Strategies for Privacy Negotiation in Online Social Networks

Dilara Keküllüoğlu
Bogazici University
34342 Bebek
Istanbul, Turkey
dilara.kekulluoglu@boun.edu.tr

Nadin Kökciyan
Bogazici University
34342 Bebek
Istanbul, Turkey
nadin.kokciyan@boun.edu.tr

Pınar Yolum
Bogazici University
34342 Bebek
Istanbul, Turkey
pinar.yolum@boun.edu.tr

ABSTRACT

Preserving privacy of posts in online social networks is difficult. One reason for this is that a post can be related to the poster as well as various others that are related to the post. Hence, it is possible to share information that pertains others without their consent. This pathological situation often leads to privacy violations that are hard to revert. Recent approaches advocate 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. This paper continues in the same line by proposing to use negotiation for reaching privacy agreements among users and introduces a negotiation architecture that combines semantic privacy rules with utility functions. We study various negotiation strategies that are inspired from negotiations in e-commerce. We discuss meaningful metrics for measuring privacy negotiation outcomes. We compare various strategies on a benchmark set of scenarios, similar to the ones that appear in real life social networks.

CCS Concepts

•Security and privacy → Social network security and privacy; •Computing methodologies → Multi-agent systems;

Keywords

Privacy, Agreement, Negotiation

1. INTRODUCTION

The content that is shared over online social networks is increasing rapidly. Contrary to traditional Web systems, such as e-commerce Web sites, where information about a user is managed solely by the user herself, in online social networks other users can contribute to the content that is shared about an individual. The shared content may reveal information about the user, which the user might not wanted to be shared herself. This creates a privacy breach on the user's side. 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. 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 a content was not shared in the first place. Recent studies on social network users show that users are willing to collaborate to ensure that their friends' privacy is preserved [8, 4].

Recent work on privacy management has focused on applying agreement technologies to solve privacy problems before they take place. Two important works in this line are that of Mester *et al.* [5] and Such and Rovatsos [9]. These approaches apply negotiation techniques to resolve privacy conflicts among users. They both consider negotiation before a content is being shared. Mester *et al.* employs 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 an 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 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. While their approach provides a decision making capability for many cases, they do not provide agents to reason on why their privacy is being violated. Further, both approaches assume that negotiation is being performed on a single content and cannot account for ongoing interactions. However, it has been observed that users build reciprocal trust in online social networks and respect others as much as others respect them [4]. Hence, it is of utmost importance to consider repeated interactions to study privacy leakages.

Accordingly, this paper first develops a hybrid negotiation architecture where privacy domain and rules are represented semantically but the agents can benefit from utility functions in reaching decisions. Second, the paper develops various negotiation strategies including one that exploits reciprocity. The key idea is that each agent keeps track of whether a certain other user has been helpful before 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. Hence, helping others to preserve privacy serves as an incentive. We evaluate this strategy under various settings as well as in relation to existing strategies. Further, we propose a new metric to measure the extent of privacy

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

preservation in a system. This metric considers how much each agent had to compromise on their privacy as well as the fairness among the agents. We show that our proposed strategy helps agents preserve their users' privacy better than existing strategies, especially in cases where agents interact with each other multiple times.

The rest of this paper is organized as follows. Section 2 describes our hybrid negotiation architecture with an emphasis on semantic representation and decision making. Section 3 explains our utility functions, evaluation metric, and strategies that an agent can use to reach an agreement. Section 4 introduces our reciprocal strategy that aims to facilitate reciprocity in preserving privacy over interactions. Section 5 compares the strategies and evaluates them on a scenario set. Finally, Section 6 compares our work with related work in the literature.

2. NEGOTIATION ARCHITECTURE

Our proposed negotiation architecture is based on semantic representation of negotiation concepts and privacy rules, but enables each agent to use its own utility functions to evaluate negotiation offers.

2.1 Semantic representation

We use PriNego [5] as the basis for the semantic aspects of negotiation. PriNego proposes a negotiation framework for privacy where each agent represents a user in the social network. Each agent is aware of the privacy concerns of its user but also 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).

Ontology: A social network consists of *users* who are connected to other users via *relations* and share some *content* with a target audience. An `Agent` sends a `PostRequest` to other agents. Each `PostRequest` is intended to be seen by a specific `Audience`, where *hasAudience* relates these two concepts. An audience is a group of agents, *hasMember* describes agents that are members of an audience. An agent may reject a post request, which is described via *rejects*. A `PostRequest` may contain some `Content` such as textual information `Text`, visual information `Medium` or `Location` information. *hasText*, *hasMedium* and *hasLocation* are used to relate corresponding concepts to `PostRequest`. A person may be mentioned in a text (*mentionedPerson*), tagged in a medium (*taggedPerson*) or at a specific location with other people (*withPerson*).

In a social network, agents may be connected to other agents via various relationships. *isConnectedTo* is a property that connects an agent to another one. The sub-properties of *isConnectedTo* (*isColleagueOf*, *isFriendOf* and *isPartOfFamilyOf*) allow us to describe relations in more detail.

Many times privacy constraints rely heavily on the context of a post. However, the context of a post is difficult to judge even if the factual information such as time and location are available [6]. A picture taken in class may depict a student at a learning context and an instructor at a working context even though the time and location are the same. To capture the fact that users can have different privacy constraints based on context, we define various `Contexts` that can be associated with a post request. Each agent analyzes a post request and infers the context information according to its observations. Following the above example, a post request with a picture in a class will reveal `Learning` context for the student and `Working` context for the instructor. We use *isInContext* to associate context information to a post request.

Privacy Constraints: Each agent captures its user's privacy constraints as semantic rules represented with a Semantic Web Rule Language (SWRL) rule [2]. 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 Eat & Drink 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 post request. An agent accepts or rejects a post request according to the user's privacy concerns. An agent can provide rejection reasons by the use of *rejectedIn* and *rejectedBecauseOf* properties. For example, a user may reject a post request 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. On the other hand, if an agent would like to give reasons about a rejection, then *rejectedIn* property is used to declare whether the rejection is caused by a medium, a post text or an audience. Furthermore, *rejectedBecauseOf* is used to specify more details about the rejection. For example, an audience can be rejected in a post request because of an undesired person in the audience. Or a medium can be rejected in a post request because of the location where the medium was taken.

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. Alice's and Bob's privacy constraints are shown in Table 1. Each rule is denoted as $P^w_{X_i}$, where X_i is a rule of agent X and w is the weight of this rule. In Example 1, Alice has two privacy concerns: P_{A_1} and P_{A_2}. P_{A_1} states that Alice's agent (`:alice`) rejects any post request if a colleague of her is in the audience of this post request, which is in leisure context. P_{A_2} states that if `:errol` is in the audience of a post request, 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.

Following the above example, when Bob initiates a negotiation with Alice, Alice evaluates Bob's post request according to her rules and decides whether to accept or to make a counter offer. This is followed by a similar move from Bob. That is, the negotiation continues in a turn-taking fashion.

The evaluations done to decide whether to accept a proposal as well as to create a new counter-offer 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 agent that initiates the negotiation (i.e., initiator) will have a different utility function than an agent that negotiates for her privacy (i.e., negotiator).

2.2 Decision Making

A negotiator agent (ng) is responsible for evaluating a post

Table 1: Privacy Rules (P) of Alice and Bob as SWRL Rules

$P^6_{A_1}$:	*hasAudience*(?postRequest, ?audience), *hasAudienceMember*(?audience, ?audienceMember), `Leisure`(?context), *hasMedium*(?postRequest, ?medium), *isInContext*(?medium, ?context), *isColleagueOf*(?audienceMember, `:alice`) → *rejects*(`:alice`, ?postRequest), *rejectedIn*(?audience, ?postRequest), *rejectedBecauseOf*(?audience, ?audienceMember)
$P^3_{A_2}$:	*hasAudience*(?postRequest, ?audience), *hasAudienceMember*(?audience, `:errol`) → *rejects*(`:alice`, ?postRequest), *rejectedIn*(?audience, ?postRequest), *rejectedBecauseOf*(?audience, `:errol`)
$P^5_{A_3}$:	*hasAudience*(?postRequest, ?audience), *hasAudienceMember*(?audience, ?audienceMember), `Party`(?context), *hasMedium*(?postRequest,?medium),*isInContext*(?medium,?context),*isPartOfFamilyOf*(`:alice`,?audienceMember) → *rejects*(`:alice`, ?postRequest), *rejectedIn*(?audience, ?postRequest), *rejectedBecauseOf*(?audience, ?audienceMember)
$P^4_{B_1}$:	*hasAudience*(?postRequest, ?audience), *hasAudienceMember*(?audience, `:harry`) → *rejects*(`:bob`, ?postRequest), *rejectedIn*(?audience, ?postRequest), *rejectedBecauseOf*(?audience, `:harry`)
$P^8_{B_2}$:	*hasAudience*(?postRequest, ?audience), *hasAudienceMember*(?audience, `:george`) *hasMedium*(?postRequest, ?medium), `Work`(?context), *isInContext*(?medium, ?context) → *rejects*(`:bob`, ?postRequest), *rejectedIn*(?audience, ?postRequest), *rejectedBecauseOf*(?audience, `:george`)
$P^6_{B_3}$:	*hasAudience*(?postRequest, ?audience), *hasAudienceMember*(?audience, `:irene`) *hasMedium*(?postRequest, ?medium), `Work`(?context), *isInContext*(?medium, ?context) → *rejects*(`:bob`, ?postRequest), *rejectedIn*(?audience, ?postRequest), *rejectedBecauseOf*(?audience, `:irene`)

request (p_i) and making a decision about this post request based on its utility ($u^{ng}_{p_i}$). In case where it wants to reject it, it also provides rejection reason(s) depending on the strategy that it follows. On the other hand, an initiator agent is responsible for initializing the negotiation with other agents, collecting responses, updating a post request and make a decision about it. It can choose to share the post, continue or terminate the ongoing negotiation according to its utility function.

Each negotiator agent makes a decision about a post request regarding its threshold and utility function. So, it can accept or reject a post request. We define such an evaluation function in Definition 2.1.

Definition 2.1 (Evaluation of the Negotiator Agent)**.** Given a post request p_i, a negotiator agent ng makes a decision about p_i regarding its threshold t^{ng}.

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

During negotiation, the initiator agent collects all responses from other agents. If all agents agree on sharing the post request, then it shares the post. Otherwise, it will try to update the post request according to rejection reasons of others. For this, the initiator agent (in) computes a utility for the updated post request ($u^{in}_{p_i}$).

Definition 2.2 (Evaluation of the Initiator Agent)**.** Given a post request p_i, an initiator agent in makes a decision about p_i regarding its threshold t^{in}.

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

3. STRATEGIES FOR PRIVACY NEGOTIATION

In PriNego [5], the initiator agent has one strategy, where it accepts every rejection reason provided by agents in the system. Such a strategy puts the initiator agent at a disadvantage. For example, if the initiator agent wants to show a picture to ten people but the audience is reduced to three 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.

3.1 Utility Functions

The utility functions help the initiator and negotiator decide how much a post request supports their privacy concerns. The utility functions give a score between 0 and 1, where larger numbers are preferred.

$$u^{ng}_{p_i} = u_{max} - (\frac{\Sigma u_{r_i}}{w_{max} \times v_r}) \tag{1}$$

$$u_{r_i} = w_{r_i} \times v_{r_i} \tag{2}$$

In Equation (1), we show how a negotiator agent (ng) computes a utility upon receiving a post request (p_i). u_{max} is the maximum utility that can be computed for a post request. w_{max} is the maximum weight of a privacy rule. v_r is the total number of users that violate privacy rules initially. In this work, u_{max} and w_{max} are set to 1 and 10 respectively. u_{r_i} is the utility value of a specific rule, which is calculated as shown in Equation (2). w_{r_i} is the weight of the rule r_i, 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_i}) that violate a rule.

$$u^{in}_{p_i} = 1 - \frac{\mid a_0 - a_i \mid}{\mid a_0 \mid} \tag{3}$$

In Equation (3), we consider how many people are removed from the audience of the original post request (a_0). a_k is the audience of the post request at kth iteration. So the initiator agent's utility will decrease if users are removed from a_0 regarding rejection reasons of others. However, if the computed utility is above threshold, the initiator agent will send the updated post request to relevant agents to have their consent. If the computed utility is below the threshold, then the negotiation will terminate and the initiator agent will not share the post. Definition 2.2 specifies how the initiator agent behaves during negotiation.

3.2 Evaluation Metric

In negotiation, the aim is to find a common ground where one party is not significantly advantageous than the other parties in the long run. Therefore, we would like to have cases where: (i) utilities are high, and (ii) utilities are close to each other. Such and Rovatsos [9] 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. Consider the case where one agent's utility is 0.9 and the other agents' is 0.3. If we look at the product of the utilities, it can lead to a best case in a strategy however one agent's utility is very low compared to the other. Hence, we cannot consider such a strategy a good one since one agent is in a disadvantageous situation. We propose a new evaluation metric *Fair Product (FP)* that gives penalty when utilities are widely apart. We calculate this metric as shown in Equation (4). FP metric rewards utilities that are close to each other by considering the difference of utilities.

$$FP(u^{in}, u^{ng}) = -\log(\mid u^{in} - u^{ng} \mid) \times \frac{u^{in} \times u^{ng}}{2} \qquad (4)$$

In the following, we introduce two strategies that agents can use at negotiation time. Agents evaluate a post request according to their roles (initiator or negotiator) as defined in Definition 2.1 and Definition 2.2. 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.

3.3 Good-Enough-Privacy (GEP)

In this strategy, the initiator agent sends a post request to relevant agents. As in PriNego, at each iteration, each agent provides a rejection reason if it rejects the post request. For this, each agent evaluates a post request by computing a utility. If this utility is above the agent's utility threshold, then the agent accepts the post request as it is (Definition 2.1). Otherwise, the agent finds its most important rule by computing u_{r_i} as shown in Equation (2), it rejects the post request and provides the corresponding rejection reason.

Recall Example 1 where Bob wants to share a picture, which is also about Alice. They try to negotiate before posting the picture so that Alice's privacy is protected as well. When Alice uses GEP, the negotiation steps are as follows:

1. Bob creates the post request, which is a PriNego entity that consists of the picture, audience and context information. Bob sends this post request to Alice since she is tagged in the picture.

2. Alice evaluates this post request, which fires two of her privacy rules. P_{A_1} is fired because (i) the context of post request is inferred as `Leisure` context (i.e., Eat & Drink concept is a sub-concept of Leisure in the ontology), (ii) `:irene` and `:david` (colleagues of Alice) are in the audience of this post request. P_{A_2} is fired because `:errol` is in the audience of post request. Alice computes her utility and rejects this post request because the computed utility is below her utility threshold. According to GEP strategy, her agent finds the most important rule (i.e., the fired rule with the highest u_{r_i}), which is P_{A_1}. Hence, Alice wants Bob to remove `:david` and `:irene` from the audience of post request.

3. Bob gets the rejection reason and applies it to the original post request. He computes a utility for the updated post request, which is higher than his utility threshold. Hence, he accepts to share the updated post request.

4. Alice gets the new post request and calculates the utility again. This time the utility is above her utility threshold, and she accepts the post request as it is.

5. Bob and Alice reach an agreement. Bob shares the updated post request.

3.4 Maximal-Privacy (MP)

In this strategy, the initiator agent may be willing to revise the post request by considering multiple rejection reasons. Hence, the negotiation could terminate in fewer iterations with this strategy. For example, the initiator agent might want the negotiation to be over in two rounds, and an agent relevant to the post request might have three rules that are violated. The initiator agent may be actually ready to prevent all these violations. If the negotiator agent uses GEP strategy, then at most two rejection reasons can be considered. In MP, an agent will send all rejections reasons to the initiator agent. If the initiator agent rejects the post request, then the negotiator agent will start narrowing the set of rejection reasons by removing rejection reasons that are less important than others.

When Alice uses MP, she decides to send all rejection reasons as a result of her post request evaluation. We only show the second step, which is the only step that is different from the previous ones.

2. Alice evaluates this post request, which fires two of her privacy rules as before (P_{A_1} and P_{A_2}). Alice computes her utility and rejects this post request because the computed utility is below her utility threshold. According to MP, her agent uses all the fired rules to prepare the rejection reasons. Hence, Alice wants Bob to remove `:david`, `:irene` and `:errol` from the audience of post request.

Bob modifies the post request in the way Alice wants, and shares the updated post request. This example shows that the initiator agent may be willing to revise a post request by considering all rejection reasons of another agent. For this, the initiator agent's utility should not be lower than its utility threshold.

4. RECIPROCAL-PRIVACY (RP)

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 the negotiating agents; however one party is usually better than the others. The difference may get disadvantageous for the others if one party is favored most of the times. To prevent this, we propose a new strategy called *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 another words, collective exchanges of gratifications strengthen a social system hence reciprocity [1]. 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 have one in return [3].

In RP, agents negotiate regarding the previous behaviors and negotiations. If one party is favored more in previous negotiations, then this strategy tries to favor the other party. 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 5pts). For every negotiation, agents exchange points depending on who is the initiator and how much benefit they get from that negotiation. Points are always transferred from the initiator agent to the negotiator agent. Points are defined between every two agents and points one has against an agent cannot be used when negotiating with another agent. For this strategy, we assume there are only two people negotiating.

Prior to sharing a post, the initiator agent sends the post request to the agents relevant to the post request. The negotiator agent evaluates the post request by computing its utility. If it decides to reject it, then it prepares an ordered list of users to be removed from the audience. The negotiator agent sends this list to the initiator agent. For example, if the initiator agent receives a list as {George, Filipo, Jill}, then the initiator agent will consider this ordering and remove George from the audience first, if necessary. It will continue removing others from the audience if the updated post request is acceptable for the initiator agent.

At every negotiation iteration, the initiator agent sends the post request together with a point offer to the negotiator agent. In the previous strategies, the negotiator agent was calculating a utility per post request, and if this utility was below its utility threshold, it would send a rejection reason. In this strategy, agents also consider point offers of each other while computing their utilities. Hence, they try to compensate the utility shortage by the points that they get from others. If the computed utility is below the threshold, the negotiator agent asks the initiator agent for sufficient points to accept the post request. Otherwise, the negotiator agent accepts the post request as it is.

The negotiator agent computes its utility $(u_{p_i}^{ng})'$ according to Equation (5). Note that we again refer to Equation (1) for the computation of $u_{p_i}^{ng}$. Addition to this, the negotiator agent considers points offered by the initiator agent (P^{in}). w_P^{ng} shows how important point offers are for the negotiator agent. It is a value between 0 and 1, and 0.5 is the default value. P_0 is the amount of points received by both agents in the initial state, which is fixed at 5 in this strategy. If w_P^{ng} is high, the negotiator agent can accept low utility post requests even if P^{in} is small.

$$(u_{p_i}^{ng})' = u_{p_i}^{ng} + (P^{in} \times \frac{w_P^{ng}}{P_0}) \tag{5}$$

The negotiator agent uses a similar evaluation function as described in Definition 2.1. The only difference is that the negotiator agent accepts or rejects a post request by also considering its current point. If the computed utility is not lower than the utility threshold, then the negotiator agent accepts the post request, and gets the points offered by the initiator agent (P^{in}). If the negotiator agent rejects a post request, then it asks the initiator agent to give extra points (P^{ng}). In another words, the negotiator agent will accept the post request 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 post request. Hence, P^{ng} is calculated as shown in Equation (6).

$$P^{ng} = \frac{\mid t^{ng} - u_{p_i}^{ng} \mid}{\frac{w_P^{ng}}{P_0}} \tag{6}$$

The initiator agent computes its new utility $(u_{p_i}^{in})'$ according to Equation (7). Note that we again refer to Equation (3) for the computation of $u_{p_i}^{in}$. RP strategy changes the evaluation of the initiator agent because the initiator agent should also consider points offered by the negotiator agent (P^{ng}) regarding w_P^{in}, the importance of point offers. The initial utility $(u_{p_i}^{in})'$ will decrease at the expense of given points (P^{ng}).

$$(u_{p_i}^{in})' = u_{p_i}^{in} - (P^{ng} \times \frac{w_P^{in}}{P_0}) \tag{7}$$

The initiator agent uses a similar evaluation function as described in Definition 2.2. If the negotiator agent accepts the post request then the initiator agent shares the post. If $(u_{p_i}^{in})'$ is equal or bigger than t^{in} then the initiator agent can accept the negotiator agent's offer hence the negotiation continues. If the utility of the initiator agent is not sufficient, then the initiator agent will revise the post request according to the list of people sent by the negotiator agent in the first step. The revising is done by removing people from the audience one by one. If there are no more people to remove from the audience and the utility is still not sufficient, then the initiator agent terminates the negotiation and do not share the post. Otherwise, it creates the revised post request p', and calculates the points (P^{in}) it needs to give to the negotiator agent for this request. In case where the initiator agent does not have sufficient points to offer, then it will revise the post request until it can find a suitable one to its needs. If there is no such post then it will terminate the negotiation. If it can find such a post request, 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.

P^{in} is the amount of points that the initiator agent can give to the negotiator agent if it accepts the updated post request p'. The utility $(u_{p_i}^{in})'$ needs to be at least equal to t^{in} so that the initiator agent accepts the post request. Therefore, the goal is to find the P^{in} that satisfies $(u_{p_i}^{in})' = t^{in}$. Hence, P^{in} is calculated as shown in Equation (8).

$$P^{in} = \frac{\mid u_{p_i'}^{in} - t^{in} \mid}{\frac{w_P^{in}}{P_0}} \tag{8}$$

4.1 Running Example

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

1. Bob creates the post request, which is a PriNego entity that consists of the picture, audience and context

information. Bob sends this post request to Alice since she is tagged in the picture.

2. Alice takes this post request, and checks whether it conforms to her privacy concerns. There are three people (David, Irene and Errol) that she wants to remove from the audience. Hence, 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 post request if necessary. He sends the same post request but with a point offer of 0.

4. Alice gets the post request and evaluates the post request by computing her utility. She does not accept it and asks Bob to give 3 points for her to accept the request as it is.

5. Bob gets the offer of Alice and calculates whether the new utility is above threshold if he gives 3 points to Alice. Bob sees that Alice's offer is acceptable so he sends the post request to Alice with the point offer given by Alice for confirmation.

6. Alice gets the post request and point offer. She sees that the post request 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 post request since they reach an agreement. Bob gives 3 points to Alice as promised.

When Alice was using GEP, Bob had accepted to remove David and Irene from the audience since this modification was important to Alice. When Alice was 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. When agents use RP, the outcome of the negotiation changes as well. Bob accepts to give Alice 3 points, and he shares the original post request; i.e., he does not remove anybody from the audience. After this negotiation, Bob has 2 points whereas Alice has 8 points. So if Bob wants to share a post about Alice next time, he cannot offer more than two points. In this case, Bob needs to do what Alice wants unless he gets some points from Alice when Alice shares posts.

4.2 Evaluation of Reciprocal-Privacy

Since the negotiations in this strategy change depending on the previous interactions, the effects of RP should be captured observing continuous posting. In addition to the running example, we introduce two other examples to demonstrate and evaluate strategies. Note that the privacy rules of the users are shown in Table 1.

Example 2. *Bob wants to share a picture where Alice is tagged. This picture is in Party context hence in Leisure context. Alice does not want her family (Harry and George) to see her party pictures, her colleagues (David and Irene) to see her leisure pictures and Errol to see any picture of her. In total, Alice is against five people to see this picture.*

Example 3. *Alice wants to share a picture where Bob is tagged. This picture is in Work context. Bob does not want George and Irene to see his pictures in Work context. He also does not want Harry to see his pictures. In total, Bob is against three people to see this picture.*

Table 2: Only one person shares a post. u^A and u^B are the utilities of Alice and Bob respectively.

Example	# of Posts	u^A	u^B	$u^A \times u^B$	FP
1	1	0.5	1	0.5	0.08
1	5	0.78	0.84	0.66	0.4
2	1	0.5	1	0.5	0.08
2	5	0.68	0.84	0.57	0.23
3	1	1	0.40	0.4	0.04
3	5	0.8	0.81	0.65	0.65

To understand how posting habits of the people affect the outcome of the negotiation, we have tried different cases while evaluating.

Case 1: In this case, one person shares posts about the other person. For example, if we consider two users *u1* and *u2*, *u1* is the only one who shares posts about *u2*. To mimic this, we refer to Examples 1, 2 and 3. In the first two examples, Bob is the one who wants to share some content, and Alice is the one who wants to share in the third example. In each example, we check the results for two different settings: (i) the initiator agent shares the post once, and (ii) the initiator agent shares the post five consecutive times. We show the results in Table 2. Note that when an agent wants to share multiple posts, we report the average utility.

When an agent shares a post only once, the utilities are like an *uploader overrides* system since the agent is willing to give away its points. In another words, the agent that wants to share a post can actually share it. When an agent keeps posting, its averaged utility decreases since the agent runs out of points while the negotiator agent's averaged utility increases. Moreover, we can observe that if an agent shares posts multiple times consecutively, the utilities of the both parties approach each other.

Case 2: In this case, both users share posts regularly. We want to see what would happen if one person shares posts consecutively, and the other person starts sharing after that point. We report results in Table 3. *Starter* shows the agent that starts sharing posts first. For example, consider the tenth run where Alice is the starter. Alice shares three posts with Bob, and Bob shares seven posts with Alice after that. We also added a random case where agents share randomly a total number of ten posts, and we report the average results of five such simulations.

Usually the person who posts later (the second person) gets better utilities in the end. They receive points from the earlier negotiations when initiator is the first person so they have more points to spend when they are the initiator. The changes in utilities are more noticeable for the runs 1 and 2 where both agents share five posts each. This is because the second person gathers points from the first five posts and has opportunity to spend these points in the remaining posts. If the second person only shares two posts rather than five, they would have remaining points to spend; i.e., they would have lost the opportunity to get a higher utility. For example, in the 8th run, Bob gets a small amount of points from Alice because she only uploads once, so even though he has nine opportunities to spend his points, he does not have sufficient points to spend. Likewise in the 4th run,

Table 3: Both users share posts. u^A and u^B are the utilities of Alice and Bob respectively.

Run	Starter	# of Bob's Posts - Ex. 2	# of Alice's Posts - Ex. 3	u^A	u^B	$u^A \times u^B$	FP
1	Bob	5	5	0.79	0.73	0.58	0.35
2	Alice	5	5	0.65	0.9	0.59	0.18
3	Bob	1	9	0.75	0.87	0.65	0.3
4	Alice	1	9	0.73	0.9	0.66	0.25
5	Bob	3	7	0.76	0.79	0.6	0.46
6	Alice	3	7	0.69	0.90	0.62	0.21
7	Bob	9	1	0.78	0.74	0.58	0.4
8	Alice	9	1	0.72	0.79	0.57	0.33
9	Bob	7	3	0.81	0.68	0.55	0.24
10	Alice	7	3	0.68	0.85	0.58	0.22
11	Random	-	-	0.75	0.8	0.6	0.39

Table 4: Results for different point weights. w_P^A and w_P^B are point weights; u^A and u^B are the utilities of Alice and Bob respectively.

w_P^A	w_P^B	u^A	u^B	$u^A \times u^B$	FP
0.5	0.1	0.78	0.84	0.66	0.4
0.5	0.3	0.78	0.84	0.66	0.4
0.5	0.5	0.78	0.84	0.66	0.4
0.5	0.7	0.74	0.86	0.64	0.29
0.5	0.9	0.7	0.9	0.63	0.22
0.1	0.5	0.86	0.8	0.69	0.42
0.3	**0.5**	**0.84**	**0.8**	**0.67**	**0.47**
0.5	0.5	0.78	0.84	0.66	0.4
0.7	0.5	0.78	0.84	0.66	0.4
0.9	0.5	0.5	1	0.5	0.08

Bob gets many points from Alice in the first nine posts she uploads but he does not have many opportunities to spend them. When the opportunities to get points are equal to the opportunities to spend points then the utility is the highest.

Case 3: In this case, we want to see the effect of point weights. For this, we do ten runs by using Example 1. Remember that Bob is the one who wants to share some content hence he will be the one offering points to Alice. We show the results in Table 4. In the first five iterations, we run Example 1 five consecutive times with different point weights; i.e, we fix Alice's point weight to 0.5 and we change Bob's point weight (w_P^B) between 0.1 and 0.9. We see that the results do not change when w_P^B is less than or equal to Alice's point weight (w_P^A). This is because Bob can give points to Alice when he has sufficient points. When w_P^B is higher than w_P^A, he is reluctant to give up points so he prefers giving up some of his utility in each iteration. Despite that his overall utility is higher because he has points to spend in each interaction. When he gives up his points quickly, he loses his chance to use them in next interactions. In the second five iterations, we fix w_P^B and try different values for w_P^A. We see that Alice's utility decreases while her point weight increases since she gets less points in each interaction, and Bob has more opportunities to use them. From this, we can conclude two things: (i) If a user usually shares posts about others, it is better to keep the point weight high. (ii) If a user is usually tagged in others' posts, it is better to keep the point weight low.

Table 5: Overall utility results for different strategies.

Ex.	Utilities	GEP	MP	RP	SR	UO	VV
1	u_1	0.9	1	0.84	0.9	0.5	1
	u_2	0.8	0.7	0.8	0.8	1	0.7
	$u_1 \times u_2$	**0.72**	0.7	0.67	**0.72**	0.5	0.7
	FP	0.36	0.18	**0.47**	0.36	0.08	0.18
2	u_1	0.74	0.74	0.73	0.74	0.5	1
	u_2	0.8	0.8	0.8	0.8	1	0.5
	$u_1 \times u_2$	**0.59**	**0.59**	0.58	**0.59**	0.5	0.5
	FP	**0.36**	**0.36**	0.34	**0.36**	0.08	0.08
3	u_1	0.8	0.7	0.8	0.7	1	0.7
	u_2	0.87	1.0	0.81	1.0	0.4	1
	$u_1 \times u_2$	**0.7**	**0.7**	0.65	**0.7**	0.4	**0.7**
	FP	0.4	0.18	**0.65**	0.18	0.04	0.18

5. EVALUATION

It is important to evaluate how the above strategies perform under various settings. Here, we do a comparison of these strategies with three more that are available in the literature; mainly, the strategy proposed by Such and Rovatsos [9] (SR) (we have implemented this as described in their paper), *Uploader Overrides (UO)* strategy and *Veto Voting (VV)* strategy. In Such and Rovatsos' strategy, agents firstly collect the conflicted audience, where two agents have different decisions, to create all possible privacy configurations then take the highest one in terms of utility product. The post is uploaded in the way initiator wants without considering other agents in *Uploader Overrides* strategy. In the *Veto Voting* strategy, if anyone in the negotiation rejects a person in the audience then that person is removed. We run the system for Examples 1, 2 and 3 using each strategy. Table 5 shows our results. For each example and strategy, we first report the individual utilities for each agent. Following that, we report both the utility product and the fair product (FP) of the individual utilities. The values for the RP are the averages of five consecutive runs for each example according to the best performing point weights configuration in Table 4.

For Examples 1 and 3, RP strategy has the best outcome according to our proposed evaluation metric. This is because RP relies on previous interactions to decide on the current interaction, and this helps utilities to get closer to

each other.

RP strategy performs almost as good as the SR strategy for Example 2 but still slightly less. The reason for this is that Alice asks Bob to remove many people from the audience. Initiator agents revise a post request in a way that their utility remains above their threshold. Therefore, Bob does not remove as many people as Alice would have liked. This results in a lower utility than Alice's utility in Example 1. We can conclude that if an agent is too restrictive regarding its privacy concerns, then this behavior would result in an overall lower utility using RP strategy.

GEP strategy performs closely with the SR strategy, giving same or better results according to FP without the disclosure of utility calculations. If we take the utility product to compare, then GEP performs equally with SR strategy. The MP strategy performs in a similar manner with VV. This is the result of negotiator agent sending all of its rejection reasons in the first iteration (like in VV), and if the initiator agent is fine with revising the post request regarding these rejection reasons then the strategy behaves like a VV strategy. However, this is not the case in Example 2 because the rejection reason of the negotiator agent includes too many people. The initiator agent does not prefer removing all those people from the audience. MP works better than VV according to both metrics in this example. UP is the strategy with the worst performance for both metrics, which is the default strategy used by most social networking sites, such as Facebook.

6. DISCUSSION

We have proposed a hybrid negotiation architecture that benefits from semantic knowledge and privacy rules as well as utility functions for decision making. We developed three strategies that can be used in this negotiation framework. Based on human studies, we know that privacy is preserved best if agents collaborate in long term relations. Hence, one of our strategies (i.e., reciprocal privacy) considers the fact that respecting another user's privacy will lead to preserving one's own privacy later. Since our focus is to maintain long term interactions as well as a balance between preserved privacy among individuals, we proposed a new metric. Our results show that reciprocal privacy is indeed successful in enabling agents to preserve their privacy over several interactions.

Privacy negotiation is fairly new concept in the context of social networks. Such and Rovatsos propose a negotiation mechanism that enable users to solve conflicts by agreeing on a compromise[9]. 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 they can propose a solution that will maximize the product of the utilities. However this is not the case with the strategies that we propose. That is, we accommodate the fact that their utilities for each agent might be different and are hidden from each other.

Squicciarini *et al.* propose a method for collective privacy management using an incentive mechanism [7]. Each post has an owner and some co-owners who are included in the post. The incentive mechanism is used for encouraging social utility. They use Clarke-Tax method to find the privacy setting that maximizes the social utility. Owner and co-owners bid for each privacy setting separately, 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 reciprocal privacy 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 who the agent is negotiating with. However in reciprocal privacy, there is a point system between every user pair. This is intuitive since our strategy is reciprocity based and reciprocity is done pairwise.

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. This would reflect real life relations more closely. Another important point is to enable the negotiation framework to be updated such that privacy rule weights can be learned over time.

7. ACKNOWLEDGMENTS

This work has been supported by TUBITAK.

8. REFERENCES

[1] A. W. Gouldner. The norm of reciprocity: A preliminary statement. *American sociological review*, pages 161–178, 1960.

[2] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al. SWRL: A semantic web rule language combining OWL and RuleML. *World Wide Web Consortium Member submission*, 21:79, 2004.

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

[4] A. Lampinen, V. Lehtinen, A. Lehmuskallio, and S. Tamminen. 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*, pages 3217–3226. ACM, 2011.

[5] Y. Mester, N. Kökciyan, and P. Yolum. Negotiating privacy constraints in online social networks. In F. Koch, C. Guttmann, and D. Busquets, editors, *Advances in Social Computing and Multiagent Systems*, volume 541 of *Communications in Computer and Information Science*, pages 112–129. Springer International Publishing, 2015.

[6] A. Schmidt, M. Beigl, and H.-W. Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[7] A. C. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. In *Proceedings of the 18th international conference on World wide web*, pages 521–530. ACM, 2009.

[8] M. G. Stewart. How giant websites design for you (and a billion others, too). https://www.ted.com/talks/margaret_gould_stewart_how_giant_websites_design_for_you_and_a_billion_others_too?language=en, 2014.

[9] J. M. Such and M. Rovatsos. Privacy policy negotiation in social media. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 11(1):4:1–4:29, 2016.