Red Hook, NY, 539–547.

[18] Debora L. McGuinness and Frank van Harmelen (Eds.). 2004. OWL Web Ontology Language overview. *World Wide Web Consortium Recommendation* 10, 10.

[19] Yavuz Mester, Nadin Kökciyan, and Pınar Yolum. 2015. Negotiating privacy constraints in online social networks. In *Advances in Social Computing and Multiagent Systems*, F. Koch, C. Guttmann, and D. Busquets (Eds.). Communications in Computer and Information Science, Vol. 541. Springer International, 112–129.

[20] Kristina R. Olson and Elizabeth S. Spelke. 2008. Foundations of cooperation in young children. *Cognition* 108, 1, 222–231.

[21] Sarvapali D. Ramchurn, Carles Sierra, Lluis Godo, and Nicholas R. Jennings. 2006. Negotiating using rewards. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems.* ACM, New York, NY, 400–407.

[22] Albrecht Schmidt, Michael Beigl, and Hans W. Gellersen. 1999. There is more to context than location. *Computers and Graphics* 23, 6, 893–901.

[23] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* 5, 2, 51–53.

[24] Manya Sleeper, Rebecca Balebako, Sauvik Das, Amber Lynn McConahy, Jason Wiese, and Lorrie Faith Cranor. 2013. The post that wasn't: Exploring self-censorship on Facebook. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work (CSCW'13)*. ACM, New York, NY, 793–802.

[25] Anna Cinzia Squicciarini, Mohamed Shehab, and Federica Paci. 2009. Collective privacy management in social networks. In *Proceedings of the 18th International Conference on World Wide Web*. ACM, New York, NY, 521–530.

[26] Margaret Gould Stewart. 2014. How giant Websites design for you (and a billion others, too). Retrieved March 13, 2018, from https://www.ted.com/talks/margaret_gould_stewart_how_giant_websites_design_for_you_and_a_billion_others_too.

[27] Jose M. Such and Natalia Criado. 2016. Resolving multi-party privacy conflicts in social media. *IEEE Transactions on Knowledge and Data Engineering* 28, 7, 1851–1863.

[28] Jose M. Such and Michael Rovatsos. 2016. Privacy policy negotiation in social media. *ACM Transactions on Autonomous and Adaptive Systems* 11, 1, 4:1–4:29.

Received January 2017; revised September 2017; accepted October 2017