NEGOTIATION STRATEGIES FOR PRIVACY IN ONLINE SOCIAL NETWORKS

by

Dilara Keküllüoğlu

B.S., Computer Engineering, Boğaziçi University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2017

ACKNOWLEDGEMENTS

Firstly I would like to thank my thesis supervisor Prof. Pınar Yolum for guiding and supporting me for these two years. It has been fun and immensely educative in aspects of academics and life. I honestly admire her work ethics and discipline. I believe that completing my degree has become much easier because of her principles.

I would like to thank Assist. Prof. Reyhan Aydoğan and Assist. Prof. Emre Uğur for accepting to be in the thesis committee.

I also would like to thank Nadin Kökciyan who was helpful and kind throughout these two years. She provided significant guidance to me about the workings of academics, she was always patient with my questions. These two years with her and Prof. Pınar Yolum was the most enjoyable years of my academic life.

I want to thank the Artificial Intelligence Laboratory members who supported me through their friendship. They were all kind and it was so much fun to be included in that community.

I also thank my friends who supported me through thick and thin, encouraged me when I felt down, listened to me when I ranted, lent me their emotional support.

Finally, I am grateful to my parents and sister who loved and supported me through my life. My parents gave the utmost importance to our education to the degree that they sacrificed from their own life quality. I would not be able to repay them ever and I feel lucky for being their child.

This work has been supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 113E543.

ABSTRACT

NEGOTIATION STRATEGIES FOR PRIVACY IN ONLINE SOCIAL NETWORKS

Online Social Networks (OSNs) are web-services that enable users to connect with other users, share content and view other users' content. Spread of online social networks brings privacy problems that are not addressed before. Privacy is defined as the right to conceal certain information from designated people. Users share personal information, photos and videos about themselves as well as their friends. OSNs give management rights of a content to the user who uploads them. This can lead to privacy violations for other users that are related to the content. These users may not want some people to see the content and the uploader may not be aware of this preference. Ideally everyone related to a content should have a say on how it is shared.

We propose a hybrid negotiation architecture that helps users to solve privacy violations. Every user is represented by an agent that knows the relations and the privacy concerns of the user. We represent these agents and their relations semantically, also enable usage of utility functions for them to reach decisions. We develop various negotiation strategies as well as a trade-off mechanism that uses reciprocity principal to have negotiations that consider past interactions. We introduce a new evaluation metric for measuring the outcome of the negotiations. We run simulations to compare our negotiation strategies. As a result, our proposed strategies perform better than the existing methods of OSNs.

ÖZET

ÇEVRİMİÇİ SOSYAL AĞLARDA MAHREMİYET KORUNUMU İÇİN MÜZAKERE YÖNTEMLERİ

Çevrimiçi sosyal ağlar kullanıcıların, diğer kullanıcılar ile bağlanmasını, içerik paylaşmasını ve diğer kullanıcıların paylaşımlarını incelemesini sağlamaktadır. Bu sosyal ağların yaygınlaşması, daha önceden üstünde düşünülmemiş mahremiyet sorunlarını yanında getirmektedir. Mahremiyet belirli kişilerden belirli bilgileri saklama hakkı olarak tanımlanmaktadır. Bu ağların kullanıcıları kendilerinin ve arkadaşlarının kişisel bilgilerini, fotoğraflarını ve videolarını paylaşabilmektedir. Çevrimiçi sosyal ağlara yüklenen bir içeriğin yönetim hakları, bunları yükleyen kullanıcıya verilmektedir. Bu, içerik ile alakası bulunan kullanıcılar için mahremiyet ihlaline imkan vermektedir. İçerikte bulunan herkesin onun hakkında söz sahibi olması hedeflenmelidir.

Bu çalışma, mahremiyet ihlallerini çözmekte kullanıcılara yardımcı olan melez bir müzakere mimarisi önermektedir. Her kullanıcı, kullanıcının sosyal bağlantılarını ve mahremiyet ölçülerini bilen bir etmen tarafından temsil edilmektedir. Bu etmenler anlambilimsel olarak temsil edilmekte ve fayda fonksiyonları ile de kararlar almaları sağlanmaktadır. Çeşitli müzakere yöntemleri ile birlikte mütekabiliyet ilkesini kullanarak, geçmişteki etkileşimleri göz önünde bulundurup müzakere yapmayı sağlayan bir değiş-tokuş sistemi de geliştirilmektedir. Yapılan müzakerelerin sonuçlarını ölçmek için yeni bir değerlendirme metodu sunulmaktadır. Müzakere yöntemlerimizi karşılaştırmak için simülasyonlar yapılmaktadır. Sonuç olarak geliştirdiğimiz yöntemler çevrimiçi sosyal ağlar tarafından kullanılmakta olan yöntemlerden daha iyi sonuçlar vermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

hr	History-request
$iterList$	List of previous negotiation iterations
$myPointOffer$	Point offer needed for the current post-request
$newP$	Revised post-request
P	Post-request
p	Previous post-request
$PointOffer$	Point offer of the revised post-request
$points$	Available points of the initiator agent
R	The negotiator agent's last response
$RList$	Previously offered or recommended post-requests
$utility$	Utility of post-request
$utilityThreshold$	Utility threshold for acceptable post-requests

LIST OF ACRONYMS/ABBREVIATIONS

API	Application Programming Interface
CoPE	Collaborative Privacy Management
GEP	Good Enough Privacy
HybridG	Good Enough Privacy Plus Reciprocal Privacy
HybridM	Maximal Privacy Plus Reciprocal Privacy
JSON	JavaScript Object Notation
MP	Maximal Privacy
OSNs	Online Social Networks
OWL	Web Ontology Language
PriArg	Privacy Argumentation Framework
PriNego	Privacy Negotiation Framework
RP	Reciprocal Privacy
RPG	Good Enough Privacy Over Reciprocal Privacy
RPM	Maximal Privacy Over Reciprocal Privacy
SP	Scaled Product
SWRL	Semantic Web Rule Language
UO	Uploader Overrides

1. INTRODUCTION

People are connected to each other by relationships. These relationships create social networks [1]. Online social networks (OSNs) allow these relations to be represented online. According to Boyd and Ellison, social network sites are web-based services that have three main functions; creating profiles, connecting with other users and observing the connections of the users in the social network [2]. The profile information and the nature of relations may change depending on the provider and the users even though the basic purpose is online interaction and communication [3]. OSNs are used for connecting with existing friends (Facebook [4]), for creating business networks and find jobs (LinkedIn [5]), or even for matchmaking (Spritzr [6]).

Popularity of online social networks increased rapidly. In 2005, 8% of the internet users around the world used OSNs where in 2013 this increased to 73% [7]. In 2010 0.97 billion people around the world were regular users of OSNs, now the number is estimated to be 2.34 billion, which is roughly equal to one third of the world population [8]. The popularity of OSNs also increases the contents shared online. As of May 2013, 4.75 billion contents are shared daily on Facebook [4].

Privacy protection is an important aspect of these OSNs. Warren and Brandeis define privacy as “the right to be let alone” [9]. Privacy is a complex concept and changes depending on time. According to Posner, the definition of privacy has changed from seclusion to selectively concealing information. People want to be able to hide certain information about them if there is a potential for other people to use their information against them [10]. Similarly in our work, we focus on the privacy violations that stem from online friends having unintended access to contents of the user.

Users share their names, genders, nationalities in their user profiles. This information can be seen by unintended people. They connect with other users, effectively exposing their social circle. In addition to these, users share opinions, pictures and videos in OSNs. These users generally represent their real identity and expose their

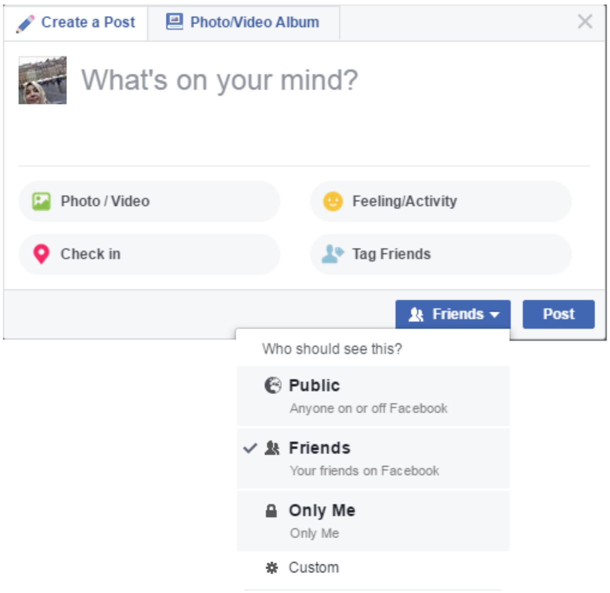


Figure 1.1. Privacy configurations for individual posts in Facebook [4].

information to malicious users which can lead to abuse [11]. Even social security numbers can be constructed from the information that are available such as hometown and date of birth [3].

Even though there are harmful things that malicious users can do, non-malicious users can cause harm using information of other users, too. A well-known privacy violation is that of Sean Lane [12]. Sean Lane bought a diamond ring for his wife as a surprise from a website that collaborated with Facebook. The purchase was shared in the Facebook feed of Lane, effectively ruining the surprise. Lane represented the users who were harmed by this feature and they won the lawsuit against Facebook. The information can be used for monitoring students [13] or citizens [14, 15]. Employers hire companies that check social media accounts of potential employees to select the best fit for them [16, 17]. It is clear that information exposure to unintended people can lead to unwanted and harmful situations.

Users of OSNs can select who can see their posts; everyone, friends, only me etc. (e.g. see Figure 1.1). There is an option for showing the post to some group of friends. While users can create special group of friends to show the post, they may have different preferences for different posts. This will force user to create groups for every post that has different privacy requirements which is very time consuming. There is also an option for creating custom configuration for posts, however this is cumbersome as friends who should see the post must be selected for every post individually. Even though the user makes all these configurations, privacy preferences of her may change and this will require configuring every post that is shared, again.

A user may be careful about how she depicts herself in the social network to avoid unwanted situations. However another user can share contents about her if they are connected in the network. While this capability creates an effective mechanism to interact, it also yields tremendous privacy violations to take place. Consider the case where a family member shares a picture of a user on a beach and colleagues learn where she was for her vacation. Another friend can share a picture in a bar and the boss may see this. The information that the posts yield may very well be private and they are shared without her consent. This creates 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. 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 [18, 19]. From a different perspective, it has been observed that some users are self-censoring their posts, assuming that they might violate their own privacy or a friend's privacy [20]. It would tremendously help if users could discuss the ramifications of a post before posting it.

Users need a method to discuss these ramifications and agree on an outcome. One method that can be used for this is negotiation. Negotiation is a process where a group of people solve conflicts on some matter and reach an acceptable outcome that every person agrees [21]. People make concessions by offers and counter-offers to reach an agreement [22]. We use negotiations in real-life when we buy products, agree on a job offer, or decide on a color of a furniture. While negotiation is an essential part of our lives, they are essential for software agents that communicate and cooperate too. Negotiation techniques can be used for resource allocation in multi-agent systems [23], product purchase in e-commerce [24], bilateral contracting in electricity markets [25], or assigning tasks during crisis management [26].

Negotiations can be divided into parts depending on various features such as number of negotiators, number of issues, and so on [22]. If the negotiation is between two people then it is called bilateral negotiation (e.g., buyer-seller), if there are multiple people negotiating then it is called multilateral negotiation (e.g.,task sharing in crisis management). If the negotiators have only one issue they should agree on (e.g., price of a product), then it is called a single-issue negotiation. Otherwise the negotiation is multi-issue (e.g., price and shape of a product).

According to Jennings *et al.*, there are three broad topics that are considered by automated negotiation researchers; negotiation protocols, negotiation objects and decision making models [21]. Interactions by the negotiating agents are conducted by the negotiation protocol. It defines which types of agents can negotiate, what are the valid actions that these agents can take, states of the negotiation and the manner of transition between them. Negotiation objects are the issues that are negotiated. Agents decide on actions to take for reaching their goals using the decision making models. For example the manner of concessions is defined here.

Even though there are considerable work on both privacy and negotiation, using negotiation in the context of privacy protection in OSNs is fairly new. Two important works in this line are that of Mester *et al.* [27] and Such and Rovatsos [28]. These approaches apply negotiation techniques to resolve privacy conflicts among users. They both consider negotiation before a content is being shared. Mester *et al.* 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 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 [19]. Hence, it is of utmost importance to consider repeated interactions to study privacy leakages.

Aside from the negotiation, argumentation which is also an agreement technology is used to solve privacy problems before they take place. The system proposed by Kökciyan *et al.* to protect the privacy of users in OSNs uses argumentation [29]. They use an agent-based social network where every agent represents a user. These agents can use arguments to convince the other party for a desired outcome.

There are also studies that approach this problem in a continuous manner. Squicciarini *et al.* propose a method for collective privacy management using an incentive mechanism [30]. Past interaction between users affect the current outcome. Another work in the same vein is the work proposed by Ramchurn*et al.* [31]. 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.

Protecting privacy in OSNs can be divided into two parts. Firstly the privacy violation must be detected. If there is no way to detect privacy violations, it is not possible to devise a way remove them. After detecting the violations, second part would be resolving the privacy violations, if possible. Kökciyan and Yolum propose a system to detect privacy violations [32]. They propose a meta-model to represent online social networks as agent-based social networks. Their system, PriGuard, which is based on their meta-model use commitments to catch privacy violations.

Accordingly, we develop an agent-based hybrid negotiation architecture where privacy domain and rules are represented semantically but the agents can benefit from utility functions in reaching decisions. Agents here represent users; they are aware of the privacy preferences and the social network of their users. The reason why we use agents is to settle differences when there is a privacy violation effortlessly and efficiently. Discussing privacy configuration of every post is time consuming for users. They may not be available at the same time to discuss or have to consider every people on the audience to solve the violation. An agent-based system will reduce the time spent on this nearly to zero except the need for the initial configuration of privacy preferences so that the agent can represent the user.

We develop strategies for privacy negotiation using our agent-based system where agents relay their privacy concerns and try to find a common ground. The decision to agree or disagree to a post getting shared is done by the utility functions, which can vary depending on the agent. A key idea in the architecture is the use of a reciprocity mechanism to equip agents 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 propose a new metric to evaluate the performance of these strategies. We use simulations with various settings to compare the strategies with each other and also with the general method used by OSNs. As a result all of our negotiation strategies perform better than the method OSNs generally use.

Chapter 2 describes our negotiation architecture; semantic representation, decision making and utility functions. Chapter 3 introduces our negotiation strategies and trade-off mechanism. It also includes evaluation of the mechanism. Chapter 4 explains the user study we conducted and present the results. Chapter 5 is the evaluation of the strategies. It describes the implementation of our system and simulation environment for the comparison. Lastly, Chapter 6 discusses related works in the field and their comparison to us.

2. NEGOTIATION ARCHITECTURE

When two or more people have different demands regarding a situation, they can try to find a solution that everyone involved agrees by using negotiation. For example a product seller may negotiate with a possible buyer about the price. We use negotiation in our system to solve privacy violations. Every user is represented by a software agent and we need to have a framework for these agents to work efficiently. Agents need to decide according to their users' preferences. They need to know when and how they are going to compromise, when are they going to accept an offer. They also need means to communicate these decisions to other agents.

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 [27] as the basis for the semantic aspects of negotiation. It 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) [33].

Ontologies are formal specifications of concepts in a domain. They define attributes of these concepts and the relations between them. These specifications enable software agents to communicate with each other in an organized manner. Humans and software agents share knowledge using ontologies [34].

Our ontology includes specifications of the social network of our users and the negotiation framework used between software agents. A social network consists of *users* who are connected to other users via *relations* and share some *content* with a target audience.

We use `Agent` to represent the users of the network. 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* (*isWorkRelatedOf*, *isFriendOf* and *isPartOfFamilyOf*) allow us to describe relations in more detail.

We use `PostRequest` to represent the contents of the social network. When an `Agent` wants to upload a post, it sends a `PostRequest` to other agents. This agent is related to the post-request by *hasOwner* property. 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. A `PostRequest` may contain some visual information, `Medium`, and *hasMedium* is used to relate it to `PostRequest`. An agent may be tagged in a medium (*includesPerson*) and this gives the agent the right to reject the post-request, which is described via *rejects*.

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 [35]. A picture taken in bar may depict a customer at a leisure context and a bartender 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, a post-request with a picture in a bar will reveal `Leisure` context for the customer and `Work` context for the bartender. We use *isInContext* to associate context information to a medium. Aside from the context information, a medium can also have a `Location`, and *includesLocation* is used to relate it to the medium.

A medium can be taken in an `Event` and they are connected by *isTakenIn* property. An event has an organizer agent and a set of agent who are not invited to the event, they are defined by *isOrganizedBy* and *didNotInvite* respectively.

All the properties we explained previously were object properties which are defined between two concepts. We expand the specifications of the ontology by data properties, which relates a concept with data values. These data values can be in formats such as strings, integers, booleans, dates and so on. An agent has a city name represented by a string, they are related by the *livesIn*. A medium has two boolean properties *hasMood* and *hasMatureContent*, indicating the mood of the medium with the former and whether the medium has mature content or not with the latter. Lastly we have *inCity* defining where the context `Vacation` is taking place.

2.2. Privacy Rules as SWRL Rules

The privacy concerns of the users should be defined formally so that agents can reason about them. Hence, it is important to choose a formal language for the representation of privacy concerns. Note that not all privacy concerns can be expressed by the chosen language.

In this work, each agent captures its user's privacy concerns as semantic rules represented with a Semantic Web Rule Language (SWRL) rule [36]. SWRL rules are encoded in the agent's ontology and contribute into the ontological reasoning. SWRL rules are of the form $Body \rightarrow Head$. Both the body and the head consist of a conjunction of ontological entities. In a rule, when the body holds then the head must also hold.

Table 2.1. Privacy rules (P) of the users 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`)

Consider the scenario in Example 2.1, which we will use as our running example.

Example 2.1. 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, $P^6_{A_1}$. Moreover, she does not want Errol to see any of her pictures, $P^3_{A_2}$.

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 depicted situation is violent. Conversely, the agent can choose not to provide a reason. In this case, head of the privacy rule only includes the *rejects* predicate. Otherwise, the agent can indicate whether the reason of rejection is the medium, or the audience with the *rejectedIn* property. Furthermore, the details about the rejection is specified in more detail by the *rejectedBecauseOf* property. For example, a medium can be rejected because of its context or mood whereas an audience can be rejected because of an unwanted person in it.

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 privacy rules are shown in Table 2.1. Each rule is denoted as $P^w_{X_i}$, which is a rule of agent X and w is the weight of this rule. In Example 2.1, 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.

We assume that the privacy rules and their weights are given to the system. Users can enter their privacy preferences personally if the system enables them to form rules and decide on a ranking between them. Another way can be using machine learning techniques to learn the privacy rules of the users and their order according to importance [37, 38].

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).

2.3. Decision Making

A negotiator agent (ng) is responsible for evaluating a post-request (p_i) and making a decision about this post-request based on its utility ($u_{p_i}^{ng}$). 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 making 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 2.3.1. It should be noted that these evaluations can change depending on the strategy that the agent uses.

Definition 2.3.1 (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}.*

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

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 2.3.2 (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}.*

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

2.3.1. 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. Every agent can have its own set of utility functions, different from defined as below.

$$u_{p_i}^{ng} = u_{max} - (\frac{\Sigma u_{r_i}}{w_{max} \times v_r}) \tag{2.1}$$

$$u_{r_i} = w_{r_i} \times v_{r_i} \tag{2.2}$$

In Equation (2.1), 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 (2.2). 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. Note that the utility of a rule can also be calculated using only the weight of it. This is also an acceptable utility function. However, we decided to emphasise the number of the users that violate a rule.

The utility of the negotiator agent, $u_{p_i}^{ng}$, can be minimum zero and maximum one. The maximum value the Σu_{r_i} can take is $w_{max} \times v_r$, which is the case where every violated rule has a weight of w_{max} and $\Sigma v_{r_i} = v_r$ (initial case, violations remain as they are). Hence the utility can be minimum zero. Violations cannot increase the utility, in the case where there are no violations utility gets its maximum value, which is one.

$$u_{p_i}^{in} = 1 - \frac{\mid a_0 - a_i \mid}{\mid a_0 \mid} \tag{2.3}$$

In Equation (2.3), 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 2.3.2 specifies how the initiator agent behaves during negotiation.

3. TRADE-OFF MECHANISM FOR PRIVACY

In PriNego [27], there is only one strategy for the agents to use. Negotiator agent evaluates the post-request and sends every violation it finds as a rejection reason to the initiator agent. The initiator agent then accepts every rejection reason provided by the negotiator and removes all violations. While 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 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.

We created two strategies with this purpose in mind. These strategies represent how initiator and negotiator agents behave, i.e., create rejection reasons, evaluate and revise the post-request, terminate negotiation. Our strategies use utilities to find a configuration of the post-request that is beneficial for both agents. In the following two sections, we introduce these two strategies that the agents can use at negotiation time. Agents evaluate a post-request according to their roles (initiator or negotiator) as defined in Definition 2.3.1 and Definition 2.3.2. 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.

3.1. Good Enough Privacy

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 2.3.1). Otherwise, the agent finds its most important rule by computing utilities of every rule, u_{r_i}, as shown in Equation (2.2), it rejects the post-request and provides the corresponding rejection reason.

Recall Example 2.1 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:

(i) 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.

(ii) 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.

(iii) 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.

(iv) Alice receives 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.

(v) Bob and Alice reach an agreement. Bob shares the updated post-request.

This strategy usually favors the initiator more since negotiator sends violated rules one by one and negotiation terminates once the utility threshold of the negotiator is met. Which means in the space of acceptable post-requests for both agents, this strategy finds the one that usually has lower utility for the negotiator agent.

3.2. Maximal Privacy

In the GEP strategy, the negotiator was sending rejection reasons one-by-one based on importance. The violations that impact the utility the least were usually not reported to the initiator since the utility of the post-request has passed the threshold. However the initiator agent may be willing to remove those violations from the post-request if asked. This may result in better outcome overall. In this strategy, we consider the possibility that 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.

(ii) 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.

In contrary to the GEP, this strategy usually favours the negotiator more since negotiator sends all violated rules at first and removes them one-by-one if initiator rejects. The negotiation terminates once the utility threshold of the initiator is met. Which means in the space of acceptable post-requests for both agents, this strategy finds the one that usually has lower utility for the initiator agent.

3.3. 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 the negotiating agents; however one party is usually better than the others. The difference may get disadvantageous for the other agent if one of them is favored most of the time. Hence, if the one agent self-sacrifices for a given post, 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 another words, collective exchanges of gratifications strengthen a social system hence reciprocity [39]. 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 [40]. 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.

When people are eating out with a friend, they usually decide on which restaurant to go together. If both of them want restaurants with different types of food, it is reasonable to go to one of them now and to the other one next time. This result is better than the alternative where they go to a restaurant where they both are not partial to when they meet. Of course if one of the friends really dislikes the restaurant her friend chooses then it is better to find a compromise. Following this example we designed our mechanism to favor the party that compromised in past. In other words, 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 5pts). For every negotiation, agents make point offers depending on who is the initiator 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 post-request is accepted. The agents consider point offers of each other while computing their utilities. The initiator agent decreases the utility for the post-request according to the point offer of the negotiator, while 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. For this mechanism, we assume there are only two agents negotiating. Further, the points are defined between every two agents and points one has against an agent cannot be used when negotiating with another agent. The reason for deciding on a pairwise points is to mimic real-life better. For example if a person compromise to fulfil a friend's wishes, that person would not use this compromise as a leverage against another friend. The interactions between a person and a friend should not affect the interactions between the person and another friend.

We use this mechanism as a layer on the previous two strategies. The structure is explained in the Figure 3.1. We implement the point system on top of the negotiation principles of GEP and MP. The mechanism can also be used in conjunction with a different negotiation strategy.

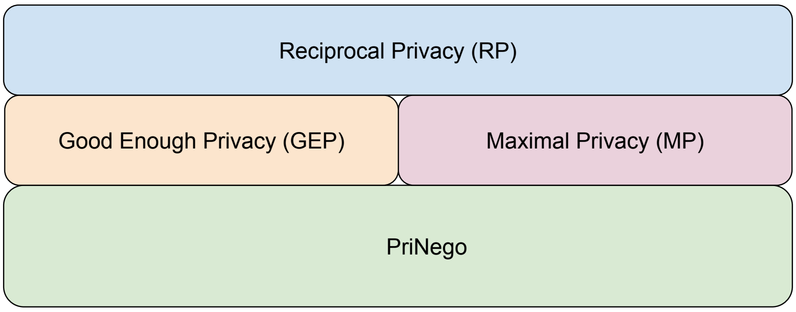


Figure 3.1. Structure of the strategies.

With this mechanism; before sharing a post, the initiator agent sends the post-request to the agents relevant to the post-request as before. 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. This list is ordered in descending manner with respect to the damage the audience members make to the privacy of the negotiator. The purpose of this move is to enable initiator agent to make informed choices when offering new post-requests in the negotiation. For example if the negotiator sends a list as $\langle George, Filipo, Jill \rangle$, that means George seeing the post harms the negotiator more than Filipo or Jill seeing it. In that case, the initiator agent will consider this ordering while creating post-requests depending on the version of the strategy.

RP over GEP : If the RP strategy is based on GEP, the initiator agent will remove George from the audience first, if necessary. It will continue removing others from the audience if the updated post-request is not acceptable for the negotiator agent and is acceptable for the initiator agent.

RP over GEP : If the RP strategy is based on MP, the initiator agent will remove all three from the audience by default and add Jill, if necessary. It will continue adding others to the audience if the updated post-request is not 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 strategies we introduced, 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. When we use RP mechanism over the strategies, 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.

Every agent has different preferences. One agent may want to compromise its privacy for more points and another may refuse to compromise it at all. We need to be able represent these differences between agents. Hence, we introduce the following concept.

Willingness to Compromise: The factor that regulates the trade-off between utility and points. It is a value between 0 and 1, and 0.5 is the default value. Every agent has a point weight, where we represent the willingness to compromise with, and it indicates how important point offers are for the agent. If a point weight is high then it means the points are important to the agent. In that case the agent may be willing to compromise from its privacy utility for small amount of points. In contrary, if the point weight is low then an agent might need more points offered to compromise from its privacy utility.

With the introduction of points, calculating utility for agents needs to change. In this strategy the negotiator agent computes its utility $(u_{p_i}^{ng})'$ according to Equation (3.1). Note that we again refer to Equation (2.1) 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} is the point weight. P_0 is the amount of points received by both agents in the initial state, which is fixed at 5 in this strategy. w_P^{ng}/P_0 gives the utility value of one point to the negotiator agent and the utility of the points offered by the initiator agent can be calculated by multiplying it with the P^{in}. If w_P^{ng} is high, the negotiator agent can accept low utility post-requests even if P^{in} is small.

$$(u_{p_i}^{ng})' = u_{p_i}^{ng} + (P^{in} \times \frac{w_P^{ng}}{P_0}) \tag{3.1}$$

The negotiator agent uses a similar evaluation function as described in Definition 2.3.1. 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 (3.2). As we stated earlier, w_P^{ng}/P_0 gives the utility gain of one point to the negotiator agent. Using this, the negotiator agent calculates the points needed to compensate the utility deficit of the offered post-request which is the $\mid t^{ng} - u_{p_i}^{ng} \mid$.

$$P^{ng} = \frac{\mid t^{ng} - u_{p_i}^{ng} \mid}{\frac{w_P^{ng}}{P_0}} \tag{3.2}$$

The initiator agent computes its new utility $(u_{p_i}^{in})'$ according to Equation (3.3). Note that we again refer to Equation (2.3) 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. w_P^{in}/P_0 gives the utility loss of one point to the initiator agent and the utility loss of the points requested by the negotiator agent can be calculated by multiplying it with the P^{ng}. The initial utility $(u_{p_i}^{in})'$ will decrease at the expense of given points.

$$(u_{p_i}^{in})' = u_{p_i}^{in} - (P^{ng} \times \frac{w_P^{in}}{P_0}) \tag{3.3}$$

The initiator agent uses a similar evaluation function as described in Definition 2.3.2. 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 or adding people to the audience one by one depending on the underlying strategy.

If there are no more changes that can be done to 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 (3.4). As we stated earlier, w_P^{in}/P_0 gives the utility loss of one point to the initiator agent. The utility surplus of the initiator agent, $\mid t^{ng} - u_{p_i}^{ng} \mid$, divided by the utility loss of one point gives the initiator agent points it can offer to the negotiator agent without decreasing its utility below the threshold.

$$P^{in} = \frac{\mid u^{in}_{p'_i} - t^{in} \mid}{\frac{w^{in}_P}{P_0}} \tag{3.4}$$

3.3.1. Revise Algorithm for Initiator

We explain a version of the revise algorithm our initiator agent uses to create new post-requests. Specifically this revise is used when the reciprocal privacy is used as a incentive mechanism. The initiator agents creates the initial post-request and sends it to the negotiator. The negotiator gets the post-request and creates rejection reasons according to the strategy it uses. Initiator gets the rejection reason from the negotiator and creates a modified post-request according to the strategy and the rejection reason. Revise algorithm is used in this step. Since this algorithm is used with the trade-of mechanism, the point offers are sent alongside with the post-requests. As we explained in Section 2, we use `HistoryRequest` as a wrapper object to unify these two. We explain the functions used in the Revise algorithm below.

- initHR() creates a `HistoryRequest` with an empty post-request and point offer of 0.
- initPR() creates an empty `PostRequest`.
- initList() instantiates a new list.
- calculateUtility($newP$, $iterList.First.P$) takes a post-request and the initial post-request, calculates the utility of the new post-request according to Equation 2.3.
- findANewPost($iterList$) takes the previous interactions between the agents and finds a new post-request that was not proposed by the initiator before.
- findANewPost($iterList$, $newP$,$RList$) is a version of the previous function. It is used to find a new post-request 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 post-request and the RList which is the list of previously recommended post-requests that were rejected by the initiator agent. It returns a new post-request.

- calculatePointOffer(*utility*) takes the utility of a post-request and calculates the number of points can be offered to the negotiator agent with the post-request using Equation 3.4.

This algorithm takes a post-request p and a list of previous interactions $iterList$ as inputs, returns a history request hr that includes the proposed post request and point offer. Firstly the agent initializes the hr (line 1). Then assigns the last response of the negotiator to R (line 3). If this is the first interaction, initiator assigns the p without changing it to the post-request of hr alongside with a point offer of zero (line 5). Otherwise, the agent looks at the utility of the p and the last point offer given by the negotiator agent. If the utility is not below the threshold and the initiator has required points, then p and $R.PointOffer$ are accepted by the initiator. If the point offer from the negotiator agent is not acceptable, then initiator agents tries to find a new post request (line 11). Initiator calculates the utility of this post request $newP$ (line 12) and the points that can be offered with it (line 13). Initiator also creates a recommendation list, $RList$, and puts this post-request to it (line 14,15). This is for the cases where this post-request is not acceptable for the initiator and another new post should be created. It prevents agents to create same post-requests they created before. If the calculated point offer is bigger than its actual points, this means initiator does not have enough points to offer with $newP$. Hence the initiator agent updates $newP$ until it finds an acceptable post-request and point offer or cannot find a unique post-request any more. If it cannot find a unique post-request, then the hr is returned empty. Otherwise, hr's post-request and point offer are assigned and returned.

3.3.2. Negotiation Steps using Reciprocal Privacy

Recall Example 2.1 where Bob wants to share a picture of Alice. In the following, we show the negotiation steps when both agents use the GEP version of the RP.

```
Require $p$, previous post request
Require $iterList$, list of previous negotiation iterations
$hr \leftarrow$ initHR()
$newP \leftarrow$ initPR()
$R \leftarrow iterList.Last.R$
if $iterList.size() = 1$ then
  $hr.P \leftarrow p$, $hr.PointOffer \leftarrow 0$
else
  $utility \leftarrow$ calculateUtility($p, iterList.First.P$)
  if $utility \geq utilityThreshold$ AND $points \geq R.PointOffer$ then
    $hr.P \leftarrow p$, $hr.PointOffer \leftarrow R.PointOffer$
  else
    $newP \leftarrow$ findANewPost($iterList$)
    $utility \leftarrow$ calculateUtility($newP, iterList.First.P$)
    $myPointOffer \leftarrow$ calculatePointOffer($utility$)
    $RList \leftarrow$ initList()
    $RList \leftarrow RList \cup \{newP\}$
    while $myPointOffer \geq points$ do
      $newP \leftarrow$ findANewPost($iterList, newP, RList$)
      $utility \leftarrow$ calculateUtility($newP, iterList.First.P$)
      $myPointOffer \leftarrow$ calculatePointOffer($utility$);
      $RList \leftarrow RList \cup \{newP\}$
    end while
    $hr.P \leftarrow newP$, $hr.PointOffer \leftarrow myPointOffer$
  end if
end if
return $hr$
```

Figure 3.2. REVISE (*p*,*iterList*) Algorithm.

(i) 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.

(ii) 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.

(iii) 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.

(iv) 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.

(v) 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.

(vi) 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.

(vii) Bob shares the post-request since they reach an agreement. Bob gives 3 points to Alice as promised.

Table 3.1 shows negotiation steps and outcomes using different strategies and mechanisms on the Example 2.1. 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

Table 3.1. Various methods applied to Example 2.1.

The Method	Alice	Bob	Outcome
GEP	Remove `:david` & `:irene`	Removes `:david` & `:irene`	`-:david`, `-:irene`
MP	Remove `:david`, `:irene` & `:errol`	Removes `:david`, `:irene` & `:errol`	`-:david`, `-:irene`, `-:errol`
RP over GEP	Give 3 pts	Gives 3 pts	A: 8 pts, B: 2 pts
RP over GEP 2nd Run	Give 3 pts	Removes `:david`, gives 2 pts	`-:david`, A :10 pts, B : 0 pts
RP over MP	Give 3 pts	Gives 3 pts	A: 8 pts, B: 2 pts
RP over MP 2nd Run	Give 3 pts	Removes `:david`, `:irene` & `:errol`	`-:david`, `-:irene`, `-:errol`, A: 8 pts, B: 2 pts

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. In the following, we show this case. First four steps are identical with the previous example hence they are not included. This is the case where Alice uses RP over GEP.

(v) 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 alongside the new request to Alice.

(vi) Alice gets the post-request and point offer. She evaluates the post-request by computing her utility considering the point offer. Alice agrees on sharing the content since the utility is above her utility threshold.

(vii) Bob shares the post-request since they reach an agreement. Bob gives 2 points to Alice as promised.

If Alice uses the RP over MP, the first negotiation is the same as the RP over GEP since Bob's points are enough to compensate for the privacy violation. In the following, we show the second negotiation's steps when Alice uses the RP over MP. Since first four steps are also identical to the previous negotiations we omit them.

(v) Bob gets the offer of Alice and sees that he does not have 3 points. He removes David, Irene and Errol from the audience, then calculates that he can give 0 points to Alice alongside the new request to Alice.

(vi) Alice gets the post-request and point offer. She evaluates the post-request by computing her utility considering the point offer. Alice agrees on sharing the content since the utility is above her utility threshold.

(vii) Bob shares the since they reach an agreement. Bob does not give points to Alice.

3.3.3. 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 2.1. Also the evaluations are done using the GEP version of the Reciprocal Privacy.

Example 3.1. 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 3.2. 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.

Ideally RP must conduct fairly independent of circumstances. One of the circumstances is the posting habits of people. To understand how this affect the outcome of the negotiation with RP, we have tried different cases while evaluating.

Case 1: There are active users and passive ones in OSNs. We want to see how the privacy is protected when one of the users keep posting about a friend. That is why in this case, one person shares posts about her friend and the friend does not post anything in return. 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 2.1, 3.1 and 3.2. 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. To evaluate the results we use two metrics. One of them is the product of utilities which is represented as $u^A \times u^B$. Other one is *Scaled Product (SP)*, which is a metric we proposed. We explain this metric in detail in the Chapter 5. However, briefly we aim to give penalty when utilities are small and/or widely apart. We show the results in Table 3.2. Note that when an agent wants to share multiple posts, we report the average utility.

When an agent shares a post only once (lines 1,3,5), 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 posts five times (lines 2,4,6), its averaged utility decreases since the agent runs out of points while the negotiator agent's averaged utility increases. Remember the initiator agent in Examples 2.1 and 3.1 is Bob, in Example 3.2 it is Alice. We can also observe that if an agent shares posts multiple times consecutively, the utilities of both parties approach each other. This shows us that an active person, who regularly uploads posts about a passive friend, does not have an advantage over her friend in the long run.

Table 3.2. Only one person shares a post. u^A and u^B are the utilities of Alice and Bob respectively.

Line	Example	# of Posts	u^A	u^B	$u^A \times u^B$	SP
1	2.1	1	0.5	1	0.5	0.25
2	2.1	5	0.78	0.84	0.66	0.62
3	3.1	1	0.5	1	0.5	0.25
4	3.1	5	0.68	0.84	0.57	0.48
5	3.2	1	1	0.4	0.4	0.16
6	3.2	5	0.8	0.81	0.65	0.64

Table 3.3. Both users share posts. u^A and u^B are the utilities of Alice and Bob respectively.

Run	Starter	# of Bob's Posts - Ex. 3.1	# of Alice's Posts - Ex. 3.2	u^A	u^B	$u^A \times u^B$	SP
1	Bob	5	5	0.79	0.73	0.58	0.54
2	Alice	5	5	0.65	0.9	0.59	0.44
3	Bob	1	9	0.75	0.87	0.65	0.57
4	Alice	1	9	0.73	0.9	0.66	0.55
5	Bob	3	7	0.76	0.79	0.6	0.56
6	Alice	3	7	0.69	0.9	0.62	0.49
7	Bob	9	1	0.78	0.74	0.58	0.56
8	Alice	9	1	0.72	0.79	0.57	0.55
9	Bob	7	3	0.81	0.68	0.55	0.48
10	Alice	7	3	0.68	0.85	0.58	0.48
11	Random	-	-	0.75	0.8	0.6	0.57

Case 2: In the Case 1, one of the users was an active user and the other was a passive one. In this case, both users are active and 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 use Examples 2.1 and 3.2 since we want both Alice and Bob to be the initiator agent in one example and the negotiator in the other. *Starter* shows the agent that starts sharing posts first. # of Bob's Posts and # of Alice's Posts indicate how many times Bob and Alice upload a post respectively. u^A and u^B is respective averaged utilities of Alice and Bob after the negotiations. $u^A \times u^B$ is the product of utilities and SP is the *Scaled Product* of u^A and u^B. We report results in Table 3.3. 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.

The person who posts later (the second person) usually gets better utilities in the end. This is because 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 one and two 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 eighth 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 fourth run, 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.

We can see that the SP of the runs are similar, this means that the amount of posts each user uploads do not affect the overall utility if both users are active.

Table 3.4. 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	$u^A \times u^B$	SP
0.5	0.1	0.78	0.84	0.66	0.62
0.5	0.3	0.78	0.84	0.66	0.62
0.5	0.5	0.78	0.84	0.66	0.62
0.5	0.7	0.74	0.86	0.64	0.60
0.5	0.9	0.7	0.9	0.63	0.5
0.1	0.5	0.86	0.8	0.69	0.65
0.3	0.5	0.84	0.8	0.67	0.64
0.5	0.5	0.78	0.84	0.66	0.62
0.7	0.5	0.78	0.84	0.66	0.62
0.9	0.5	0.5	1	0.5	0.25

Case 3: In this case, we want to see the effect of point weights on the outcome. Also find the best point weight to have when uploading a post or tagged in a post. For this, we do ten runs by using Example 2.1. Remember that Bob is the one who wants to share some content hence he will be the one offering points to Alice. We change the point weight of Alice and Bob one at a time (i.e., fix one of them to 0.5 and change other). We show the utilities of Alice and Bob (u^A and u^B), product of these utilities ($u^A \times u^B$) and *Scaled Product* (SP). We show the results in Table 3.4. In the first five iterations, we run Example 2.1 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. However in the aspect of SP, there are no great differences unless one part has a high point weight (0.9). This is because users give more importance to their points and compromise from their privacy instead if they have high point weights. This results in a lowers overall utilities and SP.

4. USER STUDY

The previous examples that are used for the evaluation of RP are based on our expectation of general public. To understand the privacy concerns of social media users and find privacy constraints based on real life, we have conducted a user study. We selected 10 people from our friends and asked them questions about real life situations about privacy in online social networks. Half of the participants were female and the other half male. Five of them were below 40 years old and other five was above 40. Participants' professional backgrounds were vary (doctor, photography, education and such) but they all use social networking sites daily. However, their frequency of sharing posts are at most once a week.

We chose the approach of Sleeper *et al.* as our guideline for this user study [20]. In their work, they aim to find the types of self-censorship online social network users make by interviewing participants. To accomplish their objective they conducted an experiment with 18 people in ages between 20 and 51 who use Facebook.They chose these people according to their English proficiency, Facebook usage, texting and self-censorship frequency. The experiment consisted of two parts. First one was the diary study where participants sent sms texts when they held back from posting something with small descriptions about the potential post. Sms was used to enable quick logging where they were asked about these potential posts in detail with nightly survey. Second part of the experiment was interviewing these people about their potential posts. They invited people who sent at least four surveys to lab and held an interview approximately for one hour to understand the content and the intended audience of the posts. According to the results, users did not want to share personal contents (e.g., personal updates and events) or offensive materials. To enact this result, we chose ten pictures that depict people in various contexts of situations to use in the study. Some of this situations are personal; couple wearing swimsuits and hugging on a beach, wedding picture, family picture. Few of them represent people in offensive or unwanted situations like a crowd protesting, a man passed out from alcohol or a picture that could be interpreted to have sexual intonations.

We carried the user study in an interview style. We sat down with the participants and asked them questions following a guideline [41]. The aim of the interview style was allowing participants to give their opinions freely without a strict structure and steer the interview depending on their responses. The interviews were not audio recorded. Each interview took about 30-40 minutes.

As stated above, we chose ten pictures and we showed five of these pictures to the participants. Participants usually answered for different set of five pictures and these five pictures were selected so that every picture had reasonable exposure to get results. In the user study, we showed the participants these pictures and requested them to answer our questions as they were the person in the picture. Our interview consisted questions regarding three main scenarios; participant shares a picture of themselves, participant is tagged in a picture, and participant shares a picture of a friend.

The flow of the first scenario was as following;

- We showed a picture and asked participants whether they will share this picture or not.
- If they chose not to share, we asked them their reasons to not disclose the material.
- If the reason for not sharing the picture is eliminated, we asked whether it would be okay to share it.

The aim of this part was to find privacy constraints of online social network users in different contexts. What kind of pictures they are not comfortable with sharing and the reasons of them provide us the privacy preferences of the users.

In the second scenario, we ask questions about the case where a friend shares a picture and it is violating privacy preferences of them.

- We asked whether they would be happy in the case where the picture is shared by a friend.

- If they were not happy, we asked whether it would be better if the friend consulted them before sharing the picture.
- If the friend is convinced to remove some of the violations of the participant, would the participant be happy if they chose which violations.

Here, we try to measure the willingness to negotiate on the privacy constraints. The last scenario was conducted without a specific picture, it was about the general behavior of the participant in various situations.

- We asked the participants if they share pictures of their friend.
- If they will remove or modify the post of their friend in the case where the friend is not happy with the situation.
- Whether they would share an unwanted picture of a friend, if said friend shared their unwanted pictures before.
- If the friend complied with the participant's wishes and will they be willing to do the same.

We measure willingness to negotiate with a friend in the case of conflict. We aim to understand the situations where a user would consider the privacy of their friends.

4.1. Observations

After conducting the user study, we collected various privacy concerns as explained in Section 4.2. In addition to these rules we observed the following;

- Most of the participants are not happy with the idea that another person can share personal content about them. However, the participants are willing to talk with the content uploader to negotiate on the content to be shared (e.g., agreeing on who can see the content).
- For most of the participants, reciprocity is an important factor. In another words, if the participant is favored in a negotiation with another user, then the partici-

pant agrees that the other user should be favored in the following negotiations.

- If the other users are not willing to negotiate, then most of the participants prefer talking to such users in person to prevent them from sharing undesired content in the future. Some of the participants would prefer sharing a content to hurt the person who is not willing to negotiate. In the worst case, we expect that the participants would choose to unfriend such users.
- Most of the participants say that they would not put up a content online if another person does not want it to be shared for some reasons. However, they would consider past interactions and the degree of relationship (e.g., a close friend) with that person to decide on sharing a content or not. Only one participant said that she could share a content if she really wants it to be shared without considering what others would say.

4.2. Privacy Concerns

In the user study, we have learnt that the participants have different privacy concerns regarding the content being shared. These concerns are mostly influenced by the meta-information that comes along with the content. For example, some participants have privacy concerns that are based on the context of content, while others prefer not sharing a content with people who are connected to these via various relationships. The privacy concerns that are driven from the user study are as follows:

(i) 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 unhappy times 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.⟩

(ii) Bar pictures: Some of the users did not want their families to see pictures 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.⟩ Some of the users did not want their friends who do not drink to see such pictures. ⟨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.⟩

(iii) 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 work context. For example, in the education context, a teacher 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 in, then the user does not prefer her friends who live in that city to see it.⟩

(iv) Event pictures: When we showed participants a wedding picture, they did not want people who were not invited to the event to see these. 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.⟩

(v) 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.⟩

(vi) Removal of specific people: Some users prefer not sharing a content with a set of users if a specific condition holds (e.g., if the user is in a party picture, Alice and Bob should not see this.). The set of users can be described as a list of people or in terms of relationships with the user. ⟨C6: If the user is in a picture and some conditions hold, then the user does not prefer a set of users to see this picture.⟩

(vii) 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.⟩

(viii) 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.⟩

4.3. Privacy Concerns as SWRL Rules

We represent a subset of the privacy concerns derived from the user study as SWRL rules with their concern ids shown in Table 4.1. While we are able to express most of these concerns, we will discuss the limitations for not being able to express the remaining ones.

We have already discussed the privacy rules of Alice (P_{A_1} and P_{A_2}) (i.e., see Figure 2.1) in Section 2.2. Note that these rules are examples for the concern $C6$. In the first one, the context of the medium and the relationship type are the conditions specified by Alice. The second one does not depend on any attribute of the content being shared. Alice does not want to share any content with `:errol`. Note that, it is possible to update the ontology in order to be able to express more privacy concerns. In the Table 4.1 we represent seven concerns extracted from the Section 4.2.

(i) If the user `:x` is included in a photo that depicts a depressed mood, then `:x` rejects the post to be shared. This represents the concern $C1$.

(ii) $C2.1$: If the photo is located in a bar then `:x` does not want her family members to see the post.

(iii) The photos on a beach should not be shared with work related people, $C3.1$.

(iv) `:x` does not want her vacation photos to be shared with people who resides in the vacation area, $C3.2$.

(v) $C4$: If `:x` did not invite someone to an event she organized, then that person should not the photos taken in the event.

(vi) The photos with mature content must not be shared with anyone, $C5$.

Table 4.1. Privacy Rules (P) of the users as SWRL Rules

$P^{w_1}_{x_1}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, ?m), *hasMedium*(?pr, ?med), *hasMood*(?med, ?mood), *equal*(?mood,”depressed”),*includesPerson*(?med,`:x`) → *rejects*(`:x`,?pr), *rejectedBecauseOf*(?aud,?m), *rejectedIn*(?aud, ?pr)	$C1$
$P^{w_2}_{x_2}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, ?m), *isPartOfFamilyOf*(?m, `:x`), *hasMedium*(?pr, ?med),`Bar`(?loc), *includesLocation*(?med, ?loc), *includesPerson*(?med,`:x`) → *rejects*(`:x`,?pr), *rejectedBecauseOf*(?aud,?m), *rejectedIn*(?aud, ?pr), *rejectedIn*(?med, ?pr), *rejectedBecauseOf*(?med, ?loc)	$C2.1$
$P^{w_3}_{x_3}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, ?m), *isWorkRelatedOf*(?m, `:x`), *hasMedium*(?pr, ?med), `Beach`(?ctx), *isInContext*(?med,?ctx), *includesPerson*(?med,`:x`) → *rejects*(`:x`,?pr), *rejectedBecauseOf*(?aud,?m),*rejectedIn*(?aud, ?pr), *rejectedIn*(?med, ?pr), *rejectedBecauseOf*(?med, ?ctx)	$C3.1$
$P^{w_4}_{x_4}$:	*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)	$C3.2$
$P^{w_5}_{x_5}$:	*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)	$C4$
$P^{w_6}_{x_6}$:	*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)	$C5$
$P^{w_7}_{x_7}$:	*hasAudience*(?pr, ?aud), *hasMember*(?aud, `:y`) → *rejects*(`:x`, ?pr), *rejectedIn*(?aud, ?pr), *rejectedBecauseOf*(?aud, `:y`)	$C6$

(vii) `:x` does not want `:y`, a specific person, to see the post. This represents the concern $C6$.

We have decided to not express some of the privacy concerns ($C2.2$, $C7$ and $C8$) in our model since we were not able to use such rules. In $C2.2$, 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 $C7$, even 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 $C8$, 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 a future work.

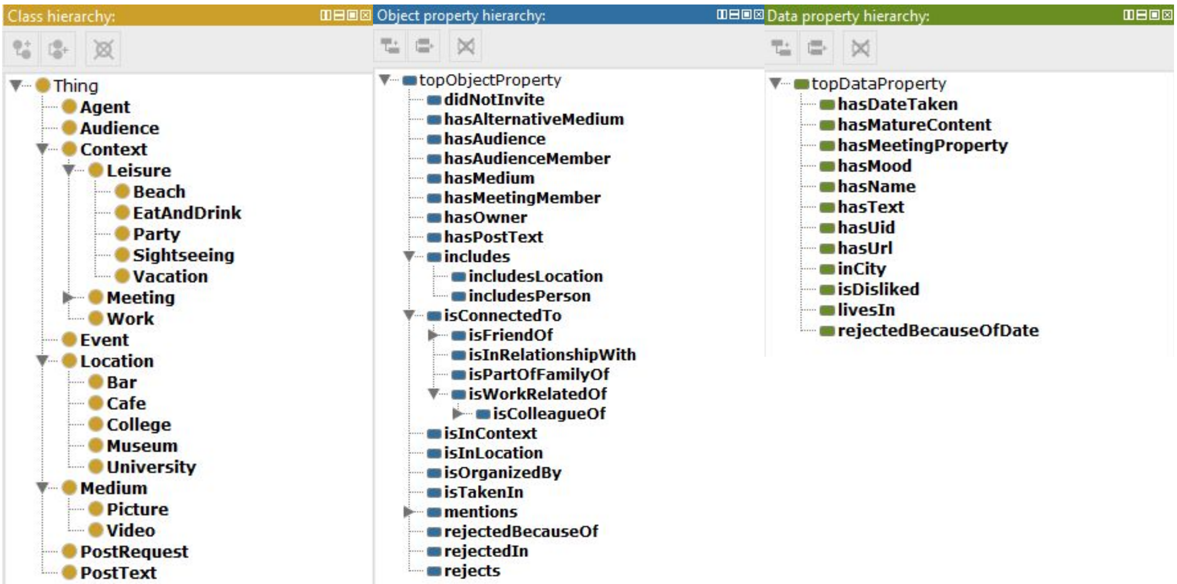
5. EVALUATION

In order to evaluate the effectiveness of our proposed mechanism, we need to have a working system and scenarios to test them. In this chapter, we firstly introduce our implementation of the system in Section 5.1. We explain our simulation environment which we use for creating different scenarios to run our strategies with in Section 5.2. We introduce the evaluation metric we use to compare between strategies in Section 5.3 and lastly we present the result of that comparison in Section 5.4.

5.1. Implementation

We implemented a system where agents can negotiate their privacy constraints. It is based on the work produced by Mester *et al.* [27]. Our system is implemented as a Java-based web application deployed using the Tomcat server. In this system, agents communicate with each other through RESTful web services in Spring framework. These web services enable agents to start a negotiation, evaluate the incoming requests and update points, if necessary. In addition to the primary system, we implemented a complementary system that demonstrates the steps of the negotiations carried on the main program to the users. For this system, we use JSP and HTML to output the results of a negotiation. We use MongoDB for database purposes.

We use ontologies to represent agents, their privacy preferences and relations with other agents. We take the ontology model created by Mester *et al.* as a base and build necessary concepts for our negotiation strategies upon it. These ontologies are created using Protégé [42] and they are kept in OWL format. Privacy rules are represented as SWRL rules [36] and the reasoning is done using Pellet reasoner [43]. Representing the social network with ontologies is advantageous since we can infer relations and privacy violations automatically. In addition to that ontologies enable us to negotiate semantically. The general outline of the ontologies (i.e., classes, object properties, data properties) need to be common for every agent. However the individuals and their relations as well as privacy rules may differ depending on the agent. The base concepts



Class hierarchy:

- Thing
  - Agent
  - Audience
  - Context
    - Leisure
      - Beach
      - EatAndDrink
      - Party
      - Sightseeing
      - Vacation
    - Meeting
    - Work
  - Event
  - Location
    - Bar
    - Cafe
    - College
    - Museum
    - University
  - Medium
    - Picture
    - Video
  - PostRequest
  - PostText

Object property hierarchy:

- topObjectProperty
  - didNotInvite
  - hasAlternativeMedium
  - hasAudience
  - hasAudienceMember
  - hasMedium
  - hasMeetingMember
  - hasOwner
  - hasPostText
  - includes
    - includesLocation
    - includesPerson
  - isConnectedTo
    - isFriendOf
    - isInRelationshipWith
    - isPartOfFamilyOf
    - isWorkRelatedOf
      - isColleagueOf
  - isInContext
  - isInLocation
  - isOrganizedBy
  - isTakenIn
  - mentions
  - rejectedBecauseOf
  - rejectedIn
  - rejects

Data property hierarchy:

- topDataProperty
  - hasDateTaken
  - hasMatureContent
  - hasMeetingProperty
  - hasMood
  - hasName
  - hasText
  - hasUid
  - hasUrl
  - inCity
  - isDisliked
  - livesIn
  - rejectedBecauseOfDate

Figure 5.1. Ontology model.

used in our system and their relations are explained in Section 2.1. Their visualization using Protégé can be seen in Figure 5.1 and Figure 5.2.

Agents represent people in the social network. They can have relationships with each other. Agents can be owners of post-requests (i.e., initiator agent), if they want to upload them. If they are tagged in a post-request (i.e., negotiator agent), they can reject it. A post-request has an audience and a medium. Audience includes agents who will see the post. Medium can be a picture or a video. A medium can have list of included agents; this indicates who are tagged in it. A media also has context, location and event info which can be empty. We represent privacy concerns of agents with SWRL rules as shown in Figure 5.3. For example the third rule in 5.3 states that `:user1223` does not want her bar located pictures to be seen by her family members.

We use OWL API [44] to work with ontologies in Java. Every class in the ontology model shown in Figure 5.1 is represented by Java classes. Object and data properties of the classes are represented by appropriate instances. For example a `PostRequest` has an `Audience` instance which includes a list of `Agents` that can see the post. An instance of an `Event` includes an `Agent` instance indicating the organizer of it. For the data properties we used types of objects Java offers. For example a `Medium` has a string that indicates the mood of it.

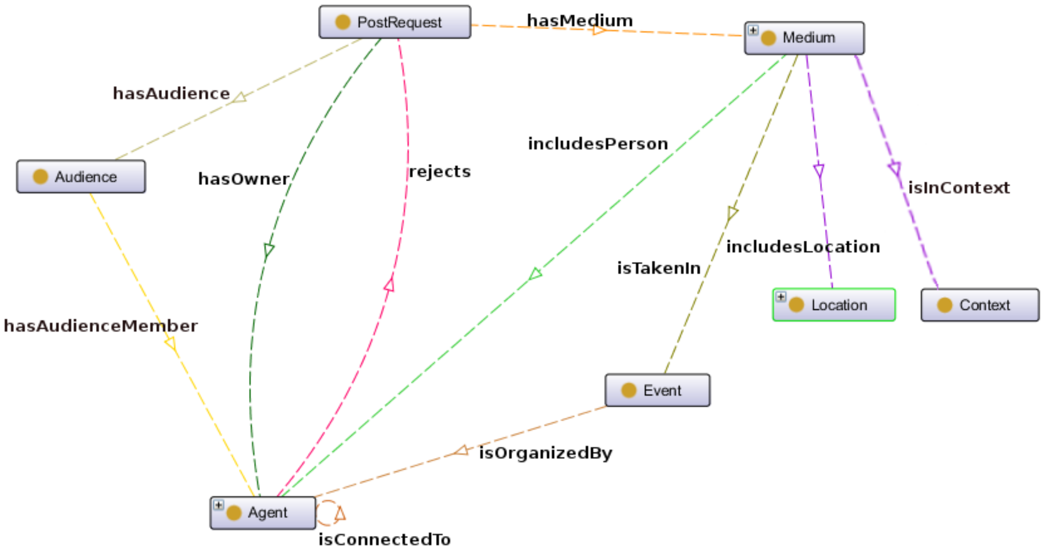
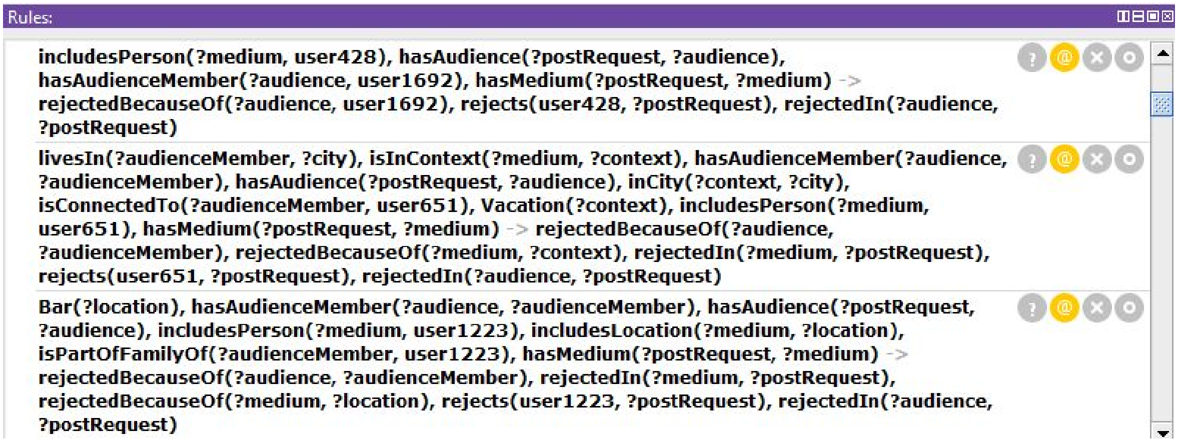


Figure 5.2. Ontology class relations.



includesPerson(?medium, user428), hasAudience(?postRequest, ?audience), hasAudienceMember(?audience, user1692), hasMedium(?postRequest, ?medium) -> rejectedBecauseOf(?audience, user1692), rejects(user428, ?postRequest), rejectedIn(?audience, ?postRequest)

livesIn(?audienceMember, ?city), isInContext(?medium, ?context), hasAudienceMember(?audience, ?audienceMember), hasAudience(?postRequest, ?audience), inCity(?context, ?city), isConnectedTo(?audienceMember, user651), Vacation(?context), includesPerson(?medium, user651), hasMedium(?postRequest, ?medium) -> rejectedBecauseOf(?audience, ?audienceMember), rejectedBecauseOf(?medium, ?context), rejectedIn(?medium, ?postRequest), rejects(user651, ?postRequest), rejectedIn(?audience, ?postRequest)

Bar(?location), hasAudienceMember(?audience, ?audienceMember), hasAudience(?postRequest, ?audience), includesPerson(?medium, user1223), includesLocation(?medium, ?location), isPartOfFamilyOf(?audienceMember, user1223), hasMedium(?postRequest, ?medium) -> rejectedBecauseOf(?audience, ?audienceMember), rejectedIn(?medium, ?postRequest), rejectedBecauseOf(?medium, ?location), rejects(user1223, ?postRequest), rejectedIn(?audience, ?postRequest)

Figure 5.3. SWRL rules.

When a user wants to share a picture of a friend, her agent prepares the post-request, adds appropriate audience and medium. It sends this request to the included person's web service with HTTP using REST. The included agent then takes this request and inserts it to its ontology to see if the SWRL rules of the agent are violated. We convert the Java object `PostRequest` to its ontological counterpart using OWL API. We insert every variable of the object to the ontology using class assertion, object property assertion and data property assertion. After that we use OWL API combined with Pellet reasoner to detect privacy violations that fires the SWRL rules. A SWRL rule fires when the body of it is fulfilled, resulting the head of it to get asserted. If there is a fired rule, then the included agent prepares rejection reason depending on the strategy they use and sends it to the initiator agent. The initiator agents also decide the next steps according to the strategy they use. Since the negotiator agent cannot add any audience members to the post-request, the initiator agent does not need to look for any privacy violations. That is why the initiator agent does not use their ontologies to decide on a next move.

5.2. Simulation Environment

Simulations enable us to mimic the user's behavior and carry out multiple negotiations so that we can measure if the mechanism actually preserves users' privacy. For this reason, we evaluate the performance of our approach using simulations.

Our 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 [45]. Each agent has at least one connection in the network. The network allows agents to be connected to each other with three different relations: family, work, and friend. Each agent has two or three privacy rules. These rules are created according to the user study we have explained in Chapter 4. We have total of seven rules as explained in Section 4.3. The rules are assigned to agents randomly as well as the weights of these rules. The weights can be integers between 1 to 10, inclusive.

The points are defined pairwise as explained in Section 3.3. Therefore, we have created 25 agent pairs out of these 50 agents and we have examined their interactions. These pairs are friends with each other and every agent uploads posts tagging their pair. For each pair, two post-requests are generated so that each agent in the pair is an initiator in one post-request and a negotiator in the other. These post-requests are created according to the pair's privacy rules to ensure that each post-request conforms to the privacy rules of the initiator and violates the negotiator's. In this way, we force agents to start a negotiation and then we observe the results of their interactions. Every privacy rule of the negotiator agent is violated so that we can compare the negotiation strategies better. For example, the difference of conduct between GEP and MP cannot be shown if only one rule is violated by the post-request. This stems from the fact that GEP sends rules one-by-one based on their importance to the initiator agent while MP sends them all in the first run. When there is only one violated rule, there is no difference between GEP and MP. We use these post-requests to conduct the simulation.

5.3. 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 [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 since it does not leave any agent at a disadvantage. To support this intuition, we propose a new evaluation metric *Scaled Product (SP)* that gives penalty when utilities are widely apart. We calculate this metric as shown in Equation (5.1). *SP* metric rewards utilities that are close to each other by considering the difference of utilities.

$$SP(u^{in}, u^{ng}) = u^{in} \times u^{ng} \times (1 - |(u^{in} - u^{ng})|) \qquad (5.1)$$

In this equation, u^{in} represents the utility of the initiator agent while 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 scaled product 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, scaled product will decrease compared to the product of utilities. The maximum value that a scaled product can have is 1, and this happens when both u^{in} and u^{ng} are 1.

5.4. Comparison

We have proposed mainly two strategies; GEP and MP. However these strategies were one-shot negotiations, which means that they disregard previous interactions while doing the current negotiation. That is why we implemented a trade-off mechanism that enables users to negotiate considering past interactions, RP, which can be considered as a layer upon the GEP and MP. We represent these strategies as RP over GEP (RPG) and RP over MP (RPM). Currently, most online social networks allow a user to post content without considering the opinion of others that might be affected by this content. For this reason, in addition to these strategies we also add *Uploader Overrides* (UO) [46] to our comparison. We use *Scaled Product (SP)* as our evaluation metric. SP's maximum value is one as explained in Section 5.3, however this maximum value cannot be reached with our scenarios. Since every post-request is designed to create conflict between agents, it is impossible for both agents to obtain a utility of 1. Comparing our strategies' scaled products to one (no conflict situation) misrepresents their performance. That is why we found the best possible negotiation for every scenario according to SP. We created possible post-requests by removing violations from the audience one-by-one and calculated the SP. Maximum of these SPs is considered as the best possible outcome and the SPs of our strategies are adjusted according to this value. For example if the best possible SP is 0.75 and the SP 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.

We have compared these strategies according to two different sharing patterns of the users: *in-order sharing* and *consecutive sharing.* In *in-order sharing* pattern, agent pairs share posts one after another. For example, consider an agent pair that includes Alice and Bob. First, Alice shares a post including Bob and then Bob shares a post about Alice. In *consecutive sharing* pattern, an agent shares posts about the other agent a specific number of times. For example, Alice shares three consecutive posts about Bob while Bob does not share any post about Alice.

To capture the features of Reciprocal Privacy, we need to have continuous posts since RP considers previous interactions of agents to negotiate. Therefore, we run these patterns five times and report the results. Agents are chosen randomly from our simulations as described in Section 5.2.

In-Order Sharing : Each agent is associated with another agent in the simulation. For each pair, agents share posts in order (i.e., one after another). After five runs, we average the utilities and compare the results of strategies according to *Scaled Product (SP)* evaluation metric.

We also split the results into three categories depending on strictness of the negotiating users. Strictness here is used in the sense of unwillingness to show the post to the majority of the audience. At first category we compare the average of all 25 pairs. The second category is the case where none of the negotiating users is strict. Lastly the third category is the case where one of them is strict. We represent these three categories as Overall, Non-strict and Strict respectively in the following figures.

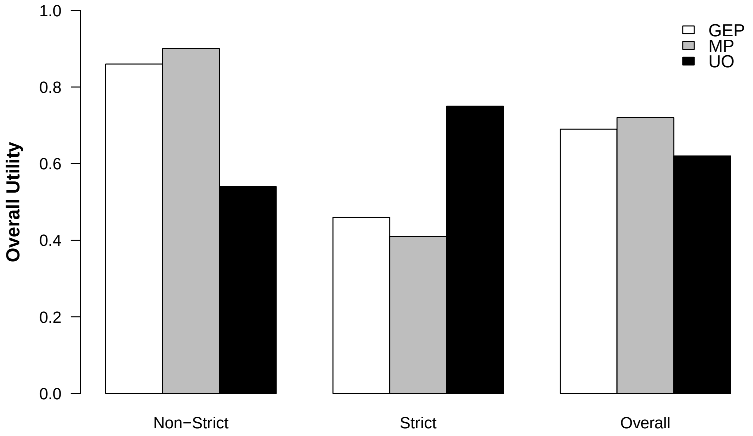


Figure 5.4. In-order sharing, utility comparison of strategies.

In Figure 5.4, we can see the results for strategies that are not dependent on the previous interactions (GEP, MP and UO) with the in-order sharing pattern. If we look at the overall results, all three of them perform closely. However if we look at the case where none of the users is strict, GEP and MP overpower UO. If one of the users is strict then UO performs noticeably better than our static strategies. This is the result of GEP and MP trying to satisfy utility thresholds in all situations. Both in GEP and MP, if there is no post-request where both of the agents satisfy their utility thresholds then the post is not shared. If one or two of the agents are strict, it is likely that agents cannot find a post-request that satisfies both the negotiator and the initiator. Likewise, if none of the users is strict then it is likely that agents can find a common ground they both agree, which is better than UO.

Even though we expected RP layer to improve our strategies, after we observed the results of our strategies with our simulation environment, we noticed that this was not always the case. We show the comparison between strategies RPG, GEP, RPM and MP in Figure 5.5. We see that in some cases RPG performed better than GEP and performed worse in other cases. This is because of the limit on the negotiation iteration number. Every initiator agent has a negotiation iteration number they tolerate, if a negotiation exceeds the maximum number of iterations allowed then initiator agent terminates the negotiation, posting the last post-request offered by the initiator. For

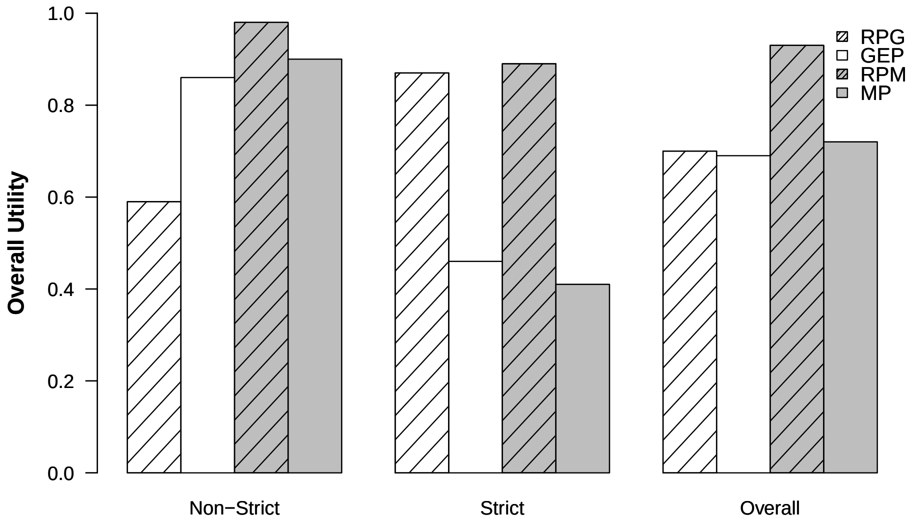


Figure 5.5. In-order sharing, effects of reciprocity.

strict negotiator agents that want majority of people removed from the audience, this limit was exceeded and the negotiation halted halfway. This resulted in low scaled product values since the common ground could not be reached. This was especially the problem with RPG compared to RPM because of the differences of revising the post-request. In RPG, initiator agent removes people who violates the privacy of the negotiator agent from the audience one-by-one to reach a common ground. While in RPM, initiator removes all violations at first and then adds people one-by-one at each iteration. Since in RP mechanism, initiator agent always offers post-requests that satisfies its' utility threshold while the utility for the negotiator agent may get compensated with points, RPM has better outcome for both agents compared to RPG if the negotiation halts halfway because the utility of negotiator agent is higher. The reason why the iteration limit is not as much of a problem for both GEP and MP is because they revise post-requests on a violated rule level while RP layer revises them on a violated audience member level. For example if a negotiator agent has one violated rule that applies to three audience members; using GEP would result all three audience members to be removed in one iteration while RPG would remove them one-by-one in three iterations. When the negotiation iteration number is not exceeded, RP layer usually performs better than GEP or MP counterparts.

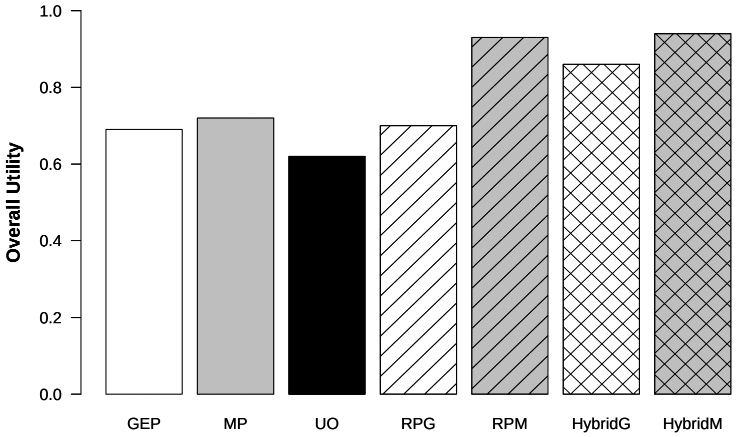


Figure 5.6. In-order sharing, results of all strategies.

That is why we implemented another strategy that combines the best of the RPG and RPM with GEP and MP respectively. We call the combination of GEP and RPG as HybridG and the combination of MP and RPM as HybridM. For these strategies, we firstly run GEP or MP, and see if the negotiation is reached. If there is no agreement then RPG or RPM are invoked respectively, to find a common ground. If there is an agreement when we run GEP or MP, we compensate 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 the Equation 3.4. If the points of the initiator agent are not enough then it tries to find a new post-request 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. This strategy performs statistically equal or better than other strategies we have (RPG, GEP, RPM, MP) as seen on Figure 5.6.

We use in-order sharing for every strategy we have and compare the results of SPs for all 25 pairs to see which strategies perform best. In Figure 5.6, we can see the average results of all strategies we implemented for the in-order sharing pattern. HybridM and RPM performs nearly equal while HybridG follows closely. For the reasons we explained above, RPG performs poorly compared to RPM when agents are strict and this is reflected in the overall results. This also affects the performance of

HybridG since it is a combination of RPG and GEP.

Another reason why RPG performs worse than RPM is the point shortage of one agent in a pair. In RPG, when one agent does not need any points to upload and its pair uses all its points earlier, the latter agent has no points to use in future posts. Hence, agents could not reach an agreement in last posts, which are not shared. That results in lower average utilities since not sharing anything means the negotiator will have a utility one while the initiator will end up with a utility 0. This is usually not the case with the RPM because as we explained in Section 3.2, MP strategy usually favors the negotiator agent. That is why the surplus of utility for the initiator agent is small, which results in minimal point transfer between the agents. Hence the probability of point shortage is really small in RPM compared to RPG which leads to better performance.

All of our strategies performs better than UO which is generally the default strategy online social networks use.

Consecutive Sharing : The agents are part of the same pairs as in the previous pattern. In this pattern, each agent shares posts about the other agent five times consecutively. After five runs, we average the utilities and compare the results of RP and UO according to *Scaled Product (SP)* evaluation metric.

After sharing the result for in-order sharing pattern, we compare the strategy performance for the consecutive sharing. For every strategy, we run consecutive sharing with 50 simulation agents. In Figure 5.7, we see the results for the strategies with the consecutive sharing pattern. Same as the in-order sharing pattern, HybridM and RPM perform nearly equal while HybridG follows closely. Similarly RPG performs poorly because of the reasons explained above; exceeding negotiation iteration number and point shortage. However if we compare the results for both of the sharing patterns, we see that RPG performs especially poorly with consecutive sharing. This is because the fact that point shortage during negotiations usually appears earlier than the in-order sharing pattern. Since the agent shares five posts consecutively and the point

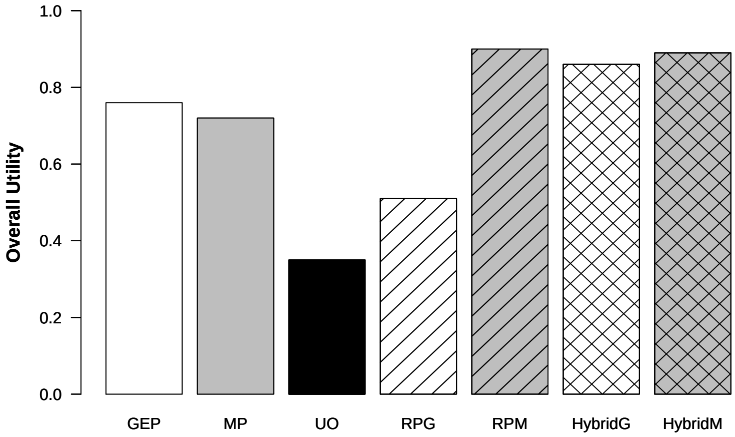


Figure 5.7. Consecutive sharing, results of all strategies.

transfer is only from the initiator agent to negotiator agent, initiator does not have any opportunities to gain points in between the postings. Hence, it exhausts its points earlier compared to the case of the in-order sharing where the agent has the possibility of gaining points.

Another strategy that performs especially poorly than its in-order sharing counterpart is UO. We see that while UO performed closely with GEP and MP previously, now it performs considerably lower than them. This is because only one user of the user pairs shares a post for consecutive sharing while both users of the pair share a post for in-order one. Even though results are not good for the posts individually, when combined they compensate each other and increase the scaled product. For example, we have a pair of agents, a_1 and a_2, and they both share one post about each other. a_1 posts and the utilities of a_1 and a_2 are 1 and 0.2 respectively, the SP of this case would be 0.04. a_2 posts and the utilities of a_1 and a_2 are 0.4 and 1 respectively, the SP of this case would be 0.16. However when we combine these posts, average utilities of a_1 and a_2 will be 0.7 and 0.6 respectively, which gives the SP of 0.38. We can see that low scaled product individual posts can increase their performance when they are shared back-to-back.

When we compare the overall results, every strategy we implemented performed better than UO in average. For GEP and MP, that is because we try to find compromises between the agents to find a better outcome. The strategies that use RP layer usually performed better than the GEP, MP and UO. The reason is that the strategies using RP layer take previous interactions into consideration to negotiate about the current post-request, unlike GEP, MP and UO that always gives the same result. For example, if the initiator agent shares a post about the negotiator agent using UO, the initiator agent will always have utility 1 and the negotiator agent may have a utility as low as 0. This will be the result of every post uploaded using UO. In strategies that use RP layer, the first negotiation might have the same results with UO. However, future negotiations will take this result into consideration and will try to compensate the negotiator agent. This leads them to give more balanced utilities in the long run hence they perform better in both patterns.

Between the strategies that uses RP layer, HybridG and HybridM perform similarly or better than RPG and RPM. Since they combine the best outcomes of the GEP and MP with RPG and RPM, it is expected that they perform better than all other strategies.

6. DISCUSSION

We propose a system to enable users to negotiate their privacy in online social networks. Our system has a hybrid negotiation architecture that uses both semantic knowledge and privacy rules as well as utility functions while making decisions. We develop two negotiation strategies and one incentive mechanism that can be combined with the existing strategies to create new strategies.

Our developed incentive mechanism (i.e., reciprocal privacy), is based on the fact that protecting another user's privacy will increase the likelihood of preserving our privacy. To measure the outcomes of the negotiations we propose a new metric that emphasizes the reciprocity principle. We have conducted a user study and incorporated the results to our evaluation that uses small network to simulate real-life. Our results show that both our strategies coupled with the incentive mechanism enables agents to protect their privacy in the long term successfully.

Even though negotiation techniques are commonly used in e-commerce systems, its use in privacy negotiation in the context of social networks is fairly new. The negotiation mechanism that is proposed by Such and Rovatsos enables users to solve privacy conflicts [28]. These conflicts are settled by compromises made by both sides. Every privacy policy concerns an item and these policies are created manually by agents. Policies identify the users that are allowed to see the post and the users are not. Every item has an intended audience and for every member of this audience, access rights to the item is decided by negotiating agents' policies. Every negotiating agent creates an action vector of 0's (deny) and 1's(accept) that has the size of the intended audience indicating their permissions. Any mismatch in the action vectors of the negotiating agents denotes a conflict. If there is a conflict, then the system moves to the negotiation step. Every agent has utility functions to evaluate action vectors and the objective of the negotiation is to find an action vector that maximizes the product of the utilities for the agents. To find this action vector, one-step negotiation is used where each agent proposes a solution that will maximize their own utility and as well as the product of the utility of both agents. The solution with the higher utility is chosen. For agents to propose a solution the utility functions of all agents must be disclosed. In our approach agents share rejection reasons to negotiate, they are not required to disclose their utilities or utility functions to each other.

In another work, Such and Criado uses the same action vector approach to detect privacy conflicts which are defined as the difference of opinion between negotiating agents regarding the audience of a post [46]. To solve these conflicts a mediator takes policies of the agents and tie-strengths with their friends to calculate item sensitivity so they can estimate the willingness to concede to achieve an agreement. Item sensitivity here is defined as the importance of an item to an agent. An agent will be less willing to share an item if it has high sensitivity for the agent. Also an item can have different sensitivities depending on the agent. Both of these approaches include disclosing sensitive information to another agent or a mediator. This might result in a privacy breach. In our system there is no third-party decision maker. Each agent only knows its utility functions; other agents' functions are hidden from them.

Squicciarini *et al.* propose a tool called CoPE that enables collaborative privacy management between people to control their private information shared in social networks [11]. The are three groups of users: content-owners (who create content), co-owners (who are tagged in a content) and content-viewers (who can view the content). For every content, its content-owner and co-owners decide on the content-viewers before sharing it. To reach this decision content-owner and co-owners each choose a group of users that can see the post from the three options; "some-friends", "public" or "co-owner only". If majority of stakeholders choose "co-owner only" then the content is only open to the stakeholders. Otherwise each agent's privacy preference is respected. They add the permitted users to the content-viewers set and remove unwanted users from it according to the preferences of each stakeholder. In this way it is ensured that there is no privacy conflict. However, as a result the content-viewers can be reduced to a small set. This is not beneficial from the point of the content-owner. Our approach aims to satisfy both content-owner and co-owners by enabling them to negotiate and reach a common-ground.

All three of the previously mentioned systems negotiate considering the current situation only. The result is not affected by the past interactions and outcome is always the same. This can be disadvantageous in the case where one of the negotiating agents' situation is considerably worse than the other continuously. For example a negotiation could result an agent with a utility of 0.3 and the other with 0.8. While this outcome may be tolerable for once, it would leave the former agent in a disadvantageous situation if the same type of result occurred continuously. Hence it is important to consider previous interactions while negotiating so that no particular agent is considerably worse than others in the long run.

There are few approaches that use past interactions in the current negotiation including our approach. Squicciarini *et al.* propose a method for collective privacy management using an incentive mechanism [30]. Every agent has a predetermined global credit that they can use while negotiating. Agents can increase their credits by sharing a content, getting tagged in a content and granting co-ownerships to agents tagged in the content. They use Clarke-Tax method to encourage the truthfulness so the social utility is maximized. In this method every agent simultaneously decides on a credit to invest into their privacy setting getting selected. The information about an agent's investments are not disclosed to other agents. The setting that has the most total investment is selected and the agents who were pivotal in this decision gets taxed, meaning they lose their previously offered credit. This approach resembles our incentive mechanism where we use points to convince negotiator agent to compromise. However in their work credits are global and can be used for any agent. In our reciprocal privacy, points are defined pair-wise and points against one agent cannot be used against another agent. Since reciprocity is based on pairs, it is meaningful that our points are not global.

Another work that uses past interactions in the current negotiation is the work proposed by Ramchurn*et al.* [31]. 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. In this system rewards are defined as the concessions of the opponent whether it is in this game or future games. Agents have utility functions to evaluate offers and every agent has a target utility they want to achieve in a negotiation. This approach only considers negotiations with two games. When an agent accepts to concede in the first game by taking a reward, they do so to fulfil their utility target in the second game by making their opponent concede. In the end, rewards are defined in the span of two games and they are in the form of concessions by negotiating agents. In our approach we do not limit the number of interactions and consider every one of them while conducting the current negotiation.

There are approaches that choose argumentation for privacy protection instead of negotiation. In the system proposed by Kökciyan *et al.* argumentation is used to protect the privacy of users in online social networks [29]. Agents can use the arguments from their own ontology or can ask other agents to provide them. In PriArg, agents share their privacy constraints with other agents to continue argumentation unlike our approach. In our approach we do not disclose the privacy policies. Different from PriArg, we also keep track of previous interactions while executing the current interaction to reach an overall balanced outcome for agents. Lastly in PriArg, there are two outcomes: share or do not share. In our system we enable agents to reach a common ground where they can share a modified version of the original post.

In addition to the systems that solves conflicts automatically, there are systems that enable users to solve them manually. Wishart *et al.* developed a tool called PriMMA-Viewer that allows co-owners to edit the privacy policy of a content collaboratively [47]. When a user wants to share a post that includes other people on social network, they create a privacy policy indicating who are allowed to see the post and who are not. They also invite included people to edit the policy. There are two types of conditions on privacy policies: Strong condition and weak conditions. Users can modify the policy in three ways: add conditions, remove weak conditions, remove their own strong conditions. An agent cannot remove a strong condition of another agent but can remove or modify a weak one. The rank of importance between conditions resembles our system as we rank privacy rules as well by giving them weights; however high weighted privacy rules are not guaranteed to be enforced in every negotiation.

The disadvantage of enforcing every strong condition is over restriction; there may be no one left in the audience. Some users can put too many strong conditions maliciously and create conflicts. In this case the approach use majority voting between co-owners to remove these users. All of these processes are done manually hence takes more time compared to an automatic system. Negotiation is automated in our system and we consider the content-owner's wish to share the post with the original audience and prevent the post from getting overly restricted.

The framework introduced by Carminati and Ferrari specifies and enforces collaborative access control policies [48]. It is a decentralized system where every user has its access rules, resources and collaborative security policies in their server. An access rule has a resource name as well as access conditions and it specifies the features of the users that can see the resource. A collaborative security policy indicates the stakeholders of the resource its connected to (i.e., tagged users) and the decision mode which can be "One","Majority" and "All". "One" means at least one stakeholder needs to give permission for requester to see the resource. "Majority" means more than half of the stakeholders need to give permission to the requester. Lastly "All" means that the stakeholders need to give permission to the requester unanimously. Even though the system is decentralized there is also a Social Manager that has the relationships, access rights and collaborative policies of everyone. Let's say the user *usr* wants to view a resource, it requests access from the owner. Social Manager finds the stakeholders of the resource and collects their access decisions. After that Social Manager evaluates the results with accordance to the type of the collaborative security policy of the resource and sends the outcome to the owner. Owner shows the picture to *usr* if there is an approval. Our work focuses on user privacy where the approach of this work focuses on resource protection and their access rights. Also the users in this system provide feedback themselves or get help from the Social Manager, in our approach agents automatically negotiate on behalf of the users.

Hu *et al.* proposes a multiparty access control model that is represented by ASP (i.e., declarative programming language) for online social networks [49]. In the usual control mechanisms the person who uploads the post is the controller. However a user can upload posts about her friends and these friends have a right to define who can see this post. In this work there are four types of controllers; owner (who uploads something about herself), contributor (who upload something about her friend), stakeholder (who is tagged in the post) and disseminator (who shares a friend's post to her own space). When there is an access request from a user, these controllers evaluate the request according to their privacy policies. These evaluations also consider the sensitivities of the items, high sensitivity means that the post is important to a user. The controllers individually decide whether the requester should see the post or not and these decisions are aggregated. The authors propose various methods for the decision aggregation. One of them uses sensitivity as a threshold. They average the decision values (can be 1 or 0) and sensitivities from controllers. If the average of decision values is bigger than the average of sensitivities then permission is granted to the requester. In another method they use voting schemes and let the owner of the post to choose the voting strategy (i.e., owner-overrides, full-consensus-permit and majority-permit). They use sensitivity scores to indicate the importance of an item. We use weights for a similar purpose, however in our system we give importance to the rules, not the items. In this work policies are created by users for every shared content whereas our users create general rules and a rule applies to every content that has the specific features stated by the rule.

Another system that detects privacy violations is proposed by Kökciyan and Yolum [32]. They represent online social networks as agent-based social networks with their proposed meta-model. PriGuard is implemented using this meta-model alongside with commitments to catch privacy violations. Commitments generally consist of four elements; debtor, creditor, antecedent and consequence. Debtor is responsible of ensuring the consequence if the creditor does the antecedent. In the case of this work the debtor is usually the social network provider. Every user has privacy requirements as commitments and these commitments are represented semantically using their DL-based model. The violations are detected by creating the negation of a privacy requirement. If the negation holds true on the system then that means the privacy of the agent is violated. They use an ontology to represent the social network like our approach. However our ontology has context information as well as the negotiation protocol in it. Another difference is that we aim to prevent privacy violations before they occur whereas this approach detects violations after they happen.

As a future work it would be interesting to incorporate trust relations into the utility functions. The idea is that agents may be willing to compromise more for agents that they trust. They may believe that their trusted friend is more likely to not do anything to harm them compared to an acquaintance. In our work, points are used as a compensation for the privacy compromises of an agent. However, they can as well be used as a reward (e.g.,for sharing a good post). This can be added in the future. Another future addition could be enabling users to trade-off between privacy and utility. Our current work does not disclose privacy rules of an agent to another agent. This sometimes results in a suboptimal solution. When having the optimal solution is not necessary, protecting the privacy may be more important for the agent. However if having the optimal solution is crucial, it may be more beneficial to disclose private information. For example, a patient will share her medical records with her doctors, if the doctors have a possibility to find a cure even though she usually keeps her health records private [50]. Hence it is important to understand in what situations such a compromise is preferred. Modelling a trade-off between utility and privacy protection would mimic real-life better.

REFERENCES

1. Garton, L., C. Haythornthwaite and B. Wellman, “Studying online social networks”, *Journal of Computer-Mediated Communication*, Vol. 3, No. 1, pp. 0–0, 1997.

2. Ellison, N. B. *et al.*, “Social network sites: Definition, history, and scholarship”, *Journal of Computer-Mediated Communication*, Vol. 13, No. 1, pp. 210–230, 2007.

3. Gross, R. and A. Acquisti, “Information revelation and privacy in online social networks”, *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pp. 71–80, ACM, 2005.

4. *Facebook Web Site*, `http://www.facebook.com`, accessed at December 2016.

5. *LinkedIn Web Site*, `https://www.linkedin.com/`, accessed at December 2016.

6. *Spritzr Web Site*, `https://spritzr.com/`, accessed at December 2016.

7. *Social Networking Fact Sheet*, 2013, `http://www.pewinternet.org/fact-sheets/social-networking-fact-sheet/`, accessed at December 2016.

8. *Nearly One-Third of the World Will Use Social Networks Regularly This Year*, `http://www.emarketer.com/Article/Nearly-One-Third-of-World-Will-Use-Social-Networks-Regularly-This-Year/1014157`, accessed at December 2016.

9. Warren, S. D. and L. D. Brandeis, “The right to privacy”, *Harvard law review*, pp. 193–220, 1890.

10. Posner, R. A., *The economics of justice*, Harvard University Press, 1983.

11. Squicciarini, A. C., H. Xu and X. L. Zhang, “CoPE: Enabling Collaborative Privacy

Management in Online Social Networks", *Journal of the American Society for Information Science and Technology*, Vol. 62, No. 3, pp. 521–534, 2011.

12. Nakashima, E., *Feeling Betrayed, Facebook Users Force Site to Honor Their Privacy*, 2007, http://www.washingtonpost.com/wp-dyn/content/article/2007/11/29/AR2007112902503.html, accessed at December 2016.

13. Maternowski, K., *Campus police use Facebook*, 2006, https://badgerherald.com/news/2006/01/25/campus-police-use-fa/, accessed at December 2016.

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

15. Shachtman, N., *Exclusive: U.S. Spies Buy Stake in Firm That Monitors Blogs, Tweets*, 2009, https://www.wired.com/2009/10/exclusive-us-spies-buy-stake-in-twitter-blog-monitoring-firm/, accessed at December 2016.

16. Carr, A., *Meet the Big Brother Screening Your Social Media for Employers*, 2010, https://www.fastcompany.com/1692172/meet-big-brother-screening-your-social-media-employers, accessed at December 2016.

17. Grasz, J., *Forty-five Percent of Employers Use Social Networking Sites to Research Job Candidates, CareerBuilder Survey Finds*, 2012, http://www.careerbuilder.com/share/aboutus/pressreleasesdetail.aspx?id=pr691&sd=4/18/2012&ed=4/18/2099, accessed at December 2016.

18. Stewart, M. G., *How giant websites design for you (and a billion others, too)*, 2014, https://www.ted.com/talks/margaret_gould_stewart_how_giant_websites_design_for_you_and_a_billion_others_too, accessed at December 2016.

19. Lampinen, A., V. Lehtinen, A. Lehmuskallio and S. Tamminen, "We're in it to-

gether: interpersonal management of disclosure in social network services", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3217–3226, ACM, 2011.

20. Sleeper, M., R. Balebako, S. Das, A. L. McConahy, J. Wiese and L. F. Cranor, "The Post That Wasn'T: Exploring Self-censorship on Facebook", *Proceedings of the 2013 Conference on Computer Supported Cooperative Work (CSCW)*, pp. 793–802, ACM, 2013.

21. Jennings, N. R., P. Faratin, A. R. Lomuscio, S. Parsons, M. J. Wooldridge and C. Sierra, "Automated negotiation: prospects, methods and challenges", *Group Decision and Negotiation*, Vol. 10, No. 2, pp. 199–215, 2001.

22. Wong, T. and F. Fang, "A multi-agent protocol for multilateral negotiations in supply chain management", *International Journal of Production Research*, Vol. 48, No. 1, pp. 271–299, 2010.

23. An, B., V. Lesser, D. Irwin and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing", *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pp. 981–988, International Foundation for Autonomous Agents and Multiagent Systems, 2010.

24. Narayanan, V. and N. R. Jennings, "An adaptive bilateral negotiation model for e-commerce settings", *Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pp. 34–41, IEEE, 2005.

25. Lopes, F., C. Ilco and J. Sousa, "Bilateral negotiation in energy markets: Strategies for promoting demand response", *2013 10th International Conference on the European Energy Market (EEM)*, pp. 1–6, IEEE, 2013.

26. Hemaissia, M., A. E. F. Seghrouchni, C. Labreuche and J. Mattioli, "Cooperation-based multilateral multi-issue negotiation for crisis management", *Rational, Ro-*

bust, and Secure Negotiations in Multi-Agent Systems, pp. 81–100, Springer, 2008.

27. Mester, Y., N. Kökciyan and P. Yolum, "Negotiating Privacy Constraints in Online Social Networks", F. Koch, C. Guttmann and D. Busquets (Editors), *Advances in Social Computing and Multiagent Systems*, Vol. 541 of *Communications in Computer and Information Science*, pp. 112–129, Springer International Publishing, 2015.

28. Such, J. M. and M. Rovatsos, "Privacy Policy Negotiation in Social Media", *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, Vol. 11, No. 1, pp. 4:1–4:29, 2016.

29. Kökciyan, N., N. Yaglikci and P. Yolum, "An Argumentation Approach for Resolving Privacy Disputes in Online Social Networks", *ACM Transactions on Internet Technology (TOIT)*, 2017, to appear.

30. Squicciarini, A. C., M. Shehab and F. Paci, "Collective privacy management in social networks", *Proceedings of the 18th International Conference on World Wide Web*, pp. 521–530, ACM, 2009.

31. Ramchurn, S. D., C. Sierra, L. Godo and N. R. Jennings, "Negotiating using rewards", *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 400–407, ACM, 2006.

32. Kökciyan, N. and P. Yolum, "PriGuard: A Semantic Approach to Detect Privacy Violations in Online Social Networks", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 10, pp. 2724–2737, 2016.

33. McGuinness, D. L., F. Van Harmelen *et al.*, "OWL web ontology language overview", *World Wide Web Consortium recommendation*, Vol. 10, p. 10, 2004.

34. Gruber, T. R., "Toward principles for the design of ontologies used for knowledge sharing?", *International journal of human-computer studies*, Vol. 43, No. 5, pp.

907–928, 1995.

35. Schmidt, A., M. Beigl and H.-W. Gellersen, “There is more to context than location”, *Computers & Graphics*, Vol. 23, No. 6, pp. 893–901, 1999.

36. Horrocks, I., 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*, Vol. 21, p. 79, 2004.

37. Calikli, G., M. Law, A. K. Bandara, A. Russo, L. Dickens, B. A. Price, A. Stuart, M. Levine and B. Nuseibeh, “Privacy dynamics: Learning privacy norms for social software”, *Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pp. 47–56, ACM, 2016.

38. Kepez, B. and P. Yolum, “Learning Privacy Rules Cooperatively in Online Social Networks”, *Proceedings of the 1st International Workshop on AI for Privacy and Security, PrAISe@ECAI 2016, The Hague, Netherlands, August 29-30, 2016*, pp. 3:1–3:4, 2016.

39. Gouldner, A. W., “The norm of reciprocity: A preliminary statement”, *American sociological review*, pp. 161–178, 1960.

40. Kunz, P. R. and M. Woolcott, “Season’s greetings: From my status to yours”, *Social Science Research*, Vol. 5, No. 3, pp. 269–278, 1976.

41. *The pictures and the guideline of our user study*, `http://mas.cmpe.boun.edu.tr/negotiation`, accessed at December 2016.

42. *Protégé*, `http://protege.stanford.edu/`, accessed at November 2016.

43. Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur and Y. Katz, “Pellet: A practical OWL-DL reasoner”, *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 5, No. 2, pp. 51–53, 2007.

44. Horridge, M. and S. Bechhofer, “The OWL API: A Java API for OWL Ontologies”, *Semantic Web*, Vol. 2, No. 1, pp. 11–21, 2011.

45. Leskovec, J. and J. J. Mcauley, “Learning to Discover Social Circles in Ego Networks”, F. Pereira, C. Burges, L. Bottou and K. Weinberger (Editors), *Advances in Neural Information Processing Systems 25*, pp. 539–547, Curran Associates, Inc., 2012.

46. Such, J. M. and N. Criado, “Resolving Multi-Party Privacy Conflicts in Social Media”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 7, pp. 1851–1863, 2016.

47. Wishart, R., D. Corapi, S. Marinovic and M. Sloman, “Collaborative Privacy Policy Authoring in a Social Networking Context”, *Proceedings of the IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pp. 1–8, Washington, DC, USA, 2010.

48. Carminati, B. and E. Ferrari, “Collaborative access control in on-line social networks”, *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pp. 231–240, Oct 2011.

49. Hu, H., G.-J. Ahn and J. Jorgensen, “Multiparty access control for online social networks: model and mechanisms”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, No. 7, pp. 1614–1627, 2013.

50. Bilogrevic, I., K. Huguenin, B. Agir, M. Jadliwala, M. Gazaki and J.-P. Hubaux, “A machine-learning based approach to privacy-aware information-sharing in mobile social networks”, *Pervasive and Mobile Computing*, Vol. 25, pp. 125–142, 2016.